

# The SOA Laboratory: a Resource for Education and Research<sup>1</sup>

Norman Wilde, John Coffey, Eric Daniels, Sharon Simmons, Anthony Pinto  
 Department of Computer Science, University of West Florida  
 Pensacola, Florida 32514

## Executive Summary

Large organizations are transitioning their IT activities to a Services Oriented Architecture (SOA) style, with business processes supported by software services which interchange messages over the internet. SOA presents significant Software Engineering challenges, in technology selection, service design, application testing, life-cycle maintenance and staff training.

Educators and researchers are hampered by the lack of accessible SOA systems to serve as testbeds. To evaluate concepts and tools, companies and researchers need model systems of realistic scale. Similarly for effective learning, students must be exposed to systems that are comprehensible, but that confront them with realistic challenges.

We describe an initial version of *Open SOALab*, a SOA Laboratory for use by educators and researchers. *Open SOALab* consists of a small Currency Exchange composite application that can be used as it is or extended by adding additional components.

Initial resources for researchers include support tools for collecting global message traces for program comprehension studies. Initial resources for educators include several proposals for student assignments, including one fully detailed assignment for teaching about cloud computing.

## Table of Contents

1	Background .....	2
2	Existing Resources for SOA Education and Research .....	3
3	The Currency Exchange SOA Application .....	4
4	A Sample Student Assignment - Multi-Currency Hotel Reservations .....	5
5	A Sample Research Project - Understanding Message Flows.....	7
6	Additional Resources.....	8
6.1	Support for Global Message Tracing .....	8
6.2	Deployment Using a Public Compute Cloud .....	8
7	Conclusions .....	8
8	Acknowledgement.....	9
9	References .....	9
	Appendix A - Possible Student Projects Utilizing <i>Open SOALab</i> .....	10

---

<sup>1</sup> This report may be cited as SERC-TR-297, Software Engineering Research Center, <http://www.serc.net>, October, 2009

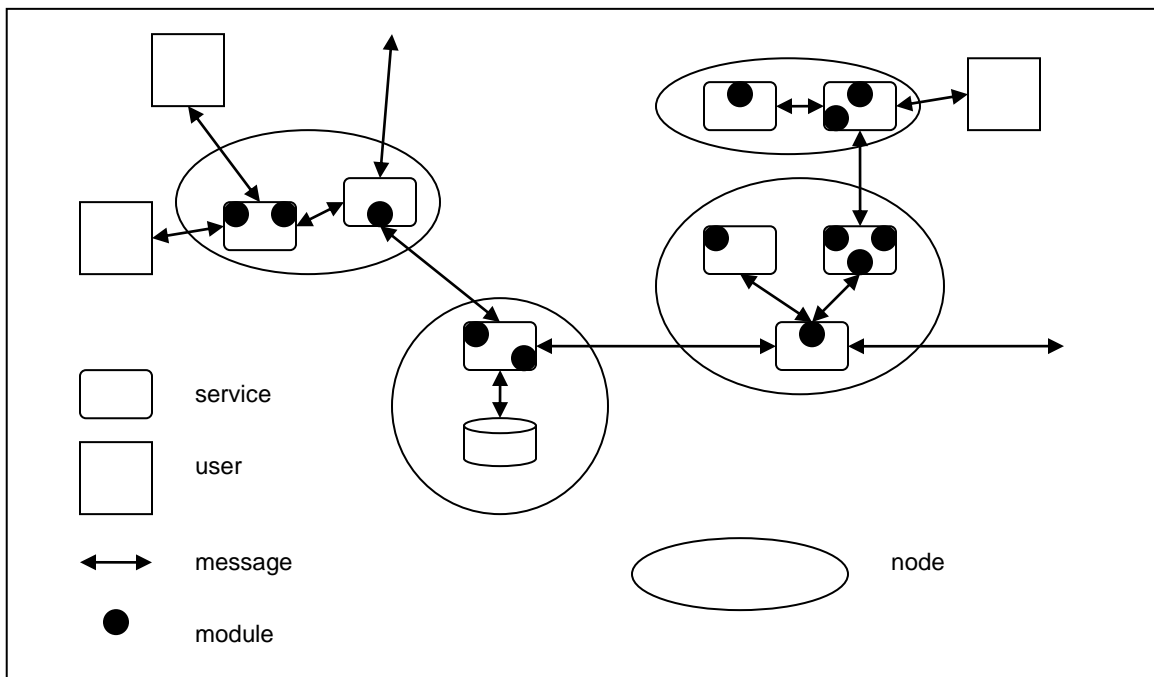
## 1 BACKGROUND

Many companies are turning to a *Services Oriented Architecture (SOA)* approach to better match their information technology systems to their business needs. SOA is a way of structuring *composite applications* (e.g. order fulfillment) by orchestrating a collection of business *services* (e.g. new\_customer) each implemented by software *modules* (e.g. customerEntry.php) and running on several different hardware or logical *nodes* (Figure 1)<sup>2</sup>.

