INTEGRATING WEB SERVICES INTO A WEB-BASED COLLEGE ADMISSION PORTAL SYSTEM

Dr. Billy Lim, Yan Sun
School of Information Technology
Illinois State University
Normal, IL 61790-5150, USA
bllim@ilstu.edu, ysun@ilstu.edu

ABSTRACT

Software reuse and systems interoperability have been primary goals of many IT organizations as means to curb software cost. While there were spots of success over the years, the state of the industry with respect to reuse and interoperability is still lagging. Two main reasons are due to lack of standards and security barriers imposed on many software systems. Web service, a self-describing service that can easily be consumed over the Web, is the latest trend in the industry to address the problems identified above. Based on XML and HTTP, Web services make it possible for involved parties across networks to communicate and produce/consume a service in a uniform manner. This article briefly discusses Web services and describes a project that integrates Web services into a Web-based college admission portal system. The integration illustrates how the aforementioned problems can be addressed using the burgeoning Web services technologies.

Keyword: Web Services, XML Technology, XML Schema, e-Business Systems

INTRODUCTION

For many years, software reuse and systems interoperability have been primary goals of many IT organizations as means to curb software cost. These organizations have software applications that use the Internet to transfer data and conduct business transactions. Object-oriented (OO) technology has been utilized to accomplish these goals with relative success over the years. Nevertheless, there are many hurdles that OO technology could not overcome.

One of them is due to lack of standards. A software component developed in one vendor's technology cannot easily communicate with another vendor's. Another difficulty is due to the fact that the majority of software applications reside behind firewalls – security barriers that restrict communication between networks. Here, even if two systems use the same protocol to communicate, the security of firewall prevents the communication to take place.

Web service (1, 3, 4, 6, 7) is the latest buzzword in the industry to address the problems identified above. It is based on technologies by the W3C, the international standard body that oversees all Web related technologies. It also has strong support from major 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."

In a nutshell, Web services can simply be thought of as self-describing services that are HTTP-addressable. This means that one can shop for a software service much like one can shop for goods on the Web, using exactly the same protocol. Web services relies on 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 1. Here, a scenario that shows a brokerage house registering its stock quote service with a registry and a financial software finding and consuming the service is depicted.

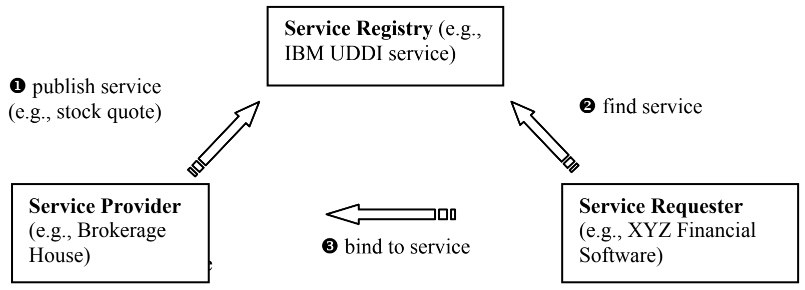

Service Registry (e.g., IBM UDDI service)
❶ publish service (e.g., stock quote)
❷ find service
Service Provider (e.g., Brokerage House)
❸ bind to service
Service Requester (e.g., XYZ Financial Software)

Figure 1: Life Cycle of a Web Service Execution (Registry, Lookup, and Consumption)

Behind the scene, SOAP uses HTTP as the protocol to transmit its message, which is in XML format. This powerful combination of HTTP and XML, both standards of W3C, provides a fully extensible mode of communication between loosely-coupled software systems. 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.

Given that HTTP is a firewall friendly protocol and XML is becoming more and more popular as a standard for data exchange, it is not surprising that Web services has received so much attention in the industry, including a new Web Services Journal for the coverage of the topic (8). It is the buzz that all major IT corporations are talking about and it is also the centerpiece of the .NET campaign of Microsoft (5), a technology that Microsoft is staking its future in, with $4 billion initial investment.

