A Recommendation for the Use of Service Oriented Architecture (SOA) to Bridge the LMS to LOR Data Movement Interoperability Gap for Education

Robert T. Mason and Timothy J. Ellis
Nova Southeastern University, Ft. Lauderdale, FL, USA

robemaso@nsu.nova.edu ellist@nsu.nova.edu

Abstract

The purpose of this discussion paper is to explain why we are convinced that Service Oriented Architecture (SOA) is the best alternative to resolve the data movement interoperability gap that exists between LMSs and LORs. An example of a successful implementation of SOA in the Healthcare and Insurance industries is discussed, followed by a recent endorsement of SOA by the Instructional Management Systems (IMS) Global Learning Consortium. An example of the use of SOA for Internet data movement between LMSs and LORs is presented. Network Centric System (NCS) architectures are defined, followed by a comparison of SOA with the relevant alternative architectures.

Keywords: Service Oriented Architecture (SOA), Learning Object Repository (LOR), Learning Management System (LMS), Learning Object (LO)

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." LOR functionality includes storage, cataloging of metadata, inquiry, and retrieval of Learning Objects (LOs) for maintenance and reuse. In contrast, LMSs are independent computer systems that manage and deliver course content to students via a web interface. 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 (Broisin et al.). These diverse, independent networks are often only connected via the Internet and leverage a variety of different software and hardware technologies. Therefore, the software that is needed to bridge this interoperability gap between LMSs and LORs will have to provide a flexible, robust and reliable Internet communication mechanism.

IMS (2006) published the IMS Tools Interoperability (TI) Guidelines to address the requirement of integration of external tools with LMSs to enhance the learning experience for students. External software tools can be used side-by-side with a LMS to enrich the learning experience via extending the functionality of a LMS. For example, an external tool could be invoked to provide additional math tutoring while a student is taking a math course via a LMS. IMS stated that the scope of the initial work was limited and that it hoped that future work would cover more complicated connections between LMSs and external software tools to provide web-based student grade management tools, calendars, and data movement capabilities (e.g. to resolve the LMS to LOR interoperability gap). The IMS TI Guidelines did not compare alternatives for Internet-based data movement architecture.

Although a number of alternatives are available, the literature suggests that SOA is a better architecture for the movement of data across the Internet from a LMS to a LOR. For example, SOA was successfully used in the Healthcare and Insurance industries to enable data movement by mitigating interoperability issues between disparate applications (Sartipi and Dehmoobad, 2008). IMS recently endorsed SOA as a solid approach to bridge the interoperability gap between LMSs and external tools (IMS, 2009). Broisin and Vidal (2005) created a framework called the Computing Environment for Human Learning (CEHL) that was based on the concept of SOA Web Services. The CEHL framework provided Internet data movement capabilities between LMSs and LORs. However, Broisin and Vidal did not specify why they chose SOA as the foundation architecture for the CEHL framework, nor did they publish a comparison of SOA with other data movement architectures. The two main objectives of this paper are to: (a) provide a comparison of alternative Internet-based data movement architectures and technologies with SOA, and (b) document the reasons why SOA is the best architecture to resolve the LMS to LOR interoperability gap.

SOA Endorsements and CEHL

SOA Provides Interoperability for the Healthcare and Insurance Industries

Sartipi and Dehmoobad (2008) investigated interoperability issues for NCSs in the Healthcare and Insurance industries and then developed an interoperability framework that lead to the successful cross-domain interoperability. The framework was based on three types of interoperability:

1. Service Interoperability – The process of defining services that could gather data using legacy system processes (or new system processes) to build standard-based interoperable systems.
2. Information Interoperability – The identification and use of different types of information models to build cross-domain models that spanned interoperability gaps in Insurance and Healthcare data.
3. Semantic Interoperability – A terminology system that allowed for the establishment of relationships between equivalent concepts. This was basically a system that enabled the mapping of terminology that had the same meaning across different domains. Often this type of data and relationship mapping is documented via ontology.

The Healthcare industry has made huge strides towards the identification, evaluation, and construction of reusable software components. The characteristics of discovery and reusability were facilitated by common standards that were developed by industry groups such as HL7 (Health Level 7) and Canada Health Infoway. Sartipi and Dehmoobad (2008) stated that the development of industry standards and the use of SOA diminished the problem of interoperability to a large extent in the Healthcare and Insurance industries. The researchers also suggested that the same standardization approach using SOA could be successfully applied to other areas, such as defense, banking and education.

