A Research Study for the Development of a SOA Middleware Prototype that used Web Services to Bridge the LMS to LOR Data Movement Interoperability Gap for Education

Robert T. Mason
Regis University
Denver, CO, USA

rmason@regis.edu

Timothy J. Ellis
Nova Southeastern University
Ft. Lauderdale, FL, USA

ellist@nsu.nova.edu

Abstract

This research study involved the development of a Service Oriented Architecture (SOA) middleware prototype that provided a solution for the data movement interoperability gap that exists between learning management systems (LMS) and learning object repositories (LOR). The prototype extracted and transferred Learning Object data from a LMS to a LOR using Web Services and FTP. The design of the prototype was based on the 28 functional requirements that were provided by a panel of Computer Information Systems (CIS) experts that participated in a five-stage, web-based, anonymous Delphi process to establish a valid set of criteria for the middleware application. During the requirements gathering process, the panel approved and ranked 28 functional requirements for the prototype. The prototype was then developed and consisted of five major components, a Web-based user interface that was developed using C# ASP .Net, an Oracle 11g Metadata Repository, a Web Service Coordination Program that leveraged the PHP programming language, a LMS (Moodle) and a LOR (FLORI).

Keywords: Service Oriented Architecture (SOA), Learning Object Repository (LOR), Learning Management System (LMS), Learning Object (LO), Web Services, C# ASP .Net, Oracle 11g, PHP, Moodle, FLORI

Introduction

Broisin, Vidal, Meire, and Duval (2005) identified the interoperability gap that exists between Learning Management Systems (LMS) and Learning Object Repositories (LOR). Briosin et al. (2005, p. 478) aptly observed, "It is clear that some sort of interface between the two components (LMS & LOR) is required to enable a system to benefit from the other one." LMSs are independent computer systems that manage and deliver course content to students via a web interface. In contrast, LOR functionality includes storage, cataloging of metadata, inquiry, and retrieval of Learning Objects (LOs) for design, maintenance and reuse. Therefore, the purpose and functionality of LMS software in comparison to LOR software is very different. LMSs and LORs exist on public and private networks located

in many different countries. These disparate, independent networks are often only connected via the Internet and leverage a variety of different software and hardware technologies. The first phase of this research study gained a consensus of CIS experts regarding the functional requirements for a new middleware application that would resolve the interoperability gap between LMSs and LORs (Mason & Ellis, 2011). The second phase of the study involved the development of a Service Oriented Architecture (SOA) middleware prototype that provided a solution for the data movement interoperability gap that exists between LMSs and LORs.

Prior research by Krishnamurthy (2006), Sartipi and Dehmoobad (2008), and IMS (2009) suggested that SOA is a sound architecture to resolve the data movement interoperability gap that exists between LMSs and LORs. Mason and Ellis (2010) argued that SOA was the best alternative for a middleware architecture because it provided: (a) loose coupling via message only communication that is conducted at the inter-organizational level, (b) the strongest adaptability and modifiability of the six middleware alternatives that they reviewed, and (c) open standards that support interoperability. Therefore, SOA was the premise for the high-level architecture that is described in the next section.

High-Level Architecture

The high-level architecture of the LMS and LOR Middleware Application (LLMA) prototype consisted of five major components: (a) a Web-based user interface, (b) a Web Service Coordination Program (WSCP), (c) an Oracle 11gR2 Metadata Repository, (d) an open source LMS called Moodle (Modular Object-Oriented Dynamic Learning Environment), and (e) an open source LOR called FLORI (Fedora Learning Objects Repository Interface). Figure 1 shows the five LLMA components.

