Integrating Service-Oriented Paradigm Into Introductory IS Courses

Billy B. L. Lim
bllim@ilstu.edu

Chu J. Jong
cjong@ilstu.edu
School of Information Technology
Illinois State University
Normal, IL 61790-5150
USA

Abstract

Web services and SOA (Service-Oriented Architecture) are two buzzwords that have received tremendous amount of attention in the software industry in recent years. While these technologies have been incorporated in many industries in the IT market place, they are only beginning to appear in the academia, primarily in upper division and graduate curricula. In this paper, the belief that Web services and SOA technologies can and should be introduced early in an IS curriculum is shared. Several scenarios that Web services / SOA can be integrated into introductory IS courses to make them more interesting and more importantly, make the students better prepared for upper division classes and for the industry upon graduation are presented. These scenarios can be incorporated without compromising the core materials presently covered in many IS curricula.

Keywords: Web services, SOA (Service-Oriented Architecture), SaaS (Software as a Service), Teaching Introductory IS, Emerging Trends in IS Curricula

1. INTRODUCTION

The computer industry has advanced tremendously in the past several decades and has impacted our lives greatly in its advancement. Many would agree that when comparing the progress of hardware versus software, the former has advanced more noticeably than the latter. This is partly because there has been lack of standards for developing software and thus it is extremely difficult to reuse software components and integrate disparate systems to form a larger one to solve an organization's problem.

Object-oriented (OO) technology has been utilized to address the above with some success over the years as a new software development paradigm. However, while OO technology has addressed some of the issues such as reuse, it lacks interoperable, cross-platform support for component integration. Also, in today's Internet/Web world, where security is of utmost importance and firewall is prevalently used, OO technology also lacks a standard protocol where communication between geographically and technologically diverse objects can be established in a firewall-friendly manner.

Now, the software industry is embracing another wave of change in software development technologies. This time it involves service-oriented architecture (SOA), particularly

the use of Web services to speed up application development and reduce costs to access data on disparate systems. Web services (Benfield, 2001; Curbera, 2002; Dyck , 2001; Kiely, 2001; Lim, 2003) and SOA (Colan, 2004; Erl, 2004; Gold, 2004) are the latest buzzwords in the industry to address the issues identified above. The service model is one that utilizes loosely-coupled, platform and language neutral framework for designing the next generation distributed systems. It is based on technologies standardized by the W3C, the international standard body that oversees various Web related technologies. It also has strong support from major industry players such as Microsoft, IBM, Sun, HP, and Oracle. As such, it is projected to be a strong technology that many IT organizations will investigate and adopt if proven viable. In fact, Gartner Research compared Web services with previous attempts and stated that this time things may be different because "With Web services, all the major vendors are on board with their support." (Mcdougall, 2001)

Recent news and studies on Web services / SOA have also shown the growth and acceptance of the technology. Accenture, a renowned global management consulting company, just announced that it will invest $450 million during the next three years to accelerate growth of its SOA capabilities (InformationWeek, 2006). According to ZapThink, a market research firm, the market for Web services platforms, application development suites, and management tools is projected to expand from a $380 million (US) market in 2001 to over $15.5 billion (US) in 2005 (Seely, 2002). Also, recent survey by market research firm TechMatrix (of 450 IT professionals and consultants) finds that 65% of small and midsize companies and 35% of large companies have adopted Web services to automate business processes between trading partners and for internal application integration. Another important observation is that the use of Web services is now widespread in many industries including retail, financial services, homeland security, transportation, etc.

This trend is further evidenced by a "leaked memo" entitled Internet Software Services from Bill Gates, Chairman of Microsoft, the largest software company in the world. Excerpts from an internal Microsoft correspondence released by The Wall Street Journal and Computerworld (Niccolai, 2005) show that Bill Gates is calling on Microsoft to jump with both feet into the trend toward software applications being delivered as a service over the Internet. Specifically, he describes the need to transition from "The Internet Tidal Wave" that he wrote in another memo ten years ago to the "services wave" that Microsoft needs to start engaging in. He further says, "Today, the opportunity is to utilize the Internet to make software far more powerful by incorporating a services model which will simplify the work that IT departments and developers have to do while providing new capabilities."