IMS SOA Endorsement for Education

IMS recently published a white paper that endorses SOA to mitigate the issue of interoperability between LMSs and external tools based on the following functional requirements (IMS, 2009):

- Integrate new systems with disparate legacy systems.
- Increase IT efficiency, reduce costs and off-load non-core functionality.
- Improve employee productivity by providing information in a timely manner.
- Reduce costs by leveraging existing assets by making them accessible for reuse.
- Share information across institutions and organizations (e.g. student assessment data).
- Reduce the time it takes for an IT organization to respond to information requests.

IMS provided best practices for the adoption of SOA for the development of enterprise systems for educational institutions. According to IMS, SOA provides an approach to integrate disparate internal applications within a particular organization and also facilitates the sharing of data across organizational boundaries.

CEHL SOA Framework

Broisin and Vidal (2005) developed the CEHL framework that leveraged SOA Web Services and enabled a LMS user to:

- Search many different LORs for LOs with specific characteristics.
- Import LOs from LORs into a LMS.
- Index/insert a new LO into a LOR from a LMS.

The open SOA architecture of CEHL consisted of four distinct levels:

- A LMS level that interfaced with LMSs and multiple APIs written in the PHP programming language.
- An Application Program Interface (API) level that called one or more services. Each API filtered and consolidated the data from the SOA level.
- A SOA level that contained services that performed specific data gathering or storage functions.
- A LOR level that interfaced with different LORs and the SOA service requests by returning data to the services via standard SOAP packages.

Each API was customized for a distinct LMS, thus Broisin and Vidal (2005) created one API for the Moodle LMS and another API for the INES LMS. Moodle is a type of LMS that is open source that has 45,852 registered sites and 24.9 million users with over 2.3 million courses offered through the web based interface. Moodle is a very popular LMS because it is free and open source, therefore Moodle can be easily enhanced with new functionality (Moodle, 2008). The INES LMS is also freeware that was developed by the University of Amiens, France, and it provides basic course management features (ARIADNE, 2008). The CEHL open framework is interesting because it was:

- Based on a SOA and was composed of four architectural tiers (levels).
- Allowed APIs to be customized for each type of LMS.
- Designed to support the customization of the processes for specific LORs.

The Internet and Network Centric Systems

Chigani and Arthur (2007) describe how the software development paradigm is evolving from a platform centric approach, which is based on tightly coupled intranet components, into a network centric approach that involves the integration of loosely coupled systems that use the Internet to span across organizational boundaries. Kaye (2003) defines loose coupling as a method of designing distributed applications that provide the ability to easily adapt to subsystem (component) changes. Kaye's book reviewed alternative architectures that are used for the design and development of Network Centric Systems (NCS). The development of a NCS usually involves the integration of new and existing software systems to resolve large and complicated problems that cannot be addressed by any particular component on an individual basis (Chigani & Arthur). Within the research literature, a NCS is commonly referred to as a System of Systems (SoS), a Family of Systems (FoS), or an enterprise-wide system.

A broader definition for a NCS is the "interconnection of software, hardware and humans that operate together over a network to accomplish a set of goals" (Balci & Ormsby, 2008, p. 272). Theoretically, a NCS should be able to connect everything with everything else, with a goal of providing services to anyone, anywhere and anytime using desktop computers, smart phones, computer kiosks, laptop computers, large mainframe computers, etc. However, NCSs require an underlying architecture that provides security, flexibility, interoperability, robust connectivity, and other characteristics. The Balci and Ormsby research emphasized the importance of having a sustainable NCS architectural development methodology. In addition, they provided an approach to evaluate NCS architectural development methodologies called the Military System Architecture Assessment Methodology (MSAAM).

NCSs can leverage a variety of middleware products to provide services and therefore can be composed of heterogeneous components (Balci & Ormsby, 2008). Middleware is defined as a software layer between applications and network operating systems that is intended to easily resolve the issues of heterogeneity and distribution for software engineers (Emmerich, Aoyama, and Sventek, 2007). Emmerich et al. discussed the origins of middleware and highlighted the critical role that computer science research played in development of middleware. They noted that the worldwide demand for middleware has increased rapidly to create a sizeable new market with annual license revenues of 8.5 billion dollars. According to Chigani & Arthur (2007), NCSs offer the characteristics that make them very suitable for system to system integration via the Internet. LMSs and LORs are heterogeneous systems that operate independently, span organizational boundaries, exist on disparate networks, and must be integrated via the Internet. In our opinion, the optimum architectural solution to resolve the interoperability gap between LMSs and LORs will have NCS architectural characteristics and should adhere closely to NCS requirements.