This new way of application development is mimicking how hardware vendors have been producing hardware components for years. Now, the software vendors even have the Web, one of the most important revolutions in the computer industry, on their side. As stated by Bill Gates (5), Chairman of Microsoft, “The power of the XML Web Services model is amazing. A company offering an online electronic-payment service can expose its service to partners, so that they can deliver it as part of their own offering – regardless of what platform they are using. An airline can link its online reservation system to that of a car-rental partner, so travelers can book a car at the same time they book a flight. … XML Web services help your business break free of its boundaries.”

Eyeing on the popularity and importance of Web services and their economic argument (2), this paper describes a project that integrates Web services into a Web-based college admission portal system (CAPS). This proof-of-concept system illustrates how the aforementioned problems can be addressed using the burgeoning Web services technologies. This prototype system also features the use of another important technology, XML Schema (9), for capturing the semantics of a transcript, a key ingredient to college admission.

WEB-BASED COLLEGE ADMISSION PORTAL SYSTEM

With the advent of the Web, more and more universities have started to add online application to their admission systems to make the application process easier and to better manage the student admission information. However, due to technological limitations, it is rare for an admission office to electronically gather all the information needed from an applicant via an online system. Among the documents required for an application, official documents such as transcripts from the applicant's old colleges are the most difficult to submit online for obvious reason – transcripts need to be authentic and it is not easy for the recipient to determine if an electronic copy is authentic, especially if it comes directly from the applicant.

Thus, applicants have to resort to asking for the transcripts the old fashion way and need to make sure sufficient time is given for the postal mail to deliver the documents on time. Also, from the admission office point of view, since the application materials are not delivered at the same time in one package, it is error-prone to collect documents and information from many different places and assemble them into a single application package for a particular applicant.

With the aim to address the above drawbacks of a typical college admission system and to conduct a proof-of-concept of various burgeoning technologies, this project sets out to build a Web-based portal system to facilitate the student application process by means of allowing the transmission of electronic transcripts over the Web between two colleges. Students can also use the portal to request application materials, check applicant status, etc. to name a few.

As stated, the technologies of Web services have the characteristics of supporting communication between "loosely-coupled" systems. The two participating systems do not need to be running on the same platform, using the same OSs, etc. This ensures system interoperability, which is very important in this project since each college may build its own transcript Web service system. One may have different authentication requirement than another, one may be implemented in Java while another may be a .NET application, etc. All that is needed here is that a target system supports a Web service that can be consumed to produce a transcript according to a specified format (see the next section for the format). Figure 2 below depicts the overall architecture.

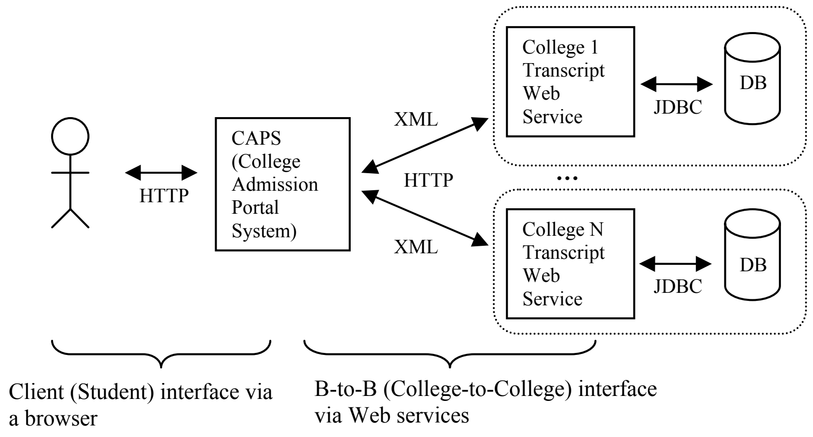
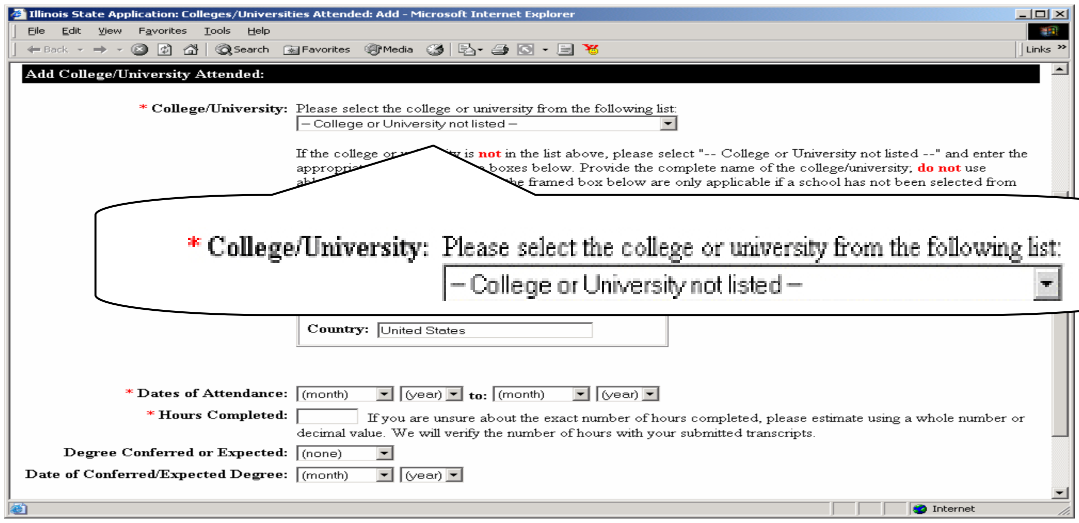