SOA gives companies a way of defining the agile, business-oriented systems that they want in the future and at the same time mapping out a transition path starting from the existing legacy systems they have in place [MARKS2006, Chapter 1]

The emergence of SOA creates new challenges for universities, both in teaching and research:

- **Teaching:** CS, CIS and SE graduates should be prepared to deal with services oriented software to take advantage of emerging job opportunities.
- **Research:** The engineering of SOA systems provides many challenges to meet cost, reliability, and agility goals.



**Figure 1 - Terminology for an SOA Composite Application**

Note that some messages communicate with external services, not owned by the same organization

The specific needs of educators and researchers are a bit different, but they overlap. The concern for educators is to communicate generalizable principles while at the same time providing specific course exercises to support learning. Employers who hire our graduates have emphasized the need for students to internalize the concept of *interoperability*, that is, they must grasp the fact

<sup>2</sup> SOA terminology is not yet standardized. We will use these terms, *composite application*, *service*, *module* and *node*, to describe the structure of an SOA system.

that software they will write after graduation will be part of a much larger system with which it will need to work smoothly. In the SOA terminology of our first paragraph, they must understand that they will build *modules* or *services*, not stand-alone applications. At the same time they must come to see their work in the broad picture of a heterogeneous composite application implementing business processes that provide value for the enterprise.

Thus a minimal need for SOA course exercises is an existing composite application to serve as a context for the exercise. It is usually unrealistic to construct and use such an application within a single semester course - there is simply not enough time to develop such a complex artifact from the ground up. Thus there is a need for an existing, non-trivial, documented composite application available from the first day of the semester.

Researchers may face a similar problem. For some years our main research foci have been program comprehension and software maintenance. This kind of research needs a realistic software artifact so that ideas can be evaluated in experiments and case studies. Software maintenance researchers have traditionally found it very difficult to get access to real, industrial software systems due to legal and technical barriers, not to mention the time pressure under which industrial software engineers work. Much software maintenance research has turned to open source projects where data collection, while difficult, may at least be feasible.

For SOA systems access to real code will be even more difficult since real composite applications are likely to be executing as part of ongoing corporate operations. Experimentation in such contexts is frowned upon so researchers, just like educators, need an existing open source composite application that they can deploy and run at will.

The goal of our *Open SOALab* is to provide a living and growing SOA composite application, with freely available source code and documentation. Educators and researchers can download, adapt, and deploy *Open SOALab* in varied configurations to suit their goals. Software tools will also be provided that students and researchers can use in specific projects as needed.

We intend to continue adding to *Open SOALab* as our SOA teaching and research grows at the University of West Florida. We would also welcome suggestions and contributions from sister institutions. We hope that the laboratory will thus help lower the start-up cost for teaching and research projects and facilitate the spread of SOA technology.

## **2 EXISTING RESOURCES FOR SOA EDUCATION AND RESEARCH**

There are many resources available for teaching SOA in different contexts, but they are rather fragmented. Some of these include sample code that might be used for teaching, though rarely anything as complete as a full composite application. We have not located any code resources that would seem very useful for researchers desiring to study the interactions of a complete system.

If a course goal is limited to teaching students how to create *mashups*, then there are many services available on-line that they can link to. For example StrikeIron, Inc. provides a list of free "lite web services" providing census data, historical exchange rates, etc. [STRIKE2009].

At a higher level there are many books, some that address the overall SOA landscape with a focus on management and governance issues (e.g. [MARKS2006]), while others go more deeply into the technology (e.g. [JOSS2007] or [PAPA2008]). The book format, however, does not really allow the detailed presentation of a specific concrete executable example. The exposition tends to be conceptual rather than immediately practical at that level.