NCS Architectural Characteristics and System Requirements

Chigani and Arthur (2007) identified common architectural characteristics of NCSs and then proposed a style for modeling NCSs. The GEON (GEOscience Network) system was designed to integrate multidisciplinary datasets to simulate the complex dynamics of earth systems such as the interaction of oceans and the atmosphere. GEON was based on Service Oriented Architecture (SOA) and provided a functioning example of a NCS that allowed them to apply their evolving modeling technique to a real-life project. Note that SOA is discussed in more detail within the next section of this paper. GEON integrated processes and data from over 22 institutions around the world.

The four distinguishing architectural characteristics of a NCS identified by Chigani and Arthur (2007) were:

- A networked based system of systems.
- Dynamic runtime behavior that may not be known until runtime as a result of the collaborative behavior of the system components.
- An underlying network configuration that restricts component interaction to message (information) exchange.
- A fluid, decentralized dynamic control mechanism that allows various components of a system to control processing based upon the processing objectives to gather specific information.

Similar to the Chigani and Arthur (2007) research, Krishnamurthy (2006) identified four architectural characteristics of a NCS as:

- A system of systems that integrates heterogeneous software and hardware platforms.
- Components that are connected via a network, such as the Internet, Local Area Networks (LAN), Wide Area Networks (WAN), etc.
- A system that can cross organizational boundaries.
- A decentralized dynamic control mechanism that offers runtime dynamism.

Krishnamurthy (2006) evaluated architectural alternatives for NCSs and elaborated on the four basic architectural characteristics by defining six NCS quality and capability requirements. These multipart requirements are used throughout this paper to compare NCS architectural options. Krishnamurthy's six requirements for a NCS are:

- Openness to (a) allow the use of the system components by different organizations, (b) facilitate interoperability, and (c) support open standards.
- Interoperability of data elements, communications, and new components.
- Integration with disparate systems that may be new or legacy applications located on heterogeneous networks.
- Adaptability, modifiability, configurability, and runtime dynamism.
- Dependability in the form of high availability, fault tolerance, resilience and security.
- Scalability and acceptable performance.

NCS Alternatives

Krishnamurthy (2006) provided a detailed comparison of four NCS architectures: Component-based Architecture (CBA), Client-Server Architecture (CSA), Peer-to-Peer Architecture (P2P), and SOA. This section provides a brief overview of the four alternatives identified by Krishnamurthy, plus extraction, transformation, and loading *(ETL)* middleware, a fifth approach to bridge the LMS to LOR Interoperability gap not considered by Krishnamurthy. The following section discusses strengths and weaknesses of the alternatives.

CBA and Two Associated Middleware Products

Butler, Mayo, Weiler (2003) stated that CBA represented a major IT paradigm shift from a traditional software development methodology.   Three objectives of the research were to:

- Document the rise of CBA.
- Examine major CBA implementation issues.
- Provide guidance to government business and technical organizations that would be transitioning to CBA Middlware products.

CBA evolved from an object oriented software engineering approach and therefore encouraged the reuse of existing software components (applications) (Butler et al., 2003). The ability to reuse existing applications resulted in reduced time to market and development costs. Therefore, CBA became an especially attractive option for many IT organizations during the 1990s. Large "Common Off The Shelf" (COTS) application integration projects have always been a challenge for IT organizations (e.g. the implementation of Oracle Financial Applications). Often, system integration teams are tasked with the seamless integration of the new COTS data with existing business application data. CBA products provided value to the IT organizations by making the integration process faster and easier. Two CBA Middleware products that are commonly discussed in the research literature are CORBA and DCOM.

CORBA: The CORBA technical specification was developed by a consortium of more than 700 companies called the Object Management Group (OMG) (Huang & Gannon, 2006). CORBA is a framework based on CBA that provides an Interface Definition Language (IDL) to serve as a language neutral interface between disparate computing environments. Therefore, CORBA was designed to be independent of programming languages, operating systems, and vendor software. OMG accomplished the design goal of a neutral interface by defining common interfaces for different application programs to communicate via Object Request Brokers (ORBs). Since CORBA is a mature technology that was developed in the 1990s, multiple vendors offer ORBs.

CORBA uses custom application-level Internet communication protocols that communicate with the commonly known TCP/IP protocols to facilitate Internet communication (Huang & Gannon, 2006). CORBA uses a binary format known as Common Data Representation (CDR) to prepare data prior to transferring the data across the Internet.