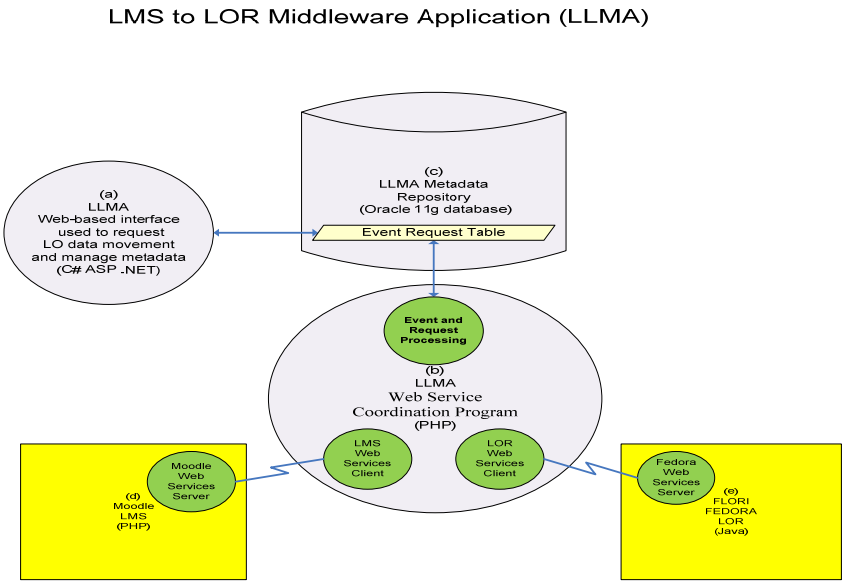


Figure 1. LLMA Five Components

Prototype Development

One of the first steps during the analysis and design process of the prototype was the identification and documentation of 14 high-level use case diagrams. The use case diagrams and functional requirements were then leveraged as input for the modeling of an entity relationship diagram (ERD) to support the Metadata Repository. The data model was created using the third normal form and served as the foundation for the construction of tables, constraints and indexes

that were instantiated within an Oracle 11gR2 database. The use of the third normal form model avoided data redundancy and data integrity errors that can occur when using denormalized tables. The Metadata Repository consisted of 20 tables, 33 indexes, 3 stored functions, 147 constraints, and 7 sequences.

The design process continued with the creation of a UML state processing chart for the Web Service Coordination Program (WSCP). The WSCP was designed as an event processing engine to interface with external applications via Web Services and exchange SOAP messages. The UML state processing chart was an appropriate tool to model the high-level processing mechanism for the WSCP and this approach was supported by the research methodology. Each state that was modeled in the diagram was later coded into a section of the WSCP program using the Hypertext Preprocessor (PHP) programming language (version 5.3.3). Apache Web Server version 2.2 was used for the runtime environment of the application. Therefore, the state chart served as a foundation for the coding of WSCP PHP module (Figure 2).