A further resource are the many tutorials, both in book form (e.g. [GEND2006], [DAI2007]) and on-line (e.g. [ECLIPSE2009], [NETBEANS2009]). The tutorials vary widely in scope but do usually provide some code, though very rarely a complete composite application. The tutorial audience is usually practicing software professionals rather than students, and the emphasis is on the use of a specific tool or IDE to perform a specific task.

We have used some of these tutorials in academic classes and have found them valuable, but with a danger that students wind up only learning what buttons to click to follow a particular tutorial thread. They may be lost when asked to generalize their experience to other similar tasks. Additionally, the tools and development environments themselves evolve rapidly, so students are confused when they find the tutorials to be out of step with what they see on the screen. We hope to mitigate that problem in *Open SOALab* by providing examples that are not tied to a particular development environment.

Finally, practical on-line courses are available through programs such as the Sun Academic Initiative [SUN2009b]. Universities can affiliate with this initiative to give their students access to on-line education on many topics, SOA included. Courses are generally small in scope, intended to be completed in a few hours to a few days. However, they do chain together since many courses have prerequisites. Some courses include code fragments to be modified as part of student work; it is not clear what the copyright implications would be of using these fragments in other educational or research activities.

### 3 THE CURRENCY EXCHANGE SOA APPLICATION

The *Open SOALab* software consists of a basic composite application and a series of additions and software tools. The basic *Currency Exchange SOA Application* is a collection of services that simulate banks and currency brokers participating in an on-line market for foreign exchange. Banks offer time-sensitive quotes to convert currencies (e.g. Euros to pounds). Currency brokers model entities such as a currency exchange booth in an airport. They receive currency exchange requests from clients, ask for quotes from one or more banks, and accept the most favorable quote. If a quote is accepted within its time limit the bank performs the transaction by registering the currencies bought and sold with a settlement house clearing service (Figure 2)<sup>3</sup>.

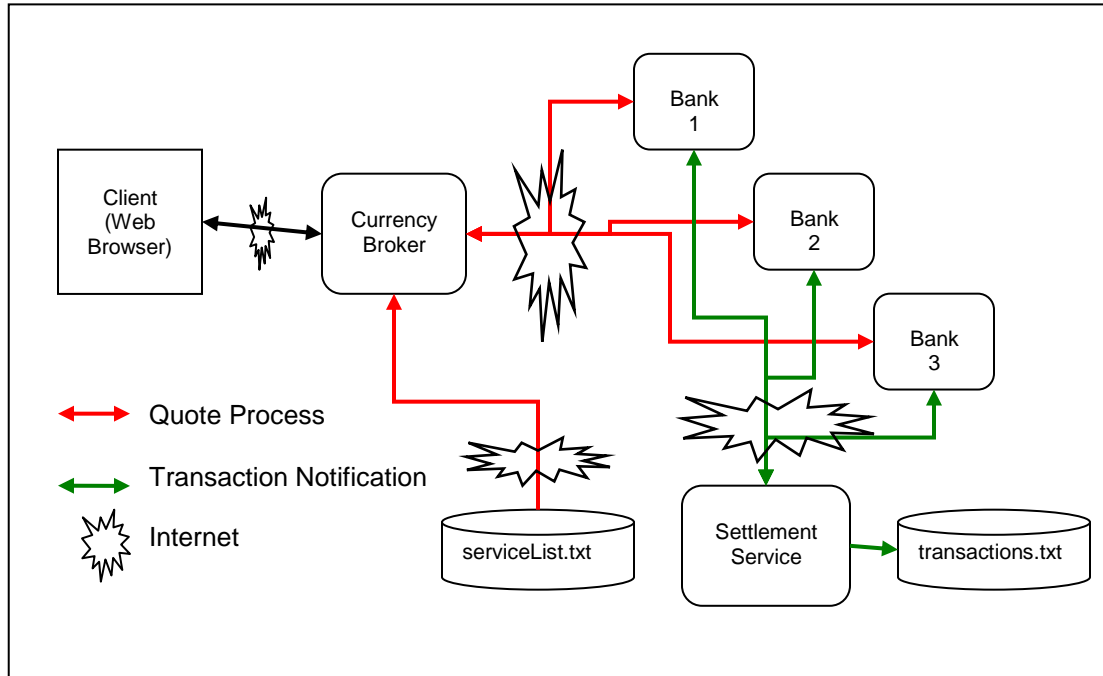
The Currency Exchange was developed as an exercise in SOA design and programming in a graduate Software Design course at the University of West Florida in the Spring, 2007 semester. Each student, as well as the instructor, wrote one currency broker service and one or more bank services. All services were required to interoperate - a currency broker should be able to make use of any of the banks.