Distributed Common Object Model (DCOM): The Distributed Common Object Model (DCOM) framework is another example of a CBA technical implementation that supports distributed transactional processing (Davis & Zhang, 2002). DCOM has become the major technology for distributed computing on the Microsoft Windows platform. DCOM offers the capability to use several conventional programming languages, such as Visual C++, Visual Basic, and C#, in conjunction with Microsoft IDL and the Windows Registry. The Windows Registry is a type of metadata for the MS Windows operating environment. Similar to CORBA, data is converted to a binary format prior to transferring the data across the Internet. DCOM uses the application-level Internet protocol HTTP to communicate with the TCP/IP protocols to facilitate data transfer across the Internet.

ActiveCOM: During the late 1990s, researchers attempted to integrate DCOM and CORBA to leverage the attractive features of both products and to mitigate some of the limitations. However, the Daniel, Traverson, and Vallee (1999) research highlighted the lack of compatibility between DCOM and CORBA. The researchers provided a framework called ActiveCOM for spanning the DCOM and CORBA interoperability gap. They were able to establish a new DCOM instance inside of a CORBA core software instance to facilitate communication between the two products. The de facto alternative for making the two middleware products communicate effectively was achieved by developing gateway software between specific modules (program to program). Using gateways added an additional software development requirement that exacerbated the problem of tighter coupling for both products, therefore increasing development and maintenance costs. They concluded the research by stating that ActiveCOM was a more flexible and efficient approach in comparison to building a gateway. ActiveCOM did not require the development of new software and thus reduced costs.

CSA

Krishnamurthy (2006) defined CSA as a method of building middleware that partitions tasks or work loads between service providers and service requesters (clients). The sharing of resources is one sided because clients do not share resources or provide services to a service provider. A task is therefore completed partially by the server and partially by the requesting client. CSA became a practical architecture after the development of smaller computers (e.g. PCs) because the PC CPU resources were leveraged to accomplish a portion of the processing requirements.

CSA can have different configurations that consist of two and three tiers. With a two tier configuration, clients will typically establish a connection with the server, make a request of the server, and then wait for a response from the server. A three tier CSA configuration will have a middle tier (often called an application server) that serves as the interface between a database server and a thin client. The application server will facilitate the processing of business rules for the application, thus allowing the client to be a smaller program (thin) with limited functionality.

P2P Middleware

Krishnamurthy (2006) defined P2P Middleware as a type of NCS where each workstation has software that has equivalent capabilities and responsibilities (e.g. the same program exists at different network locations). This configuration is different from classic CSA where the systems are designated as a client, application server or database server, which have distinct processing capabilities. P2P modules are commonly used for the sharing of music, images, games, and other software products.

Therefore, P2P architecture is a type of distributed computing referred to as collaborative computing because it facilitates the use of unused CPU processing and free disk space of numerous computers attached to a network (Krishnamurthy, 2006). Organizations such as United Devices, have harnessed the unused CPU processing power of over 2 million PCs via the Internet. Another use of P2P architecture is Instant Messaging (IM) that allows people to communicate real-time with each other via the Internet using their PCs. Many corporations have adopted IM as a commonly accepted tool needed to conduct business on a daily basis.

Service Oriented Architecture

SOA has three primary components (Huhns & Singh, 2005):

- Service Provider –an organization or individual that designs, creates, and publishes a computer software process called a service to a service registry.
- Service Registry –a public or private repository of services that have been published by service providers.
- Service Requesters –organizations or individuals that find services within the service registry either manually or by using an automated software process and then use the service to fulfill a processing requirement.

A simple example of SOA is a provider that creates a lookup service to find the USPS zip code for a particular city, county, and state. A service requestor can review a service registry for a service that provides this type of zip code information based on the metadata (data about the service) that is included in the registry. Once found, the service requester can access the service repeatedly to find zip codes by providing the city, county, and state abbreviation as input data to the service.

The SOA Service Registry is enabled by two open standards, Web Services Description Language (WSDL) and Universal Discovery Description Integration (UDDI). WSDL is a standard leveraged by SOA that facilitates the detailed description of the services including details about how a consumer can use the service (Nandigam & Gudivada, 2006). UDDI is another open standard that is leveraged by SOA that facilitates the documentation of services in a Service Registry. The UDDI standard is managed by OASIS (Organization for the Advancement of Structured Information Standards) and contains the three components of white pages, yellow pages, and green pages. The UDDI components are similar to what a consumer will find in a phone book.