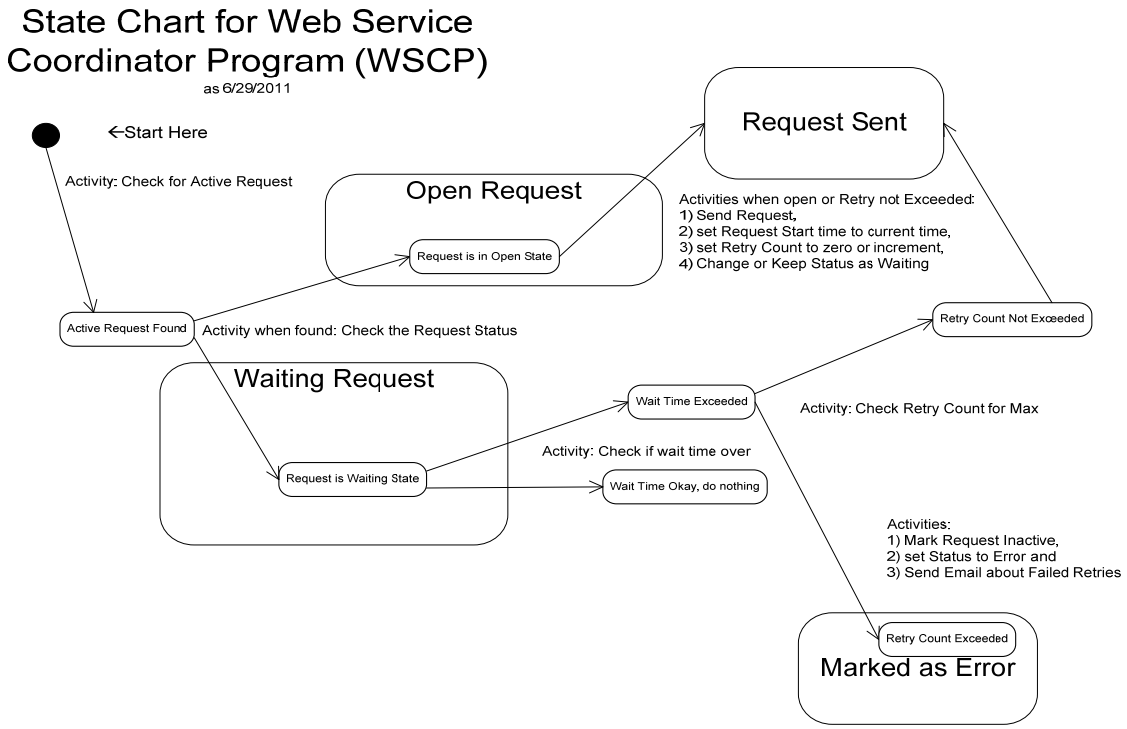


Figure 2. LLMA WSCP State Chart

As shown in Figure 2, the two main states available for active requests were either waiting or open (ready for processing). New event request rows were initially created in the Metadata Repository by the LLMA Web-based interface with an open status; these requests were inserted into an Event Request table (shown in Figure 1c). When the WSCP found an open request, the program immediately processed the request by sending a Web Service request message to the appropriate repository (e.g. LMS or LOR). Event request rows that were found in a wait state were evaluated to see if the WSCP should continue to wait or process the request again. Using metadata, if it was determined by the WSCP that the wait time had elapsed and more retries were permitted, then the request was processed immediately. Otherwise, if the wait time had elapsed and

no more retries were permitted, then the request was marked as an error. Event requests were processed in order of priorities that were assigned when the row was created by the Web-based interface or the WSCP as it processed steps for a series of tasks.

Another important part of the development process was the design of Web pages to facilitate the capture and use of metadata. The Web-based interface was developed using Microsoft Visual Studio 2008 Version 9.0 and the Microsoft .NET Framework Version 3.5. One of the main objectives of the LLMA Web-interface was to allow a user to assemble the necessary metadata needed to send a request to a LMS to copy the LO to a LOR. The metadata that was displayed on many of the Web pages was retrieved from the Metadata Repository. Initially, basic metadata was entered by the user or sourced from external LMSs or LORs using Web Services. For example, to display a list of LORs and LMSs, the user first had to define the name, IP address and user id/password for each LMS and LOR. Once this basic LMS/LOR information was entered by the user, a Web Service could retrieve a list of Learning Objects from the LMS or LOR that were available for processing by that particular user. Figure 3 shows a webpage that allows the user to request a full copy of a LO, a partial copy or refresh of the LO learning assessment data from a LMS to a LOR.