The Currency Exchange is kept as simple as possible so that there is little dependence on tools or IDE's, and beginning students can grasp it and modify it quickly. Modules are written as PHP files using a text editor and are deployed to an Apache web server. Though the programming language is the same for all modules, coding style is heterogeneous reflecting the different backgrounds of the coders. Messages are in SOAP [W3C2007] using the NuSOAP library [NICH2004]. Service discovery is modeled by simply publishing a text file with the URI's of each service. Authentication is modeled by a password-based authentication service.

---

<sup>3</sup> Readers may try out one of the on-line currency brokers at <http://cs-zipper.cs.uwf.edu:53011/currencyExchange/dml19/>

The version of the Currency Exchange currently in *Open SOALab* includes three currency brokers, five bank services, and the settlement and authentication services. It can be easily deployed on one to four nodes. A detailed description and instructions for deployment are provided as part of the zip file on the *Open SOALab* web site.



**Figure 2 - Currency Exchange Overview**  
(Authentication messages are omitted for greater clarity)

#### 4 A SAMPLE STUDENT ASSIGNMENT - MULTI-CURRENCY HOTEL RESERVATIONS

Projects using the *Open SOALab* should facilitate teaching students key concepts of SOA such as:

- Business processes and their implementation using a collection of services
- Code reuse by exploiting existing services defined by their interfaces (e.g. WSDL's)
- Use of languages such as BPEL for SOA choreography / orchestration of services
- Timely development by making efficient use of software engineering tools, such as those provided by environments such as Netbeans.

In this section we provide a description of a specific project as it might be presented to students. Other project topics are sketched briefly in Appendix A.

##### ----- Student Assignment - Multi-Currency Hotel Reservations

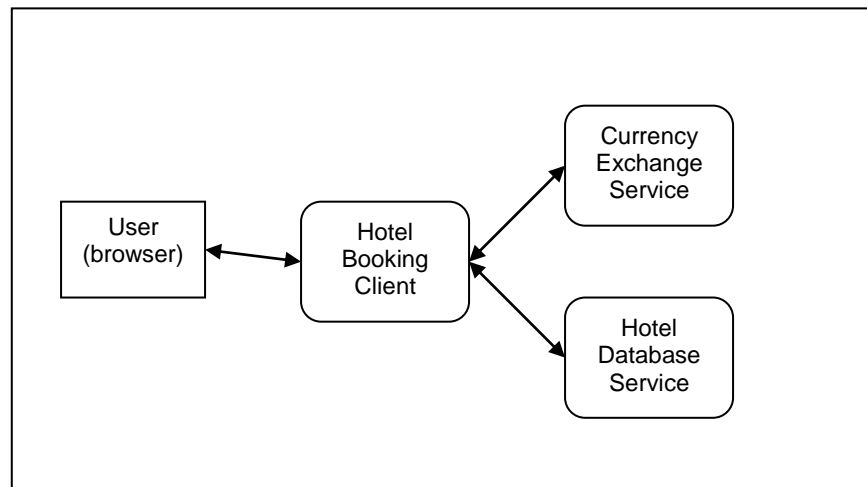
###### Overview

The goal of this project is to simulate a real-world system by integrating a new service with the currency exchange system. It will involve establishing a hotel database service that provides rates and availabilities for hotels, and a new hotel booking client that will be able to invoke both the currency exchange service and the hotel services (Figure 3).

###### Specifications

The hotel service provider will be able to provide two services:

- access a database providing information on hotels in various locations
- book available rooms in these hotels



**Figure 3 - Multi-Currency Hotel Reservations**

Here is a typical scenario for use of the composite application:

- 1) The user accesses the client and selects a city, a date, and his preferred currency (dollars, Euros, yen)
- 2) The client uses the "getQuote" service of one of the banks to get currency rates (note that it does not actually do a transaction at this point - this quote is just allowed to time out).
- 3) The client accesses the hotels database and displays to the user the room alternatives in that city, with their rate converted to the preferred currency. The display contains a note saying prices are approximate until the room is actually booked.
- 4) The user may select one hotel room and click "purchase" (or else he may go back to step (1) or exit).
- 5) The client books the room (updates the database), does a full currency exchange transaction ("getQuote" followed by "purchaseQuote") and displays the final price being charged to the user in his preferred currency.