The combination of the UDDI and WSDL standards as components of SOA, enable the publishing of the service to a public or private registry, thus allowing the services to be manually or dynamically discovered by consumers. Krishnamurthy (2006) noted that since WSDL was created using XML that is an open standard, WSDL is much more extensible and flexible for the addition of descriptive data. The IDL that is commonly used by CORBA and DCOM is a closed standard and is consequently very rigid, offering fewer opportunities for documentation in comparison to XML.

Although initially designed as a replacement for HTML by the W3C in 1998, XML has evolved into the generic building block for data exchange on the Internet (Bourret, Bornhovd, and Buchmann, 2000). XML has five advantages over other data exchange formats because it:

- Contains self describing metadata that facilitates dynamic interpretation and allows for validation of the data content.
- Does not require conversion to a different data format because the data is stored in the ASCII text format.
- Supports Unicode for international language text data transfer.
- Facilitates the later use of the data by applications that were not the original target application because of the metadata that describes the content.
- Is extendable, thus it provides the capability for users to embellish the foundation XML language by defining new vocabulary elements.

BPEL is another SOA open standard that is a XML-based language that is used to standardize the interaction between business processes for distributed computing (Nandigam & Gudivada, 2006). BPEL uses WSDL to define the message format for incoming and outgoing messages.

Nandigam, Gudivada and Kalavala (2005) discussed the underlying core technologies of SOA and then elaborated on the recent transition of the Internet from a people centric oriented environment to a software centric environment. They described an open source component included in SOA as the Simple Object Access Protocol (SOAP) which is an additional safeguard to improve process interoperability for the transfer of XML data across the Internet. XML data can be packaged into larger cohesive bundles using SOAP. The W3C accepted SOAP as an open packaging protocol for Internet data transfer in June of 2003. SOAP provides a verification mechanism that ensures that complete messages (e.g. a bundle of XML data) have been received at the destination network (end-point network). In addition to packaging data for transfer, SOAP supports the ability of a message to invoke a process on a remote server to gather or manipulate data in that environment.

Extraction, Transformation, and Loading (ETL) Middleware

Skoutas & Simitsis (2007) described Extraction, Transformation, and Loading (ETL) applications as a type of middleware that gathers data from various data sources, transforms the data, and then loads the data into an Operational Data Store (ODS) or a Data Warehouse (DW). Their research focused on using semantic web technologies and the development of an ontology to derive complex ETL transformation workflows to load a DW. An ODS or DW can provide value to organization because it establishes a common integration point for disparate data that originates from new and older legacy applications. The extraction process may collect data from many data sources that contain data in different storage formats such as sequential text files or Relational Database Management Systems. The transformation process may involve the cleansing and formatting of the data to a common format. Therefore, ETL workflows can be complex because they are driven by diversity of the data that must be extracted and the constraints of the destination ODS or DW that facilitate integration.

NCS Alternatives in Comparison to SOA

CORBA

One of Krishnamurthy's (2006) composite system requirements states that a NCS should have adaptability, modifiability, configurability, and runtime dynamism. He argues that CDR (binary data transfer format) is a limitation of CORBA in terms of modifiability and adaptability when compared to SOA. Although CDR facilitates faster data transfer because it reduces the volume of data (e.g. data in binary format has less volume than the same data in text format), CDR results in tighter system coupling by raising the level of system complexity. Increased system complexity often results in higher development and long term maintenance costs. Pressman (2005) discusses the Law of Demeter for processes (e.g. Principle of Least Knowledge) that states that processes should only have limited knowledge of processes in other subsystems and should only send messages to their neighbors which supports the concept of loose coupling.

According to Krishnamurthy (2006), the use of CDR forces process-to-process communication to be established at a much lower program level (e.g. tight coupling). To establish connectivity, software engineers must have intimate knowledge of the software components that are involved in the data exchange process to build a connection. The requirement for detailed technical knowledge is a severe limitation because it may not be possible to know the internal details of the underlying software at the inter-organizational communication level. Alternatively, text-based message communication that uses the SOA XML standard, allows messages to be exchanged at a much higher inter-organizational level. A higher level of communication facilitates looser coupling between the participating systems. If the lower level software components change, it does not affect the ability of the applications to communicate. Consumers of text-based messages are only required to understand the format of the data that is being exchanged and are unaware of the underlying software configuration. Since XML provides self describing metadata, the interpretation of the data that is exchanged between processes becomes much easier to accurately interpret. The use of XML helps to mitigate issues with data interoperability.