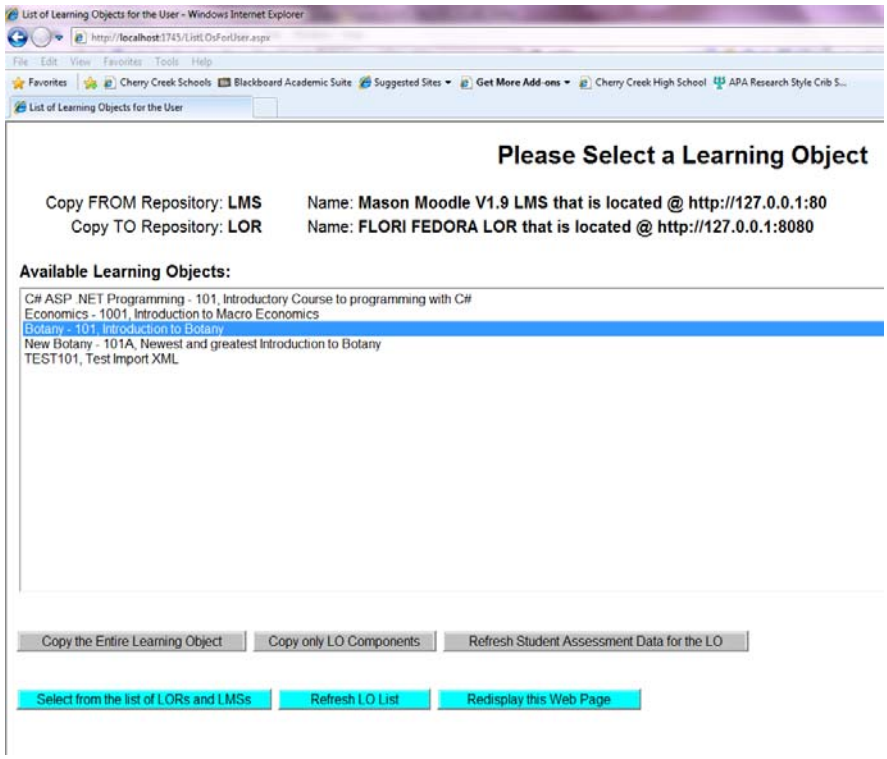


Figure 3. LLMA Webpage to Request LO Data Movement.

LO data was moved from the LMS to the LOR via a two step process. One of the functional requirements stated that the LMS should push the data to the LOR. This push process was accomplished by a Web Service request that called a Moodle procedure that pushed the LO to a staging area. A second event request process completed the pushing the LO to the LOR. Figure 4 shows the design of the first Web Service process that pushed the LO from the LMS to the LLMA staging area.

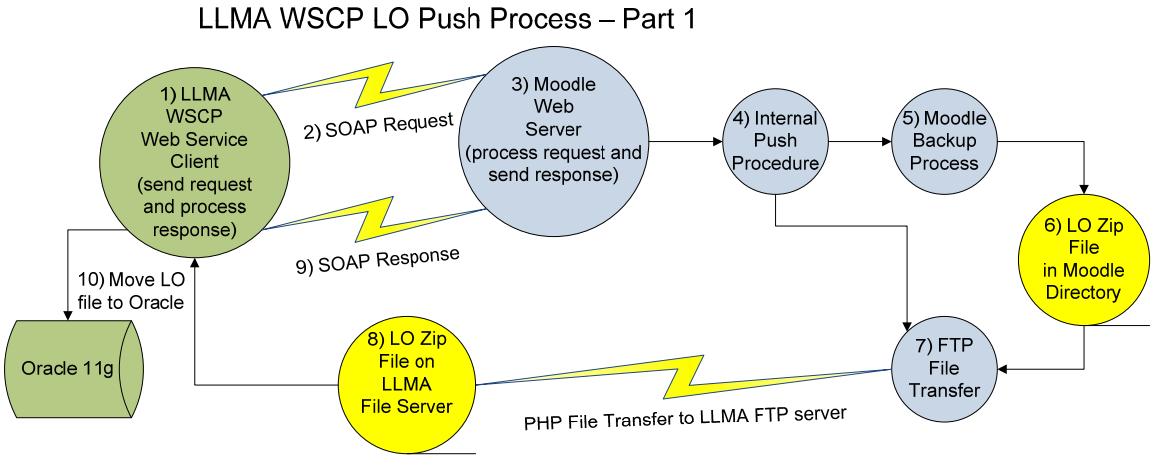


Figure 4. LLMA WSCP 1st Data Push Process

The steps of 1st LLMA WSCP data push process from the LMS to the staging area are listed and described below:

1. The LLMA client Web Service sends a SOAP request to the Moodle Web Server that includes: (a) the LLMA FTP server URL, (b) the name of a Moodle remote procedure to call, and (c) the internal Moodle LO identification number which is stored previously in Metadata Repository when the list of LOs is retrieved from Moodle.
2. The SOAP message is transferred across the Internet to the LMS.
3. The Moodle Web Server receives and processes the request by starting the internal push procedure (e.g. developed by the researcher to extend Moodle).
4. The first step of the push procedure extracts the LO to a zip file.
5. The extract process is a predefined process within Moodle and is normally called the Moodle LO backup process. It is called by the push procedure with the appropriate parameters for one particular LO. The internal LO number is provided by the SOAP message and is sent within the SOAP message.
6. The LO components are copied into a new zip file that contains all the text, graphics, and the other LO file types.
7. The push procedure starts a file transfer process to transfer the file across the Internet to the LLMA FTP file server.
8. The new file is placed on the LLMA FTP Server.
9. The Moodle Web Service procedure responds to the client by returning a SOAP message across the Internet with the file transfer results (e.g. transfer status, file size transferred, name of the new file, and the location of new file in the LLMA staging area).
10. The LLMA Web Service client receives the SOAP response and verifies the results. Then, the LLMA WSCP triggers a load of the zip file into the Oracle 11g Metadata Repository as a Binary Large Object (BLOB). Note that the LO zip file is not actually loaded into an Oracle table, but is copied to a directory that is registered with the Oracle database. Oracle is instructed to create a pointer to the external file. This process is done to reduce the size of storage space needed internally by Oracle 11g. During the transfer process, the name of the LO zip file is changed to match the internal ID of the LO in the LLMA Metadata Repository.

The naming method was adopted to avoid potential collisions with LOs that are loaded from different Moodle repositories and could have the same Moodle internal ID for different LOs. For example, a Moodle repository from Denver and a Moodle repository from Seattle could have two different LOs with the same internal Moodle ID of 5. The LLMA LO ID is unique for every LO referenced in the Metadata Repository.

The 2nd process shown in Figure 5 illustrates how the changes are made to the LO prior to sending the LO to the LOR. The LO Zip file contains both the LOM and the various components of the LO such as weekly lessons, pictures, exam questions, etc.