CAPS (College Admission Portal System)
HTTP
XML
HTTP
XML
College 1 Transcript Web Service
JDBC
DB
…
College N Transcript Web Service
JDBC
DB
Client (Student) interface via a browser
B-to-B (College-to-College) interface via Web services

Figure 2: Architecture of the Web-Enabled College Admission Portal System with Web Services

There are two major components in this prototype college admission Web service system. One of them is the client interface. The client interface is where a potential student interacts with the CAPS of the college that he/she is applying to using a Web browser. As far as the student is concerned, this process is merely about interacting with a standard set of Web pages. The fact that CAPS is making requests to other systems via Web services is totally abstracted from the users.

The second part is the core of the system – the service part. This part resides in the colleges that the student used to attend. CAPS sends requests to their Web services, and after the identity of the applicant is authenticated, necessary data (in our case, the transcript contents) are retrieved from the databases and returned to CAPS. The transcripts are then transformed from XML format into HTML and presented to the client. Thus, the complete system functions as follows:

1. When an applicant applies for admission, he/she needs to add colleges/universities to the list of schools he/she has attended. As Figure 3 shows, CAPS maintains a list of schools that it has association with, i.e., the ones that offer transcript Web services. (For the ones that do not, the applicant is asked to provide the school information and to send in official transcripts.)
2. Each time a college/university on the list is added, CAPS sends a request to the Web service system of that college/university and asks for the transcript. The request is made with the necessary authentication information for the Web service in question. For example, a Web service may require the applicant's SSN and birth date before the service can be consumed.
3. When the Web service system receives the authentication information from the client, it accesses the registrar system, searches the student information, and attempts to find the matching record.
4. If the applicant's identity is authenticated, the Web service system then retrieves the transcript of that particular student. The transcript contents are then mapped (presumably from relational table format) into XML format and sent back to the client as the response. If the identity cannot be authenticated, an appropriate XML message is sent back.
5. Upon receipt of the transcript, the client stores the transcript (now in XML format) in Oracle9i database, which supports XML. The transcript information is now ready for retrieval by the admission office staff and/or the applicant. XSLT is used to transform the transcript contents from XML data format into HTML format so that the admission office staff and/or the applicant can view the transcript using a Web browser in a user-friendly format.


Illinois State Application: Colleges/Universities Attended: Add - Microsoft Internet Explorer

Add College/University Attended:

* College/University: Please select the college or university from the following list:
– College or University not listed –

If the college or u… is **not** in the list above, please select "-- College or University not listed --" and enter the appropri… boxes below. Provide the complete name of the college/university; **do not** use ab… the framed box below are only applicable if a school has not been selected from

* College/University: Please select the college or university from the following list:
– College or University not listed –

Country: United States

* Dates of Attendance: (month) (year) to: (month) (year)
* Hours Completed: If you are unsure about the exact number of hours completed, please estimate using a whole number or decimal value. We will verify the number of hours with your submitted transcripts.
Degree Conferred or Expected: (none)
Date of Conferred/Expected Degree: (month) (year)