Although critics of text-based data transfer will argue that the transfer of data in text format is too slow and cumbersome, Krishnamurthy (2006) suggests that the sacrifice of data transfer speed is reasonable, especially when compared to the gain of adaptability and modifiability that results from using text-based message communication. Since CDR is a proprietary, closed data transformation process, it lacks the characteristic of openness as described by Krishnamurthy as a NCS requirement.

Kaye (2003) confirmed Krishnamurthy's argument that CORBA is more tightly coupled than other architectures such as SOA. The use of CORBA as a middleware connectivity tool requires software engineers to have a detailed technical knowledge of the participating applications. This tighter coupling has made systems development using CORBA more costly. CORBA application development must be closely controlled with a high degree of cooperation between the software engineers. This complexity of development, a lack of extensibility, and the high cost of development, have made CORBA a less attractive alternative in recent years, especially when compared to SOA.

In regards to modifiability and adaptability, Krishnamurthy (2006) also explained that CORBA is more tightly coupled than SOA because there are fewer processes involved in the data exchange process for CBA when compared to SOA. The larger size of the CORBA modules causes tighter coupling in the form of more complexity and therefore less adaptability for reuse. As Kaye (2003) described, loosely coupled systems that use SOA have smaller, nimbler processes that meet the requirements of adaptability and modifiability.

DCOM

Another of Krishnamurthy's (2006) composite system requirements for a NCS is dependability in the form of high availability, fault tolerance, resilience, and security. Davis and Zhang (2002) highlight two limitations of DCOM in terms of the requirements for a NCS:

- DCOM is primarily a windows based technology and therefore lacks adaptability because it is highly dependent on the MS Windows platform.
- DCOM applications are difficult to deploy within a corporate network infrastructure because of issues with poor security when crossing network firewalls. Both CORBA and DCOM were initially developed for corporate intranets and then later were adapted for use on the Internet.

Lee, Kim, and Park (2002) support the prior statements about DCOM by stating that there are technical and security issues with using DCOM on a Wide Area Network (WAN). They also emphasized that the lack of compatibility between DCOM and CORBA was a major limitation of the two products. They mentioned that the DCOM and CORBA were developed for general purpose operating systems and therefore were difficult to use with other platforms such as embedded control systems (ECS). They emphasize the fact that DCOM lacks adaptability with different operating environments.

The focus of the Davis and Zhang (2002) research was to compare the DCOM binary transfer mechanism and the SOAP standard. They concluded the research by stating that although the SOAP standard provides more cross-platform interoperability and stronger security than DCOM, the use of SOAP results in slower data transfer times when compared to DCOM. Because DCOM and CORBA transfer binary data, they lack adaptability and are tightly coupled when compared to SOA.

ActiveCOM

Research by Bechini, Foglia and Prete (2002) indicated that the ActiveCOM architecture had a much slower (unacceptable) run-time performance when compared to custom built gateways between CORBA and DCOM. They also questioned the prudence of integrating two disparate middleware products like CORBA and DCOM, as opposed to encouraging an organization to standardize on one Middleware platform.

CBA Alternatives in Comparison to SOA

SOA offers a number of advantages over the various CBA approaches by virtue of the following features (Huhns & Singh, 2005):

- Views disparate systems as components that can be integrated with loose coupling.
- Transcends various applications technologies and operating systems.

- Supports high level system to system communication via the Internet.
- Uses (vendor) neutral, commonly accepted communication protocols.
- Offers discoverable services via a catalog.

In comparison to CORBA and DCOM, SOA Web Services do not require the intimate technical knowledge of the remote applications and therefore is more loosely coupled (Kaye, 2003). As described previously, SOA messages can be sent from one system to another system without a detailed knowledge of the underlying technological configuration. Therefore, if a remote application requires technical changes on the program level, these changes will be transparent to the system that uses the remote application as long as the messaging format remains static. Krishnamurthy (2006) also noted that SOA is more loosely coupled than CBA because SOA communication is at an inter-application level in comparison to CORBA and DCOM that use program to program level communication. Therefore in our opinion, CBA is a less attractive option when compared to SOA to resolve the interoperability gap that exists between LMSs and LORs.

CSA Limitations

In regards to CSA, Emmerich et al. (2007) noted that CSA is limited because it does not offer the required scalability that is available from more robust distributed system architectures such as SOA. Scalability and acceptable performance are NCS requirements identified by Krishnamurthy (2006). The lack of scalability can result from the fact that CSA has the tightest coupling of all the NCS architectures that Krishnamurthy examined. CSA demonstrated the finest grain of program to program communication between the client and server. In addition, CSA allowed developers to by-pass the middle tier and access lower level components (e.g. database server) directly. As mentioned previously, lower level communication processes are expensive to develop and maintain. Since CSA is very tightly coupled and therefore lacks the requirements of scalability, adaptability and modifiability, we consider CSA a less attractive alternative to resolve the LMS to LOR interoperability gap when compared to SOA.