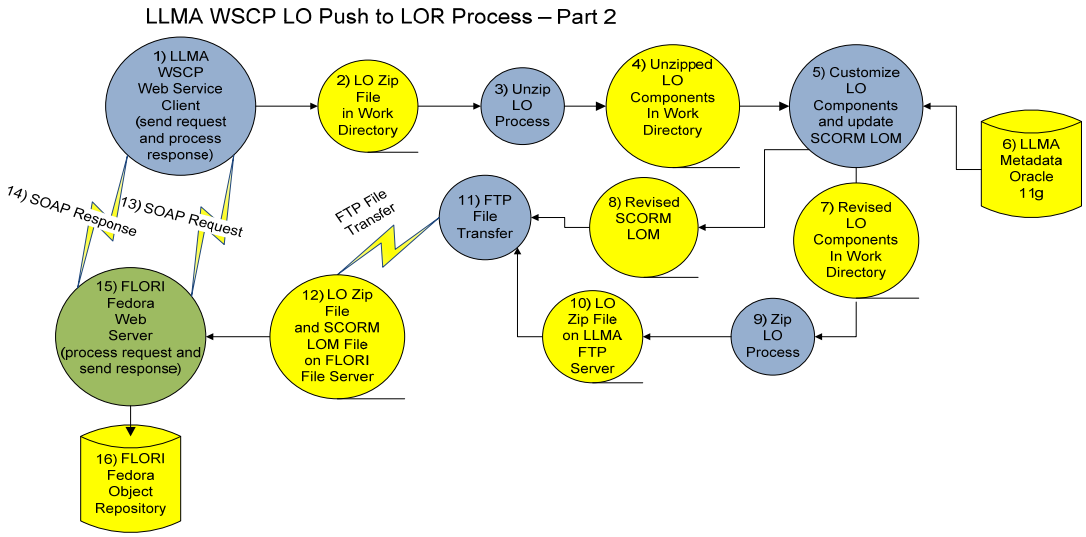


Figure 5. The 2nd LLMA WSCP Data Push Process

For example, an instructor may request a partial copy of the LO and insertion of the LO into the LOR with a new LO name. This type of transformation and manipulation is handled by the second event request process shown above.

The following descriptions correspond to Figure 5:

1. Based on the open event request, the WSCP initiates the LO manipulation process using LLMA metadata to drive the process.
2. The LO Zip file is the source for process 3.
3. This WSCP process unzips the file into a working directory created using the LO name so that the LO components can be altered in case a partial LO copy is requested. In the case of a full LO copy, the file is unzipped to obtain the SCORM metadata file. The SCORM LOM file needs to be inserted into the FLORI Fedora LOR as a separate file that defines the LO; this is a technical requirement of FLORI.
4. A new working directory is created that contains the LO components.
5. This WSCP process revises the LO based on the user request. For example, if the request is for a partial LO copy, then the unwanted sections of the LO are deleted. The SCORM LOM is then revised to reflect the deleted sections from the LO. Also, the LO name is changed in the LOM to reflect the new LO name, if appropriate.
6. The LLMA LOM is used to make revisions to the LO as part of the process in step 5.
7. These are the LO data files after the LO is manipulated by the process in step 5.
8. This the revised SCORM LOM file.
9. This WSCP process zips the revised LO components into the file for transfer to the LOR.

10. This is the LO zip file created from the step 9.
11. This WSCP process transfers both the SCORM metadata file and the new zip file across the Internet to the FLORI file server in preparation for loading into the LOR.
12. The new zip file and the SCORM LOM file are located on the FLORI file server.
13. The LLMA WSCP sends two SOAP requests across the Internet to FLORI. The first request loads the SCORM LOM file into the FLORI repository. The second request loads the revised zip file into the FLORI repository.
14. After each request is processed by FLORI, a response is sent from FLORI to the LLMA WSCP with success or failure messages. As mentioned earlier, failed attempts are retried at a later time until a maximum retry threshold is reached.
15. The FLORI Fedora Web Server processes the SOAP requests to load the two files as described in step 13.
16. FLORI stores the LO zip file and SCORM LOM in a FLORI internal repository.

Discussion

The second phase of this research resulted in the analysis, design and development of an operational prototype that demonstrated the capability to resolve the interoperability gap by moving LO and LOM data from a LMS to a LOR. The development of a data movement middleware application software prototype served as a proof-of-concept that validated the functional requirements agreed upon by the panel of CIS experts. The high-level architecture of the LMS and LOR Middleware Application (LLMA) prototype consisted of five major components: (a) a Web-based user interface, (b) a Web Service Coordination Program (WSCP), (c) an Oracle 11gR2 Metadata Repository, (d) an open source LMS called Moodle (Modular Object-Oriented Dynamic Learning Environment), and (e) an open source LOR called FLORI (Fedora Learning Objects Repository Interface).

Recommendations

Because the LLMA was a prototype, the functionality of the application needs to be extended to create a robust, commercially viable software product. In addition, it is highly recommended that a usability study be conducted with a diverse group of potential users to get their feedback about the ergonomics of the Web-based interface. The results of the usability study would be leveraged to make design changes to the Web-based interface.

The prototype was designed to be extendable, therefore PHP code and Web Services can be added to provide additional functionality, such as the movement the LO data from a LOR to a LMS or new interfaces to external applications (other LMSs and LORs). Therefore, another recommendation is to add more Web Services to extend the prototype to interface with other software applications (LMSs, LORs, LO design tools).