File Edit View Favorites Tools Help
Back Search Favorites Media Links
Internet

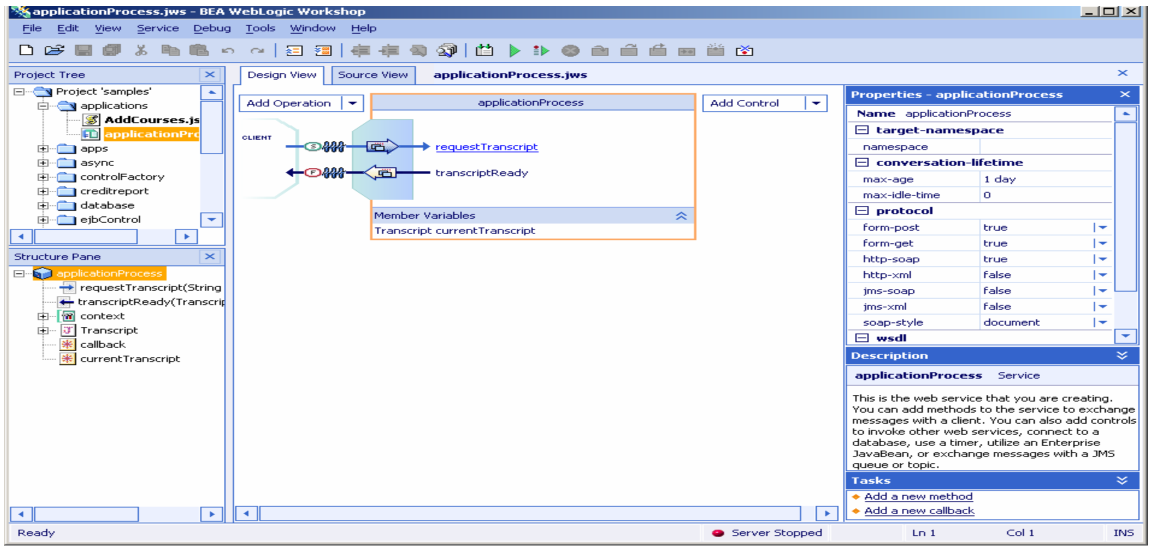
Figure 3: User interface that allows the selection of a college with the transcript Web service support

If the applicant has attended multiple colleges, the above steps will need to be repeated for each college until all colleges have been added. When the request for a college is completed, the applicant is brought back to the screen that displays the colleges added. The applicant can select a particular college and check on the transcript received. (Note that since the request for transcript is asynchronous, an applicant's transcript may not be immediately available for browsing.) Once the transcript request step is completed, the applicant then proceeds with the rest of the application process until the end.

TECHNICAL UNDERPINNINGS

This project uses Java to implement the transcript Web services, specifically the Web service development tool called WebLogic Workshop from BEA systems. This tool, with its visual environment (in its Design View), is the first to provide "drag-and-drop" style development of Web services (see Figure 4 below); one simply adds the business logic using Java in its Source View. (In this project, JDBC is used to access student information stored in an Oracle database.) It supports Web services that represent asynchronous interactions between loosely-coupled systems.

WebLogic Workshop also has a powerful feature called XML maps with which developers can specify the external XML data structure and the tool maps it to the internal Java implementation. For complex or recursive XML structures, it has an embedded JavaScript engine that enables easy mapping into and from XML documents by means of simple JavaScript codes.



Properties - applicationProcess

Name	applicationProcess
target-namespace	
namespace	
conversation-lifetime	
max-age	1 day
max-idle-time	0
protocol	
form-post	true
form-get	true
http-soap	true
http-xml	false
jms-soap	false
jms-xml	false
soap-style	document
wsdl	

Description

applicationProcess Service

This is the web service that you are creating. You can add methods to the service to exchange messages with a client. You can also add controls to invoke other web services, connect to a database, use a timer, utilize an Enterprise JavaBean, or exchange messages with a JMS queue or topic.