P2P Limitations

Although P2P systems are generally simpler in design, Krishnamurthy (2006) noted that they do not support heavy workloads because they often rely on the processing power at either end of the connection. Often P2P applications are implemented on smaller computers (e.g. desktop PCs) and therefore have limited processing power in comparison to a mainframe or other large types of commercial servers. Since P2P architecture does not offer the concept of a service description, the discovery of services must be accomplished by the manual review of simple text descriptions or a review of the functionality of the code. Krishnamurthy noted that in some cases, there may be a XML document that describes a P2P service. However, a P2P XML document will not follow a standard format as is the case with the standard format of the WSDL XML documentation. Also, P2P documentation may not be located in a centralized location. Therefore, dynamic discovery of P2P documentation becomes a tedious, manual task in comparison to SOA Web Services that offers a Service Registry.

Since each node that participates in a P2P network runs the same application, coupling is tighter because changes to the application affect all the instances of the application (Krishnamurthy, 2006). A P2P application has the potential to grow and become very large in size. Unlike Web Services, where the underlying technology is not exposed to the service, P2P is developed on a very low level of granularity and therefore, network communication is at a program to program level. P2P is also decentralized, so there is no concept of a central directory for service discovery. Broadcast protocols are used to discover peers on the network (Internet). Reliability of P2P architecture can be an issue because peers can join and leave the network on a regular basis, thus there is no continuity in regards to the peers that will be available to assist in a processing request.

Although P2P offers some attractive features, such as the capability to utilize unused CPU resources across the Internet, it is more tightly coupled than SOA. As described above, P2P architecture does not offer an easy mechanism for the dynamic discovery of services in a central repository. Therefore, because of the lack of the NCS requirements of adaptability and modifiability, we consider P2P architecture a less attractive alternative to resolve the LMS to LOR interoperability gap when compared to SOA.

ETL Limitations

Tziovara, Vassiliadis, and Simitsis (2007) explain that the ETL complexity will often lead to ETL workflows that are error prone and time consuming to develop. They provide detailed analysis for the optimization of ETL physical implementations for different classes of logical ETL workflows. The major objective of the research was to identify the best physical implementation for a particular logical ETL workflow. From a NCS standpoint, ETL middleware as described by Skoutas & Simitsis (2007) is an example of a platform centric system. Although an ETL middleware is able to gather data from a variety of disparate sources located across a network, ETL Middleware is considered a closed system because the information exchange is internal to the system and therefore does not meet the NCS requirement of open information exchange. In addition, ETL system control is usually centralized and the runtime behavior is not dynamic. Generally, ETL middleware operates within the confines of an intranet and consolidates data for one particular organization, and therefore ETL rarely spans organizational boundaries. Although ETL software has attractive features for data movement, we consider it a poor solution to resolve the LMS to LOR interoperability gap because it does not qualify as a NCS according the requirements of a NCS defined by Krishnamurthy (2006).

Summary

After reviewing alternative architectural approaches in terms of Krishnamurthy's (2006) six NCS requirements, SOA is the best alternative to resolve the data movement interoperability gap that exists between LMSs and LORs in our opinion. SOA provides the loosest coupling of the NCS alternatives because process communication can be conducted on the inter-organizational level via message only communication. SOA offers the strongest adaptability and modifiability of the NCS alternatives that we reviewed. SOA supports openness because it is composed of open standards. The use of open data and communication standards enhance the interoperability and adaptability of SOA.

As described previously, CBA, CSA, and P2P have much tighter coupling because they communicate on the program to program level. CBA Middleware, such as CORBA and DCOM, uses a binary data conversion mechanism prior to transferring data across the Internet. Although the binary transfer method is faster and reduces the volume of data that is transferred across the Internet, it causes tighter coupling for the modules. In the case of CSA and P2P architectures, tighter coupling reduces the scalability of the NCSs. Although P2P architecture offers attractive features, the lack of easily discovered documentation for modules (no central repository) is an issue for reuse. The looser coupling of SOA that uses nimbler, smaller services reduces development and maintenance costs in comparison to other NCS architectures such as CBA and P2P. In our opinion, SOA is an attractive alternative for Educational Institutions for a variety of reasons.