Another recommendation is to test the LLMA with a large volume of LO data to measure performance. The application may have to be enhanced to handle increased LO data movement (traffic). The LLMA Oracle 11gR2 database may need performance tuning improvements to handle an increased work load. These Oracle changes could manifest themselves in the form of additional LLMA indexes, memory reallocation, Oracle parameter changes and/or the tuning of the physical I/O channels (e.g. disk drives). A review of the tool from the information assurance (data security) perspective would be useful.

Additional testing in the area of fault tolerance and error recovery is recommended. Although basic testing of the error handling was conducted during unit, integration, and system testing, a

more intensive error testing of all the code by independent testers should be conducted prior to full scale LLMA implementation.

Final Conclusion

Unanswered Questions

One major unanswered question from this research is if a compensation mechanism should be included in the LLMA. A majority of the experts felt that learning materials (LOs) should be shared free-of-charge with other academic institutions. However, until the LLMA is made available to the academic community, it is unknown how many people will be willing to share their learning materials with other institutions or even within their own institution.

Future Research

Future research in this subject area can include a study to identify the components of student assessment data that should be included in LOM that would benefit the academic community. This research data can be gathered via a more detailed literature review of the types of assessment data that are currently available. A next step could be confirmation of the assessment data types with a panel of experts that are knowledgeable in this subject area. The deliverables from the research could be a ranked list of learning assessment content that would be used to expand the SCORM LOM and application profiles used throughout the world.

Although this research made huge strides in resolving the interoperability gap between LMSs and LORs, future research could include the integration of LO design tools with the LLMA. LO design tools facilitate the customization (editing) of LOs. There are a variety of LO design tools that are currently available (commercial and freeware), however a detailed analysis of the tools is needed in the context of using these tools with the LLMA. A review of the functionality, platform, and APIs would be a few of the tool characteristics that could be examined to determine the tool integration feasibility with the LLMA.

References

Broisin, J., Vidal, P., Meire, M., & Duval, E. (2005). Bridging the gap between learning management systems and learning object repositories: exploiting learning context information. *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/ E-Learning on Telecommunications Workshop*, 478-483.

Dalkey, N. C. (1969). *The Delphi Method: An experimental study of group opinion.* Retrieved from http://www.rand.org/pubs/authors/d/dalkey_norman_crolee.html

Dajani, J. S., Sincoff, M. Z., & Talley, W. K. (1979). Stability and agreement criteria for the termination of Delphi studies. *Technological Forecasting and Social Change, 13*, 83-90. Santa Monica, CA: Rand Corporation.

Delbecq, A. L., Van de Ven, A. H., & Gustafson, D. H. (1975). *Group techniques for program planning: A guide to nominal group and Delphi processes.* Glenview, IL: Scott, Foresman and Company.

English, J. M. & Kernan, G. L. (1976). The prediction of air travel and aircraft technology to the Year 2000 using the Delphi method. *Transport Research, 10*, 1-8.

Franchak, S. J., Desy, J., & Norton, E. L. (1984). Involving business, industry, and labor: Guidelines for planning and evaluation vocational education programs. *Research and Development Series, No. 250*, 1-83. The Ohio State University at Columbus: The National Center for Research in Vocational Education.

Holey, E. A., Feeley, J. L., Dixon, J., & Whittaker, V. J. (2007). An exploration of the use of simple statistics to measure consensus and stability in Delphi studies. *BMC Medical Research Methodology* (2007 ed.), *Vol. 7*, pp. 7-52.

IMS Global Learning Consortium, Inc. (2009). *Adoption of Service Oriented Architecture (SOA) for enterprise systems in education: Recommended practices.* Retrieved August 10, 2009, from http://www.imsglobal.org/community/forum/index.cfm?forumid=10

Krishnamurthy, L. (2006). *Comparative assessment of network-centric software architectures.* Masters of Science Thesis, 1-98. Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Mason, R., & Ellis, T. (2010). A recommendation for the use of Service Oriented Architecture (SOA) to bridge the LMS to LOR data movement interoperability gap for education. *Proceedings of the Informing Science & IT Education Conference (InSITE) 2010*, 43-56.