Clearly, a database will be needed. The database will include a number of hotel chains (think Marriot, Sheraton, Hilton, etc) and the rooms that they have available in various cities and countries throughout the world. Necessary attributes will include room rate and local currency, which will be those currencies that the currency system exchanges. Please adhere to the KISS principle and keep your database as simple as possible to solve the problem as specified, but no simpler. A database design document will be part of the deliverables.

#### **Implementation notes**

This implementation will be in PHP and MySQL. The currency exchange system is implemented in PHP and it serves as a comprehensive example of all the PHP functionality that is likely to be needed. However, the currency exchange system is implemented using the NuSOAP library. The new development will utilize SOAP 1.2 which can be made to work transparently with NuSOAP. Please be aware of this constraint on the problem.

The hotel database service should expose an interface that allows the client to utilize the specified capabilities. The interface is to be WSDL-based, from a permanent WSDL created in a WSDL editor such as the one included in Netbeans (others are available). Note that the examples in the currency exchange generate the WSDL on-the-fly. Implementing a permanent WSDL will afford the opportunity to see how both of these approaches work and will also provide a start on additional work in service discovery via WSDLs in a registry.

The client will have a human-computer interface (HCI), comprised of one or more web pages. It is key to keep ease of use in mind. Elements of the interface in the currency exchange system may be integrated into the solution, keeping in mind HCI design principles such as regular, consistent look and feel.

### **Simplifying assumptions**

Obviously, time will not permit a full implementation of such a complex system, so simplifying assumptions are permitted, so long as they are enumerated in an accompanying design specification. For instance, it might be assumed that the system operates in a snapshot in time – when the instructor tests it. Therefore, the hotel system will have no sense of time passing and of rooms that were booked and became available again. Other, similar assumptions and simplifications will be discussed and permitted on a case-by-case basis.

### **Deliverables**

1. Any design documents including a database schema
2. All necessary program code
3. An installer batch file
4. A User's manual specifying all installation and use details.
5. All SQL commands to create and populate the hotel database

-----

## **5 A SAMPLE RESEARCH PROJECT - UNDERSTANDING MESSAGE FLOWS**

For traditional software systems, the costs of the maintenance phase have almost always been about 40 to 80% of total software costs [GLASS2006]. We see little reason to believe that SOA will escape this rule so there will be a need for tools to support analysis of existing SOA composite applications.

For earlier distributed systems, tracing of message flow has sometimes, but not always, been a productive approach to understanding software features [WILDE2008]. Our ongoing research explores the use of message traces to understand SOA applications. Research issues include integrating trace information from the different nodes, determining the level of trace detail that is needed, and identifying meaningful message patterns in the presence of noise.

SOA middleware offers many opportunities for trace collection. For example, if Apache HTTP Server [APACH2009] is used as the application server, its access log and forensic log can provide a trace of message activity at each message receiving node. Of the two, the forensic log is to be preferred because log entries are created as each message is received; the access log is written only when the response message is sent, often some time after the message is received, so the logical order of events is much less clear.

We are currently experimenting with improved tools to sequence the messages more exactly and thus provide a global view of the behavior of a composite application.

## 6 ADDITIONAL RESOURCES

The Currency Exchange thus provides a starting point for class exercises and research projects. In this section we describe additional resources provided with *Open SOALab* to make it more useful for educators and researchers.

### 6.1 Support for Global Message Tracing

A global view of the message passing in a composite application requires identifying the two endpoints of each message, that is, the sending and receiving services. Unfortunately since each node only knows its own situation, there is no foolproof way to collect endpoint information without some support from the SOA infrastructure software

To overcome this problem one can modify the NuSOAP library to add headers to outgoing SOAP messages. These headers identify the host name and the URI used to invoke the sender service. At the receiving service, the headers appear in Apache's forensic log, so that both endpoints can now be extracted from that log.

*Open SOALab* thus currently provides three resources for collecting global message traces using the forensic log:

1. The version of the NuSOAP library used for SOAP messaging in the Currency Exchange includes a slight modification to add the two additional message headers.
2. A test driver is provided for the Currency Exchange that inserts a mark into the forensic log at the moment that each test starts, thus delimiting the beginning of the set of messages related to a specific test.
3. A Perl script is provided that collects the forensic logs from different nodes and orders them, ready for analysis.