With the above stunning growth, it is inevitable that Web services and SOA will be integrated into IS, CS, and other IT related curricula sooner or later, if history is of any indication. It is thus a natural evolution for these curricula to begin exposing the beginning students with these burgeoning technologies early on given the current trend of software development. This will not only fascinate the students with its interesting collection of activities (see Section 3), but also inspire and prepare them for real-world software development scenarios when they graduate. First-year and second-year undergraduates rarely get the exposure to these technologies because of the need to get to the fundamentals first. This paper describes strategies that can be employed without compromising these fundamentals.

It should be noted that efforts to introduce SOA and Web services are beginning to surface in some CS and IS curricula. Just recently, Georgetown University and IBM Corp., together with George Mason University and the College of Charleston, are collaborating on developing academic programs to address the demand for information technology skills, specifically SOA as a way of using a company's existing technology to more closely align with business goals (campus-tech, 2006). Also, literature shows that there are other integration efforts in the form of upper-division/graduate, seminar type class (e.g., (Connolly, 2005; Humphrey, 2004; Lawler, 2005; Subramanian, 2004; Weaver, 2004)). In this paper, however, it is our belief that ideally one should start to lay the foundation of SOA and Web services in introductory programming and system analysis and design courses. Eyeing on the popularity and importance of the ser-

vice paradigm, its economic argument (Booch,2001), the need to keep IS curricula as up-to-date as possible, and the importance of laying the right foundation early on, it is thus the goal of this paper to describe scenarios that are viable for integrating Web services and SOA into introductory IS courses.

This paper is based on a framework we previously built for the CS discipline that integrates Web services technology into its introductory courses, CS1/CS2 (Lim, 2005). Applying the same concepts, it now presents a different view of the framework that targets IS courses. This effort notes the fundamental differences between IS and CS, as given in Figure 1 below, which depicts both CS and IS discipline range coverage in one unified diagram. It represents a merge of the CS and IS coverage diagrams given in (ACM, 2005). From theory/principles to application configuration (horizontal range) and from hardware architecture to organization issues (vertical range), it can clearly be seen that CS and IS are focusing on fundamentally different fields of study. However, there is a small overlap that exists in both the disciplines, as indicated in the intersection area.

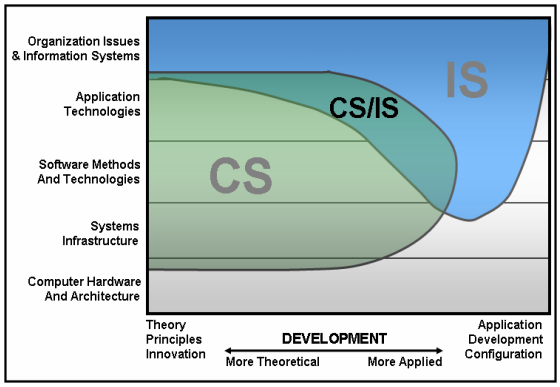

Figure 1: CS/IS discipline range, reproduced from Figure 2.4 (ACM, 2005, page 18) and Figure 2.5 (ACM, 2005, page 19)

The remainder of the paper is organized as follows. Section 2 gives a brief overview of the underlying technologies behind Web services and SOA. The aforementioned scenarios for integration are detailed in Section 3. Section 4 provides a discussion of the implementation of the scenarios, and finally the summary and conclusions are given in Section 5.

2. WEB SERVICES AND SOA: THE SERVICE-ORIENTED PARADIGM

To understand the service paradigm, the term "service" should first be defined. While there are many definitions of what a service is, the following definitions are representative ones. The first emphasizes on lower level computation unit as a service whereas the second one focuses on higher level business process.

> "Services are self-describing, platform-agnostic computational elements that support rapid low-cost composition of distributed applications." (Papazoglou, 2003)

> "[A service] is an application function packaged as a reusable component for use in a business process. It either provides information or facilitates a change to business data from one valid and consistent state to another." (Bennett, 2002)

The above definitions show that a service can be viewed from different levels of granularity. Service suppliers can assemble their services out of existing ones, or develop and evolve atomic services using traditional software development techniques.