Tasks

- Add a new method
- Add a new callback

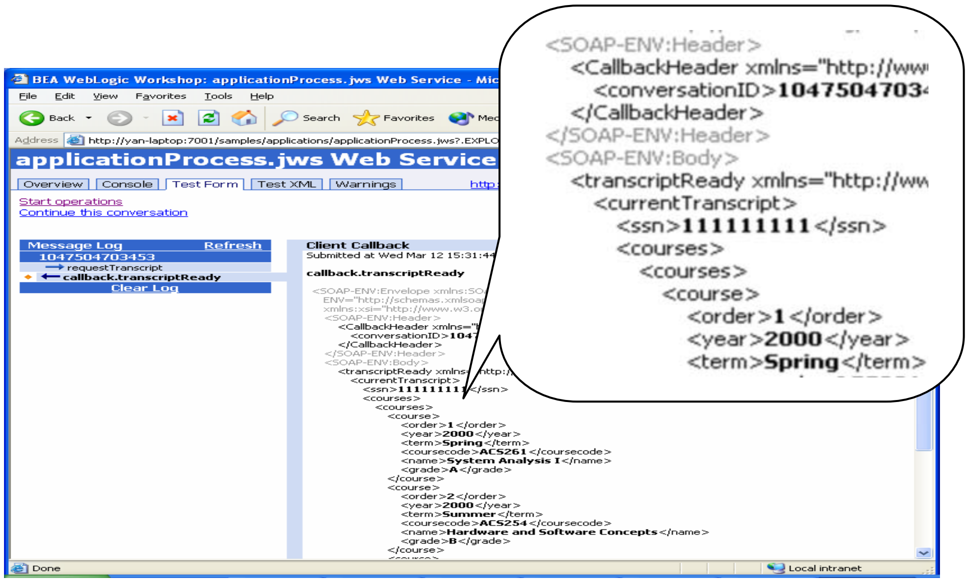
Figure 4: The main visual development screen in WebLogic Logic

Another technology used is XML Schema for structuring and validating transcripts in XML format. This project presumes that a universally accepted format for college transcripts has been defined and that an XML schema for it exists. Much like other B-to-B e-Commerce systems, the business partners must agree on a format before data can be interchanged. For example, UBL (Universal Business Language) by OASIS, a non-profit standard group, is a leading specification language for specifying business-related concepts in XML (e.g., purchase orders) so that businesses can communicate and transact. Here, a sample format for transcript is used for illustration purposes. The format is represented in XML Schema below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="transcript" type="transcriptType"/>
  <xsd:complexType name="transcriptType">
    <xsd:sequence>
      <xsd:element name="ssn" type="ssnType"/>
      <xsd:element name="name" type="nameType"/>
```

```
        <xsd:element name="gender" type="genderType"/>
        <xsd:element name="birthdate" type="xsd:date"/>
        <xsd:element name="degree" type="xsd:string"/>
        <xsd:element name="major" type="xsd:string"/>
        <xsd:element name="courses" type="coursesType"/>
        <xsd:element name="gpa" type="gpaType"/>
        <xsd:element name="hours" type="xsd:integer"/>
        <xsd:element name="university" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="coursesType">
      <xsd:sequence>
        <xsd:element name="course" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType><xsd:sequence>
              <xsd:element name="year" type="xsd:gYear"/>
              <xsd:element name="term" type="xsd:string"/>
              <xsd:element name="coursecode" type="xsd:string"/>
              <xsd:element name="coursename" type="xsd:string"/>
              <xsd:element name="grade" type="xsd:string"/>
            </xsd:sequence></xsd:complexType>
        </xsd:element></xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="nameType">
      <xsd:sequence>
        <xsd:element name="fname" type="xsd:string"/><xsd:element name="mname" type="xsd:string"/>
        <xsd:element name="lname" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="genderType">
      <xsd:restriction base="xsd:string"> <xsd:pattern value="male|female"/></xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="gpaType">
      <xsd:restriction base="xsd:decimal"><xsd:minInclusive value="0"/><xsd:maxInclusive value="4"/></xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="ssnType">
      <xsd:restriction base="xsd:string"><xsd:length value="9"/></xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

Note that the schema incorporates a number of validation checks, including ones that constraint the data types of the elements, one that sets the min and max values for GPAs, and one that enumerates the gender type. A sample document representing a transcript is presented below in Figure 5.