### 6.2 Deployment Using a Public Compute Cloud

An economically attractive way for companies to deploy services is by using *cloud computing* technology, in which nodes are instantiated and retired as needed depending on the current load on the service [Sun2009a]. A company can either own the hardware for a private cloud or purchase computing resources on a public one, paying only for time actually used.

*Open SOALab* includes a student assignment with detailed instructions on how to deploy the Currency Exchange on one or more instances in the Amazon EC2 Elastic Compute Cloud [AMAZ2009]. Two levels of student exercises are provided:

- 1) A simple deployment of a "hello world" HTML file to introduce students to the mechanics of cloud computing.
- 2) A more advanced exercise to deploy and run the Currency Exchange in the cloud.

## 7 CONCLUSIONS

*Open SOALab* is an open source laboratory for SOA educators and researchers, centered around a basic Currency Exchange composite application. A growing collection of documents and tools is available to facilitate its use in student projects and in SOA research.

The current version of *Open SOALab* has three main limitations:

- 1) It is quite small. At about 9600 lines of PHP code (raw line count) the Currency Exchange is several orders of magnitude smaller than a real SOA composite application. From a teaching standpoint the small size may be desirable since it is easier for students to grasp what the application is doing. However for researchers it would be better to have a system much closer to real industrial scale.

- 2) It uses only a fraction of the important SOA technologies. Students are exposed to XML, SOAP messaging and WSDL's, but not to UDDI, BPEL, WS-Security, etc, etc. We consider the current version to be a starting point, and are adding laboratory components and exercises that will widen the student experience.
- 3) It is relatively homogeneous. Almost all the current code is in PHP so there is relatively little exposure to the difficulties of interoperability of services written in different languages or using different versions of standards. Again, we hope to add some more diverse components to address this problem.

The different components of *Open SOALab* are available for download from:  
<http://cs-zipper.cs.uwf.edu/soaResources>

All the code and documents are under version control in a Subversion repository [TIG2009].

We would like to thank all the students who have participated in different course activities that have contributed to the current state of *Open SOALab* and we would very much welcome suggestions from readers as to how it could be made more useful for their teaching and research activities.

## 8 ACKNOWLEDGEMENT

Work described in this report was partially supported by the University of West Florida Foundation under the Nystul Eminent Scholar Endowment.

## 9 REFERENCES

- [AMAZ2009] Amazon Web Services, *Amazon Elastic Compute Cloud Getting Started Guide*,  
<http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/>,  
 URL current July, 2009.
- [APACH2009] Apache Software Foundation, *The Apache HTTP Server Project*,  
<http://httpd.apache.org/>, URL current July, 2009.
- [DAI2007] Dai, Naci, Mandel, Lawrence, Ryman, Arthur, *Eclipse Web Tools Platform: Developing Java Web Applications*, Addison Wesley, ISBN 0-321-39685-5
- [ECLIPSE2009] Eclipse, Eclipse Web Tools Community Resources,  
<http://www.eclipse.org/webtools/community/new/new.php>, URL current July, 2009
- [JEND2006] Jendrock, Eric, Ball, Jennifer, Carson, Debbie, Evans, Ian, Fordin, Scott, Haase, Kim, 2006, *The Java EE 5 Tutorial*, Third Edition, Addison Wesley,, ISBN 0-321-49029-0.
- [JOSS2007] Josuttis, N. 2007 *SOA in Practice: the Art of Distributed System Design*. O'Reilly Media, Inc., ISBN 0-596-52955-4.
- [MARKS2006] Marks, E. A. and Bell, M. 2006 *Service-Oriented Architecture: a Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, Inc., ISBN 0-471-76894.
- [NETBEANS2009] Netbeans, *SOA Application Learning Trail*,  
<http://www.netbeans.org/kb/trails/soa.html>, URL current July, 2009
- [NICH2004] Nichol, Scott, *Introduction to NuSOAP*,  
<http://www.scottnichol.com/nusoapintro.htm>, URL current July, 2009.
- [PAPA2008] Papzoglou, Michael, 2008 *Web Services: Principles and Technology*, Pearson, ISBN 978-0-321-15555-9