Now on to the definition of service-oriented paradigm, which seems to be the next paradigm to follow component-based and OO paradigms, made popular during the 90's. One can define a *service-oriented paradigm* as one that utilizes services as fundamental building blocks for developing applications. It can be thought of as being reincarnated from the concept of ASP (Application Service Provider), another popular software concept of the 90's, where an application is served over a network. ASP is "a third-party entity that manages and distributes software-based services and solutions to customers across a wide area network from a central data center." (Webopedia) This resembles what service computing is trying to achieve, where services are published (by the supplier), discovered (from a directory), and bound and used (by the consumer). For more on this, detailed comparisons of ASP and service-oriented computing are given in (Gold, 2004).

Note that similar concepts of the above are described in *SaaS (Software as a Service)*

(Turner, 2003) and *service-based model of software* (Bennett, 2002). The core concepts are that the users of a service do not own or buy the service. Instead, they use and pay for the use of the service in an on-demand basis. Namely, the separation of the possession and ownership of software from its use is the key to distinguishing this new paradigm from the traditional software paradigm. Another key factor is the dynamic customization of the use of the service at runtime and the disengagement of the service upon completion.

Now on to the definition of SOA, which W3C defines as "a set of components which can be invoked, and whose interface descriptions can be published and discovered". (WS-gloss) The components referred to here are the aforementioned services and an example of the publication and discovery of the service interface can be found in the discussion of Web services, given below. It should be noted that although SOA and Web services are built on similar principles, they are not exactly the same. As explained below, Web services comes with a baggage of technologies (e.g., XML, SOAP, WSDL) but SOA is more than a set of technologies and runs independent of any specific ones. In fact, W3C defines Web services as:

> "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." (WS-gloss)

Under the hood, Web services relies on XML-based SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), and UDDI (Universal Description, Discovery, and Integration) as the underlying technologies for involved parties to communicate and produce/consume a Web service, as shown in Figure 2. Here, a scenario that shows the National Weather Service registering its weather service with a registry and a portal application finding and consuming the service is depicted.

Furthermore, standard-based and firewall-friendly HTTP is commonly used as the underlying transfer protocol. Given that all the technologies involved are non-proprietary, corporations are more willing to invest in

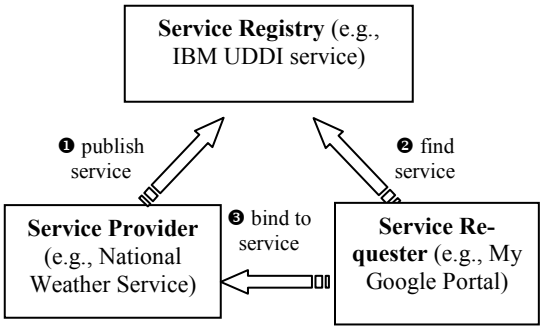

Figure 2: Life Cycle of a Web Service Execution (Registry, Lookup, and Consumption)

this new service model. More importantly, this model allows for interoperable services and the interoperability and scalability of Web services means that developers can rapidly create larger applications and Web services from smaller ones. This adds another dimension to the Web—instead of just person-to-person or person-to-system, it also handles system-to-system. The IT software industry is now banking on this new way of application development as it mimics how hardware vendors have been producing hardware components for years.

3. INTEGRATION SCENARIOUS FOR INTRODUCTORY IS COURSES

Our previous work on integrating Web service into CS1/CS2 (Lim, 2005) lists five scenarios that cover various introductory Computer Science topics. They provide an environment for students to expose themselves to the burgeoning Web services state-of-the-art Internet technology. The five scenarios are: 1) Method Invocation, 2) Sequence, Iterative, and Decision Structure, 3) Sorting and/or Searching, 4) Miscellaneous Data Structure, and 5) Use Different OS. Each of them includes a topic in question, comparison between typical and Web service delivery mechanisms, and an example.