Mason, R., & Ellis, T. (2011). A research study to gather the functional requirements for a SOA middleware application to bridge the LMS to LOR data movement interoperability gap for education. *Proceedings of Informing Science & IT Education Conference (InSITE) 2011*, 339-351.

Sartipi, K., & Dehmoobad, A. (2008). Cross-domain information and service interoperability. *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, 25-32.

Scheibe, M., Skutsch, M., & Schofer, J. (1975). Experiments in Delphi methodology. In H. A. Linstone & M. Turoff (Eds.), *The Delphi Method: Techniques and applications* (pp. 262-287).

Simon, Carbone, A., de Raadt, M., Lister, R., Hamilton, M., & Sheard, J. (2008). Classifying computing education papers: Process and results. *Fourth International Computing Education Research Workshop (ICER 2008)*, 161-172.

Skulmoski, G. J., Hartmann, F. T., & Krahn, J. (2007). The Delphi Method for graduate research. *Journal of Information Technology Education*, 6, 1-21. Retrieved from http://www.jite.org/documents/Vol6/JITEv6p001-021Skulmoski212.pdf

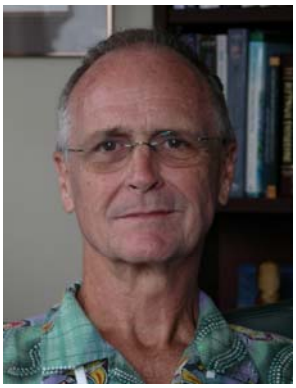
Surowiecki, J. (2004). The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies and nations. In *Books24x7*. Retrieved November 25, 2009, from Books24x7: http://common.books24x7.com/book/id_10328/book.asp

Yang, Y. N. (2008). *Methodology for testing the stability of experts' opinions between successive rounds of Delphi studies* [Data file]. Available from: http://www.tw.org/newwaves/63/1-9.html

Biographies

Bob Mason completed his Ph.D. dissertation in Computer Information Systems (CIS) at Nova Southeastern University. As part of his research, he built a SOA Middleware prototype that was based on the 28 functional requirements that were provided by the panel of seven CIS experts. This prototype helps to resolve the interoperability gap challenges that exist between Learning Object Repositories & Learning Management Systems. He has a MBA with an emphasis in CIS from the University of North Texas. Bob has an undergraduate degree in Forestry Management from the University of Tennessee.

Bob joined Regis University as an Assistant Professor in January, 2011 and is the program coordinator for the SCIS Database Technologies program. Prior to joining Regis as a full-time faculty member, he was an affiliate faculty member at Regis University for 10 years. In addition to teaching, Bob was employed by various Fortune 500 companies for 25 years as a DBA and software engineer. His email address is rmason@regis.edu.



Dr. Timothy Ellis obtained a B.S. degree in History from Bradley University, an M.A. in Rehabilitation Counseling from Southern Illinois University, a C.A.G.S. in Rehabilitation Administration from Northeastern University, and a Ph.D. in Computing Technology in Education from Nova Southeastern University. He joined NSU as Assistant Professor in 1999 and currently teaches computer technology courses at both the Masters and Ph.D. level in the School of Computer and Information Sciences.

Prior to joining NSU, he was on the faculty at Fisher College in the Computer Technology department and, prior to that, was a Systems Engineer for Tandy Business Products. His research interests include: multimedia, distance education, and adult learning. He has published in several technical and educational journals including Catalyst, Journal of Instructional Delivery Systems, and Journal of Instructional Multimedia and Hypermedia. His email address is ellist@nova.edu. His main website is located at http://www.scis.nova.edu/~ellist/