- [STRIKE2009] StrikeIron Inc., StrikeIron Lite Services  
<http://www2.strikeiron.com/Support/Developers/liteservices.aspx>, URL current July 2009.
- [SUN2009a] Sun Microsystems, *Introduction to Cloud Computing Architecture - White Paper, 1st Edition*, June 2009.
- [SUN2009b] Sun Microsystems, Sun Academic Initiative  
<http://www.sun.com/solutions/landing/industry/education/sai/index.xml>, URL current July 2009.
- [TIG2009] Tigris.org *Subversion*, <http://subversion.tigris.org>, URL current July 2009.
- [WILDE2008] Wilde, N., Simmons, S., Pressel, M., and Vandeville, J. 2008. Understanding features in SOA: some experiences from distributed systems. In *Proceedings of the 2nd international Workshop on Systems Development in SOA Environments* (Leipzig, Germany, May 11 - 11, 2008). SDSOA '08. ACM, New York, NY, 59-62. DOI= <http://doi.acm.org/10.1145/1370916.1370931>
- [W3C2007] W3C World Wide Web Consortium *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)* 27 April 2007, URL current July 2009

## APPENDIX A - POSSIBLE STUDENT PROJECTS UTILIZING *OPEN SOALAB*

*Open SOALab* affords the possibility of a range of projects associated with courses in Web Services and Service Oriented Architecture. Such projects would vary in complexity from relatively simple to complex. The following are a few examples:

1. Design and implement a new broker and service provider for an existing service (simple). Students design and build a new provider either for the currency exchange system or the hotel booking system. Although this project would entail a significant effort, it is classified as simple because several complete solutions are available as examples of how to proceed.

The deployed version of the currency exchange system contains no database operations for student processing or for use in reconnaissance/feature location. However, it is planned that the hotel service will implement a database. Consequently, two additional alternatives are available within this category of project: implement a service that utilizes a database or one that does not. Additionally, two different SOAP libraries will be utilized in the existing system: NuSOAP in the currency exchange and SOAP 1.2 in the hotel system. Faculty will have flexibility to specify the library to be used.

Additionally students would have to determine where they can reuse functionality in the existing system (for instance, in the authentication subsystem), and where they would have to conform their development to established design decisions (e.g.: the format of the “servicelist.txt” file).

2. Design and implement a new service including broker and service providers in PHP (intermediate). Students build a different service, for example, an airline ticket broker and airlines. This project would likely be implemented in PHP as the other services are. Since the existing services and brokers provide extensive examples of PHP development, careful scrutiny of design features, both good and bad, in the designs of the existing services would be an important component of this work.

The scope of the project includes design, implementation and testing of the new service brokers, providers, and authentication scheme, and would entail more work in design, implementation and

testing. Optionally, implementation might include utilization of WSDLs that already exist rather than those that are generated on the fly as in the currency exchange system. Such an arrangement would make possible later implementation of a UDDI repository to store the WSDLs. In the current version, WSDL locations would be hard-coded or retrieved from a file.

3. Integrate new Web Services into the system by retrofitting PHP or Java adapters to existing stand-alone programs that provide useful functionality (simple to complex, depending on the API of the stand-alone application or script). In this work, adapter programs are developed either in Java or in PHP and used as Web Service front-ends to invoke applications that are not network-enabled. The adapter is also able to return the results of the program or script execution to the service client.

4. Multiple language integration based upon code-level API and WSDL (complex). Projects in this category include invocation of web services in the *Open SOALab* from other languages than PHP, and inclusion of other external services (developed in Java or .NET, for instance) in composite applications along with services in the *Open SOALab*. The goal of these projects is to demonstrate multi-language interoperability. The first, multi-language integration at the code level can currently be demonstrated in Java. Working general-purpose code will be developed and analyzed that builds the SOAP envelope piece-by-piece in the client, helping students understand the SOAP vocabulary directly in relation to the data being sent.

Current capabilities include building the request envelope in a Java client and invoking a remote Java Web Service. It is anticipated that the low-level control over the creation of the SOAP request, low-level control over the structuring of the parameters, and the processing of the response will translate readily into PHP and Javascript. The SOAP response is returned as a byte stream that can be captured by the general-purpose client. This approach employs XML from the XML Schema that defines the interface to the remote function. Alternatively, Java or Javascript classes may be used to create SOAP/Http communications at a more abstract level that does not require manual assembly of the SOAP envelope.