This section extends the aforementioned five scenarios by adding the SOA concept and alternating the delivery mechanism to accommodate the IS discipline, being mindful of the fundamental difference between the

IS and CS disciplines as discussed in Section 1. In addition to the five scenarios, three new scenarios, File vs. DBMS, System Concepts, and EDI/System Integration, are added to address various IS disciplinary areas.

Method Invocation

Web Services / SOA Based Delivery: Instead of merely invoking the methods that are on the same system, introduce the notion that some methods may have been written by others and that these methods (i.e., Web services) are scattered all over the world. The method invocations are delivered to the service providers via the Web.

Example: The Web service method sayHello() may be coded to return "hello" if called from a local system in the United States, but it may return hello in the respective foreign language if the called method is housed in a foreign country (e.g., "你好 (ni hao)" from China, "hola" in Mexico, "konnichiwa" in Japan). This "hello world" of Web services allow the students to be exposed to the new world of Web services and technology with minimal complexity.

Other: The introduction of Web services in this scenario opens up many opportunities to discuss various other subject areas as well. In addition to Web services (SOA in general), one can discuss distributed computation via grid computing systems, e-business, mobile commerce, data communication, and networking in general. Also, because the returned result may contain foreign characters, one can talk about an abstract layer on hiding character encoding schemes. Lastly, the concept to polymorphism can also be tied to the above when one considers how the objects from different countries are reacting in their own ways (of saying hello) to the same message.

Sequence, Iterative, and Decision Structures

Typical Delivery: These topics are typically covered by traditional discussion of scenarios that (1) impose a certain order to solve a problem (e.g., read the rate of pay and hours worked before calculating the weekly salary), (2) require a loop be used (e.g., finding the minimum, maximum, and average among a set of integer objects), and (3) need a decision based on the condition (e.g., granting a withdrawal request provided that there is enough money in the account—a classical if-then-else construct).

Web Services / SOA Based Delivery: Instead of merely processing a collection of objects that carries some meaning but may be boring to the students, one could present a scenario where the goal is to solve a problem by using the three fundamental control structures and some existing Web services that can be composed to form a solution for the problem.

Example: A plausible scenario here is to discuss a problem where one wishes to find out the coldest temperature in all the U.S. by zip codes, within a given radius, at a particular moment in time. Further, the coldest area of the country needs to be plotted on a map (see Figure 3).

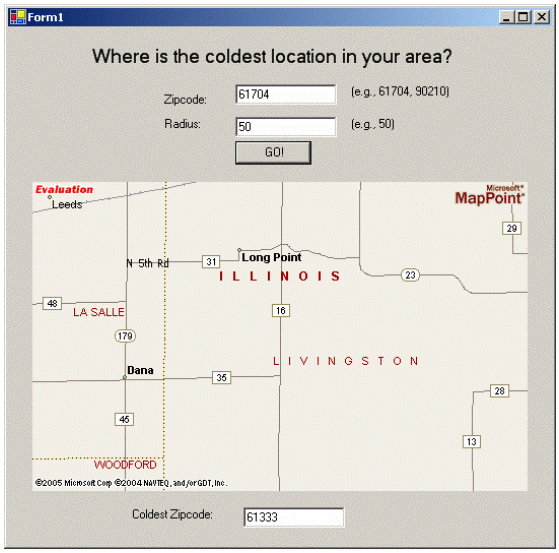

Figure 3: A Web service application to find and plot the coldest location within a specified area (by zipcode)

This scenario, which is much more interesting for today's freshmen, may seem intractable in a traditional introductory programming course environment. But there exist various publicly available Web services that can be composed together to solve this problem rather effortlessly. There exists one that retrieves all the US zip codes (GetZipCodes) (Remotemethods), another one that finds the temperature given a zip code (Xmethods), and yet another one that plots a particular area on a map given a zip code (MapPoint Web service) (Mappoint). Thus,

one can cover the sequence, iterative, and decision structures all in one shot in the above example.

Other: Variations of the above can easily be done if one wishes to get "closer to home" to find temperatures that are in one's neighborhood. There are Web services that retrieve all the zip codes within a given radius mile of another (e.g., GetNearbyZipCodes (Teachatechie)) and ones that calculate the distance between two given zip codes (e.g., (Imacination)). With these Web services, one can for example create exercises that require two different computations of all the areas that are within, say, 50 miles radius of a city like Chicago given the zip codes.