References

Balci, O., & Ormsby, W. F. (2008). Network-centric military system architecture assessment methodology. *International Journal of System of Systems Engineering, 1*(1), 271-292.

Bechini, A., Foglia, P., & Prete, C. A. (2002). Use of a CORBA/RMI gateway: Characterization of communication overhead. *Proceedings of the 3rd international workshop on Software and performance.*

Bourret, R., Bornhovd, C., & Buchmann, A. (2000). A generic load/extract utility for data transfer between XML documents and relational databases. *Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000),* 134.

Broisin, J., & Vidal, P. (2005). A computing environment for human learning at the service of learning objects virtualization. *Science and Technology Information and Communication for Education and Training, 12*, 177-204.

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.

Butler, J. C., Mayo, D. R., & Weiler, J. (2003). *Succeeding with component-based architecture in e-government.* Retrieved from http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=/published/emeraldfulltextarticle/pdf/3260010106_ref.html

Chigani, A., & Arthur, J. D. (2007). The implications of network-centric software systems on software architecture: A critical evaluation. *ACM Southeast Regional Conference,* 70-75.

Daniel, J., Traverson, B., & Vallee, V. (1999). Active COM: An inter-working framework for CORBA and DCOM. *Proceedings of the International Symposium on Distributed Objects and Applications,* 211-222.

Davis, A., & Zhang, D. (2002). A comparative study of DCOM and SOAP. *Proceedings of the IEEE Fourth International Symposium on Multimedia Software Engineering (MSE 02),* 48-55.

Emmerich, W., Aoyama, M., & Sventek, J. (2007). The impact of research on middleware technology. *ACM SIGSOFT Software Engineering Notes, 321*(1), 21-26.

Huang, Y., & Gannon, D. (2006). A comparative study of web services-based event notification specifications. *Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06),* 7-14.

Huhns, M. N., & Singh, M. P. (2005). Service-oriented computing: Key concepts and principle. *IEEE Internet Computing, 9*(1), 75-81. Retrieved from http://www.computer.org/portal/site/internet/

IMS Global Learning Consortium, Inc. (2006). *IMS tools interoperability guidelines.* Retrieved August 10, 2009, from http://www.imsglobal.org/ti/tiv1p0/imsti_guidev1p0.html

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.

Kaye, D. (2003). *Loosely coupled: The missing pieces of web services.* Retrieved from RDS Press http://fileshunt.com/rapidshare.php?file=rds+press

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

Lee, C., Park, J., & Kim, Y. (2002). A SOAP-based framework for the internetworked distributed control systems. *Advanced Internet Services and Applications, First International Workshop,* 195-204.

Nandigam, J., & Gudivada, V. N. (2006). WSEXP - A tool for experimenting with web services. *The Journal of Computing Sciences in Colleges, 22*(1).

Nandigam, J., Gudivada, V. N., & Kalavala, M. (2005). Semantic web services. *The Journal of Computing Sciences in Colleges, 21*(1), 50-63.

Pressman, R. S. (2005). *Software engineering: A practitioner's approach* (6th ed.) New York, NY: The McGraw-Hill Companies.

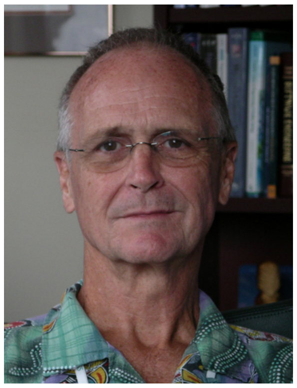
Sartipi, K., & Dehmoobad, A. (2008). Cross-domain information and service interoperability. *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services,* 25-32.

Skoutas, D., & Simitsis, A. (2007). Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *International Journal of Semantic Web and Information Systems (IJSWIS), 3*(4), 1-24.

Biographies

Bob Mason is completing his Ph.D. dissertation in Computer Information Systems (CIS) at Nova Southeastern University (Ft. Lauderdale, FL). He is currently conducting a research study us-ing the Delphi technique with a panel of international experts to determine the functional requirements for a SOA application that will help to resolve the interoperability gap challenges between Learning Object Repositories & Learning Management Systems. He has a MBA with an emphasis in CIS from the University of North Texas (Denton, TX).

Bob has served as an adjunct faculty member at Regis University in Denver for the last 10 years teaching graduate level CIS courses. In addition, he has been employed for the last 25 years by various large corporations within the areas of database administration and software engineering. He is currently employed by CIGNA Healthcare in Greenwood Village, CO.



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 Tech-nology 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 in-terests 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/