BEA WebLogic Workshop: applicationProcess.jws Web Service - Mic
File Edit View Favorites Tools Help
Back Search Favorites Med
Address http://yan-laptop:7001/samples/applications/applicationProcess.jws?.EXPLO
applicationProcess.jws Web Service
Overview Console Test Form Test XML Warnings http:
Start operations
Continue this conversation
Message Log Refresh
1047504703453
requestTranscript
callback.transcriptReady
Clear Log
Done
Local intranet

Client Callback
Submitted at Wed Mar 12 15:31:44

callback.transcriptReady

```
<SOAP-ENV:Envelope xmlns:SO
  ENV="http://schemas.xmlsoap
  xmlns:xsi="http://www.w3.o
  <SOAP-ENV:Header>
    <CallbackHeader xmlns="h
      <conversationID>1047
    </CallbackHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <transcriptReady xmlns="http:/
      <currentTranscript>
        <ssn>111111111</ssn>
        <courses>
          <courses>
            <course>
              <order>1</order>
              <year>2000</year>
              <term>Spring</term>
              <coursecode>ACS261</coursecode>
              <name>System Analysis I</name>
              <grade>A</grade>
            </course>
            <course>
              <order>2</order>
              <year>2000</year>
              <term>Summer</term>
              <coursecode>ACS254</coursecode>
              <name>Hardware and Software Concepts</name>
              <grade>B</grade>
            </course>
```

```
<SOAP-ENV:Header>
  <CallbackHeader xmlns="http://ww
    <conversationID>10475047034
  </CallbackHeader>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <transcriptReady xmlns="http://ww
    <currentTranscript>
      <ssn>111111111</ssn>
      <courses>
        <courses>
          <course>
            <order>1</order>
            <year>2000</year>
            <term>Spring</term>
```

Figure 5: A sample transcript shown as a returned XML document in WebLogic testing environment.

SUMMARY AND CONCLUSIONS

This paper overviews the technologies behind Web services and hints at the opportunities of adopting the burgeoning technologies in IT organizations. It illustrates how a prototype college admission portal system can make use of Web services to consume student transcripts over the Web. The portal shows how self-contained, internal applications can easily be turned into reusable, self-describing Web services that can interoperate with other external systems. As stated, as long as the participating institutions agree on the format to represent transcripts, they can request and consume transcript Web services from each other. A sample format, represented in XML Schema, is presented in this project for illustration purposes and perhaps as a first step towards a uniform representation of transcripts.

This project can also serve as a launching pad for anyone wishing to further develop a college admission portal and as a pedagogical tool for teaching portal development with Web services. College admission and transcripts are problem domains that students are already familiar with and the size of this project is appropriate for a college-level advanced Web development course.

Note also that while the ease of consumption of Web services is a benefit of the technology, it also represents a vulnerability that needs to be addressed. That is the reason why there are currently standardization efforts (e.g., WS-Security and SAML (Security Assertion Markup Language)) to ascertain that messages being transmitted are secured in that the messages have originated from the appropriate sender and were not modified in transit. This will make sure that sensitive information such as student transcripts is always authenticated before any transaction can take place.

For Web services to be ubiquitous so that casual/ad-hoc use of Web services is possible, the aforementioned security issues together with several other issues must be addressed, including privacy, transaction, and infrastructure capabilities. The good news is that these issues are actively being investigated and Web services are on their way to becoming a crucial IT technology in the 21st century.

REFERENCES

1. Benfield, S. (2001). "Web Services: XML's Killer App," *Java Developers' Journal*, 6(4).
2. Booch, G. (2001). "Web Services: The Economic Argument," *Software Development*, 9(11).
3. Curbera, F., et al. (2002). "Unraveling the Web Services Web," *IEEE Internet Computing*, March/April.
4. Dyck, T. (2001). "Web Services Wave," *eWeek*, Vol. 18, No. 35, September.
5. Gates, W. (2001). *Microsoft .NET Today*, open letter to Developers & IT Professionals, June.
6. McCright, A.(2001). "Writing Your First Web Service: A Tutorial," *Web Services Journal*, June.
7. McDougall, P. (2002). "Decoding Web Services," *InformationWeek*, Oct 1, pp. 28-37
8. *Web Services Journal*, Sys-Con Publications Inc., Montvale, New Jersey.
9. XML Schema, http://www.w3.org/XML/Schema