Sorting and/or Searching

Typical Delivery: This topic is typically covered by discussions on various sorting and searching methods and their respective complexities in terms of their speed and sizes. This is typically followed by examples of applying the algorithms on records of interest such as ranking the top 10 NCAA teams and searching for a team given some criteria.

Web Services / SOA Based Delivery: Web services call can be made to return the result of a sort or search, whether this is via publicly available or internally developed Web services. Combining the searching, sorting, and others allow students to learn how to put pieces together to form a service architecture.

Example: Web services that implement different searching and sorting algorithms may be provided for the students to experiment with and gather the performance of the algorithms. A discussion of the different complexities for the implemented algorithms may be reinforced here. Also, applying the same search on different searching engines such as Google, MSN, etc. and comparing their performance may initiate more discussions on the topics. Lastly, students may be asked to observe any network latency as part of the experiments and discuss how the delays compare to the overall performance.

Miscellaneous Data structures

Typical Delivery: This topic is typically covered by discussing scenarios that require the use of compound data structures (e.g., structures such as stacks, queues, and trees) to effectively produce results for certain problems. For example, one may discuss a business model implementation of supermarket queues using simulation based on queuing theory (e.g., to create express lanes or otherwise, how many such lanes). The typical scenarios generally do not involve the use of Web services to present the queue and related data structures.

Web Services / SOA Based Delivery: To make the discussions and potential exercises more interesting, students can be required to consume Web services that directly produce the data structures of interest. Alternatively, they may be asked to create such data structures from the results of Web services themselves to handle the application requirements.

Example: A plausible scenario is to expose the students to the Google Web services API at http://www.google.com/apis/ where instead of going to www.google.com to do searching, a customized version (e.g., an enhanced version of GUI for searching, representing a level of abstraction on top of the search engine) can be built to serve their own interests. Here, we are still relying on Google's powerful search engine to scour the Web, but we can add the bells and whistles (e.g., add spell check, format the input data or the searched results to exercise the methods of compounded data structures) to make the search experience more fruitful.

A discussion of how Google or other search engines typically organizes its data can then follow. For example, Google Directory reuses the data provided by the Open Directory Project and constructs a hierarchical list of Web sites (Ntoulas, 2004). Instead of generating a compound data structure and constructing a traversal algorithm, the service traverses up and down the already constructed directory tree to get the information to the caller.

Other: How Web services are at the core of designing and implementing Web portals can be discussed to elaborate the key concepts of the metadata data structures used by the Web services.

Use of Different OS

(Not So) Typical Delivery: In some curricula, part of the introductory sequence needs to

expose students to an environment that is other than the "main" one. For example, Windows environment that runs Java may be the main environment that supports the main course activities while an alternate one may be any of the Unix/Linux platforms, Mac, or even z/OS mainframes that runs Java. Here, the rationale is to expose the students to multiple OSes and the students may be given a small program to do in the alternative environment.

Web Services / SOA Based Delivery: Instead of developing just another program that runs on an alternative environment, students may be shown how to develop a Web service and be challenged to create a useful Web service in an open system compliance environment, e.g., the Java virtual machine approach. In fact, many of the virtual machine approaches (Smith, 2005) allow the applications to port from one platform to the others without changing a single line of their code. Then, one possibility is to require the students to consume the Web service that they have just built. This allows for a discussion of cross-platform interoperability when the consumer interoperates with the producer. Students would realize that the Web services do not have to be tightly coupled with the hardware and operating system platform.

Example: A Web service on the aforementioned GetZipCodes may be developed and the same service consumer discussed earlier may then be used to consume the newly created Web service. This service may, for example, read from a file that contains all the zip codes and return them to the caller.

File vs. DBMS

Typical Delivery: File and database processing are two topics that are often presented in a disconnected manner, possibly in two different courses.

Web Services / SOA Based Delivery: One viable delivery method is to hide the two data access techniques behind Web services and then reveal the techniques after experiments are carried out.

Example: Via black box approach and easily using two different Web services, students can be interacting with one Web service that implements a sequential file access to, say, a collection of customers, and another one that implements a database access to the same collection on, say, Microsoft Access. With this, an instructor can methodically reveal the underlying access approach and discuss the cost and benefits of using each.

Other: Much like Scenario 3.3, one can discuss the pros and cons of different access methods and the performance metrics here.

System Concepts

Typical Delivery: System concepts and the components and relationships of a system are typically covered in an introductory System Analysis and Design course. Structured and/or OO system analysis and design are the usual paradigms covered.

Web Services / SOA Based Delivery: In addition to the above two paradigms (or in place of some of the portions, to some degree), service paradigm can be introduced as a burgeoning computing approach.

Example: The transition from structured (SA/SD) to OO (OOAD) approach to system development has taken a long while (and it is still happening) and many curricula use the dual paradigm approach to teaching the concepts if they are not completely transitioned from the traditional approach. Textbooks also exist to support such an approach as authors see the market for the balanced-approach, as used in the authors' institution.

A switch to a trio paradigm approach to system analysis and design will certainly be challenging, to say the least. There are no textbooks to support such an endeavor and the research and theory behind service paradigm, with respect to system related issues (e.g., SOA management), is still too immature to warrant a significant coverage of the topic at an introductory level. A recommended delivery is to simply expose the students to the service paradigm at a high level, discuss its benefits (e.g., ones presented in Section 2) and challenges (e.g., when dynamically finding and negotiating for a service, how does the consumer "negotiate with" and "trust" the producer? Others include service composition, orchestration, service transaction management, etc.).

Other: When covering UML (Unified Modeling Language), one can opt to discuss how UML can be used the model the new world of SOA and Web services. This can be done by using its Component Diagrams to model the ser-

vices in question, for example, a legacy COBOL billing subsystem (Smith, 2006). Alternatively, one can also use a UML profile to extend the standard UML elements (e.g., class, association) so that domain or application specific concepts can be modeled more precisely. A profile that specifies the functional aspects (i.e., business logic) in SOA can be found in (Marcos, 2003).

EDI / System Integration

Typical Delivery: Electronic Data Interchange (EDI), which is "the computer-to-computer exchange of structured information, by agreed message standards, from one computer application to another by electronic means and with a minimum of human intervention" (Wikipedia), is a topic that is typically covered as a B2B commerce standard, probably in a 200-level e-Commerce course.

Web Services / SOA Based Delivery: As given in Section 2, Web services are quickly becoming the preferred means of machine-to-machine communication. Together with the various standard XML derivatives (see below), Web services can be introduced as a more cost effective, value-added, more simplistic, and better quality provider of information than EDI, which has been around for more than 2 decades.

Example: Instead of discussing EDI and its ANSI X12 official standard, one can discuss the Electronic Business XML (ebXML) initiative, which represents a technical framework that enables XML to be utilized in a consistent manner for the exchange of all electronic business data. ebXML and Web services hold the promise of realizing the original goals of EDI, making it simpler and easier to exchange electronic documents over the Internet. Note that the ebXML messaging specification is based on SOAP with Attachments. Also, one can discuss various other industry efforts to standardize data interchange, such as FpML (Financial Products Markup Language), which is the XML-based, freely licensed, e-commerce standard supporting OTC trading of financial derivatives.

4. DISCUSSION

First and foremost, it should be noted that the scenarios and activities discussed in Section 3 are merely intended to be ones that can be used to integrate Web services and SOA into introductory information systems courses and expose students to an increasingly important topic without compromising the core of the courses. They are not intended to replace any of the core topics and their use should be considered experimental at this point in time.

Such an experiment has been investigated in the 2nd IS programming course at the authors' institution. The overhead of introducing Web services was minimal for the instructor and would be minimal for many others who wish to try given that in many of the scenarios, "black-box" approach would be used. The underlying details are not expected to be fully disclosed until the students are at an upper division course that hopefully has a segment that reveals the details and components of Web services and SOA.

Also, with today's technologies, consuming an existing Web service or developing a new one is a relatively easy proposition. For example, to consume a Web service in the Microsoft Visual Studio.NET environment, one needs to simply provide the URI (Uniform Resource Identifier) of the Web service and include it as a *Web reference* in the Solution of the project being built, all of which can be accomplished via point-and-click. Then, to invoke a service, one simply instantiates an object from the associated proxy class (automatically generated and included in the Web reference) and makes a method call, like how one would normally make a call to a regular object.

Similarly, to develop a Web service, one needs to simply create a Web Service project in VS.NET and tag the method that is to be exposed using the `[WebMethod]` attribute, i.e., just insert this attribute before the method! When done, the method is now exposed and is callable as a Web service.

Both the consumption and the development of Web services can also be achieved easily using technology from the other camp, i.e., Java. Many Java IDEs (e.g., Eclipse with its Web Tools Platform, Oracle JDeveloper, IBM WebSphere, Sun Java Studio, etc.) provide functionality that is comparable to Microsoft VS.NET and allow for simple integration of Web services technology.

As for the coverage of SOA and the general topic of Web services (this is a true over-

head as this topic is not known to be covered in any courses before the advent of the burgeoning technologies), this can conceivably be introduced when the general introduction to software development is given. An estimate is that about 1-week of coverage (in a 16-week semester) should be dedicated to this introduction with SOA / Web services being a major component here. SOA and Web services are changing the landscape of software construction and their technologies are at the heart of software evolution and thus deserve the spotlight and time coverage.

5. SUMMARY AND CONCLUSION

With computers becoming commodity products and software becoming more and more complex to support their abundance and ubiquity, the software development technologies utilized by the software industries have evolved continuously over the decades to address challenge. SOA and Web services are the latest collection of technologies that are at the center of attention in recent years for addressing this challenge of developing interoperable, cross-platform software components that can be integrated easily. While the industries and IT businesses are working toward utilizing these technologies to enhance their software development practices, the demand of trained professionals in this field is on the horizon.

To produce graduates who are knowledgeable about these technologies, many educational institutions have already started to integrate SOA and Web services technologies into their upper level courses. Presumably, their arguments for integrating the technologies at the upper level are such that the undergraduates need to acquire basic computing foundation first before they can proceed to the advanced technologies. We, on the other hand, believe that the integration of these new technologies should be done as earlier as possible, once proper integration mechanisms are carefully formulated.

With that belief as our guide, we extracted the spirits of SOA and Web Services and synthesized them to form numerous scenarios for the IS majors. Blending in with the fundamentals of IS, we generated examples from these scenario to fit the first few courses of IS curricula based on the criteria depicted in Computing Curricula 2005 (ACM, 2005). This resulted in the presentation of eight scenarios with examples for integration in this paper. They are intended to serve as guidelines for others to consider and more importantly as examples for many other topics that we did not cover in this paper.

6. REFERENCES

ACM, AIS, IEEE-CS (2005). "Computing Curricula 2005 – The Overview Report", September 30, 2005.

Benfield, S. (2001). "Web Services: XML's Killer App." Java Developers' Journal, Vol. 6, No. 4.

Bennett, K.H., et al. (2002). "Prototype implementations of an architectural model for service based flexible software." Hawaii Int'l Conf on System Sciences.

Booch, G. (2001). "Web Services: The Economic Argument." Software Development, Vol. 9, No. 11, November.

Campus Technology (2006). "G-town, IBM Develop Curricula to Address IT Skills Shortage." http://www.campus-technology.com/news_article.asp?id=19242&typeid=150, September, 2006.

Colan, M. (2004). "Service-Oriented Architecture expands the vision of Web services, Part 1." http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html, April.

Connolly (2005). "A Funny Thing Happened on the Way to the Form: Using Game Development and Web Services in an Emerging Technology Course." Information Systems Education Journal, 3 (38). http://isedj.org/3/38/. ISSN: 1545-679X. (Also appears in The Proceedings of ISECON 2004: §2442. ISSN: 1542-7382.)

Curbera, F., et al. (2002). "Unraveling the Web Services Web." IEEE Internet Computing, march/April.

Dyck, T. (2001). "Web Services Wave (the Cover Story: Web Services Wake-Up Call)." *eWeek*, Vol. 18, No. 35, Sept.

Erl, T. (2004). "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services." Prentice Hall.

Gold, N. and A. Mohan, C. Knight, M. Munre (2004). "Understanding Service Oriented Software." IEEE Software, March/April.

Humphrey, M. (2004). "Web Services as the Foundation for Learning Complex Software System Development." 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, Virginia USA March 3–7.

Imacination, http://webservices.imacination.com/distance/

InformationWeek (2006). July 24, pp. 17.

Kiely, D. (2001). "WSDL for Defining Web Services," Cover Story, XML Magazine, Vol. 2, No. 4, August/September.

Lawler, et al (2005). "A Study Of Web Services Strategy In The Financial Services Industry." Information Systems Education Journal, 3 (3). http://isedj.org/3/3/. ISSN: 1545-679X. (Also appears in The Proceedings of ISECON 2004: §3443. ISSN: 1542-7382.)

Lim, B. L. Billy and H. J. Wen (2003). "Web Services: An Analysis of the Technology, its Benefits, and Implementation Difficulties." Information Systems Management, Vol. 20, No.1, Spring, pp. 49-57.

Lim, B. L. Billy and Chu J. Jong, Pruthikrai Mahatanankoon (2005). "On Integration Web Services From the Ground Up Into CS1/CS2." Proceedings of ACM SIGCSE 2005, February, 2005

Mappoint, http://www.microsoft.com/mappoint/default.mspx

E. Marcos, V. de Castro, and B. Vela. (2003) "Representing Web services with UML: A Case Study," Int'l Conference on Service Oriented Computing, December.

Mcdougall, Paul (2001). "Decoding Web Services." InformationWeek, http://www.informationweek.com/story/IWK20010928S0008.

Niccolai, J. (2005). "Gates memo puts online services at heart of Microsoft." http://www.computerworld.com/developmenttopics/development/webservices/story/0,10801,106069,00.html, November 09.

Ntoulas, A. and J. Cho, C. Olston (2004). "What's New on the Web? The Evolution of the Web from a Search Engine Perspective." *WWW2004*, May 17-24, New York, NY, USA.

Papazoglou, M. and D. Georgakopoulos (2003). Guest editor introduction: "Service- Oriented Computing." ACM SIGSOFT Software Engineering Notes 46, 24-28.

Remotemethods, http://www.remotemethods.com/home/valueman/validati/zipcodes

Seely, R. (2002), "Analysts: bull market for Web services," Application Development Trends, http://www.adtmag.com/article.aspx?id=5956&amp

Smith, R. (2006), "Modeling in the Service Oriented Architecture," http://www.devx.com/javaSR/articles/smith1/smith1-1.asp, retrieved Sept., 2006.

Smith, James E. and Ravi Nair (2005). "The Architecture of Virtual Machines," IEEE Computers, May 2005.

Subramanian and White (2004). "Three "Hot" Emerging Technologies: What They Are, and What They Mean for IS Education." Information Systems Education Journal, 2 (7). http://isedj.org/2/7/. ISSN: 1545-679X. (Also appears in The Proceedings of ISECON 2003: §4122. ISSN: 1542-7382.)

Teachatechie, http://teachatechie.com/GJTTWebServices/ZipCode.asmx

Turner, M. and D. Budgen, P. Brereton (2003). "Turning software into a service." IEEE Computer, 36, pp. 38-44.

Weaver, A. and J. Peden (2004). "Integrating Web Services into the Undergraduate Computer Science Curriculum." National Science Foundation, CISE Education Research and Curriculum Development Program, January 12.

Webopedia, http://www.webopedia.com/TERM/A/Application_Service_Provider.html

Wikipedia, "Electronic Data Interchange." http://en.wikipedia.org/wiki/Electronic_Data_Interchange

WS-Gloss, "Web Services Glossary." http://www.w3.org/TR/ws-gloss/

Xmethods, http://www.xmethods.net/sd/2001/TemperatureService.wsd