USING WEB SERVICES TO FOSTER GLOBAL COLLABORATION IN SOUND DESIGN

James A. Ballas and Justin Nevitt

Naval Research Laboratory
Code 5585
Washington, DC, USA
james.ballas[justin.nevitt]@nrl.navy.mil

ABSTRACT

The migration of client-server systems to web services using Service Oriented Architecture (SOA) design principles is widespread and likely to dominate the future evolution of computing. Use of web services is especially challenging for streaming content such as that which would be used for sound design. This paper describes the principles of a Service Oriented Architecture (SOA) and ways that it could support sound design and foster global collaboration across the web.

1. INTRODUCTION

The current trend in the computer industry is to migrate from client-server, point-to-point systems to Service Oriented Architecture (SOA) systems that provide loosely coupled components that are available as web services and that can be composed into functional workflows. SOAs are widely deployed for text content, such as search and e-commerce, but are not widely used for streaming content. This paper begins with a definition of SOA and its advantages. This is followed by a description of SOA components. The paper concludes with a description of how SOA can be used to support sound design. Our optimism about the potential use of SOA for sound design is based on our success in implementing web service systems in other domains that incorporate many of the same components as those described here [1] [2].

1.1. Service Oriented Architecture

SOA is a system in which services are deployed and registered by providers and able to be discovered and used by consumers. These elements exist as the SOA triangle [3], illustrated in Fig 1.

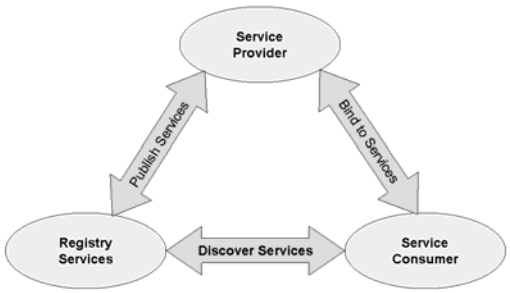

Figure 1. *The Service Oriented Architecture (SOA) triangle that illustrates the fundamental design of a web services system.*

The benefits of SOA are obtained if the enterprise is designed according to the following principles:

- distributed components that function independent of their physical location;
- loosely coupled components that can be invoked as needed;
- granular services that can be configured into workflows;
- platform independence achieved by adherence to platform-agnostic standards and specifications;
- discoverable components reachable through public interfaces.

In turn, the following benefits can be realized:

Ease of Deployment. A web service means that your code only has to be deployed at one site in the enterprise. Updating will only require redeployment to this single site.

Reuse. The deployment of core functions as services greatly increases the efficiency of code writing and execution. These core functions do not have to be replicated throughout the enterprise but instead are deployed at one endpoint with a public interface.

Agility. Software development is a time-intensive task that requires money, man hours, and other resources. SOA allows for rapid development of new functionalities to react to customers' needs. Using a technology called Business Process Execution Language (BPEL, which is described in more detail later in this paper), users can join separate pieces of functionality (services) together to form a new business process or workflow. The orchestration of several services takes a fraction of the time it would take to hand code a new application or adapt a legacy system with the same functionality [4].

1.2. Service Description

The use of web services depends on publicly available service descriptions. The specification of a service requires a Web Service Description Language (WSDL) document as well as data descriptions in the form of xml schema documents. The WSDL describes the interface, ports, messages, and binding for a web service. The schema documents are XML Schema Definitions (XSD) that include standard data types (e.g., string, integer) that are recognized universally, as well as unique data

types that are defined for the particular web service. Universal data types might include data types that have been established by an international standards body (besides Oasis, which is the controlling organization for the web). For example, one set of metadata that is commonly used is that from the Dublin Core Metadata Initiative (DCMI). While the DCMI is well developed for some domains, it is notably sparse with respect to audio, containing only the following metadata definition for *sound* among its basic types:

```
<rdf:Description
  rdf:about="http://purl.org/dc/dcmitype/Sound">
 <rdfs:label xml:lang="en-US">Sound</rdfs:label>
 <rdfs:comment xml:lang="en-US">A resource primarily
  intended to be heard.</rdfs:comment>
 <dcterms:description xml:lang="en-US">Examples
  include a music playback file format, an audio compact
  disc, and recorded speech or
  sounds.</dcterms:description>
 <rdfs:isDefinedBy
  rdf:resource="http://purl.org/dc/dcmitype/" />
 <dcterms:issued>2000-07-11</dcterms:issued>
 <dcterms:modified>2008-01-14</dcterms:modified>
 <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
  schema#Class" />
 <dcterms:hasVersion
  rdf:resource="http://dublincore.org/usage/terms/history/
  #Sound-003" />
 <dcam:memberOf
  rdf:resource="http://purl.org/dc/terms/DCMIType" />
 </rdf:Description>
```

There are other metadata documents that address aspects of audio material. The Digital Library Foundation has defined the Metadata Encoding and Transmission Standard (METS.XSD) that includes elements to encode descriptive, administrative, and structural metadata about digital objects, including digital audio recording; but as with the DCMI, there is little in its specification that is audio related.

1.3. Service Registration and Discovery

For web services to be used in new workflows, they have to be registered in the equivalent of a "yellow pages." The primary method by which services are registered and discovered (located) in a SOA is through common registries based on the Universal Description, Discovery, and Integration (UDDI) specification. UDDI is, in effect, like a phonebook for web services; both the service location and a technical description of the service offerings are available.

When the UDDI specification was being developed, several public registries were available until 2006, but were shut down with the conclusion of the successful testing of the UDDI technology[1]. While the UDDI specification is supported in commercial products (e.g., Microsoft's Office suite has a web services toolkit, which includes search as part of its Visual Basic system), there is limited public registry. One exception is http://seekda.com/, which uses crawlers to find service specifications (WSDL files), as well as to provide a mechanism to register services. Its directory is searchable through a browser, but not through the UDDI inquiry operation. A search using "audio" resulted in 86 hits. Fifty-two of these were Amazon's E-Commerce Service, which includes a capability for a vendor to describe the audio format of a product. The remaining 34 hits were for a variety of services including services to upload, download, create and play audio files, as well as to retrieve audio attributes. The service description includes an availability metric, generated by periodically testing the service.

Seekda.com also provides a mechanism to test the web services it has registered. For example, faces.com has defined a set of services to support sharing of photos and music, as well as blogging. Audio files are added to tunefeeds using a SOAP operation called Tunefeed_AddAudioInfo (string Signature, string AudioListXml, string AuthKey).

1.4. Service Binding

Use of services requires an application that can send and receive SOAP over HTML. This application will attach to the service endpoint through a process called binding. A notional binding specification for an Audio Design service is as follows:

```
<wsdl:binding name="AudioDesignerServiceBinding"
 type="tns:AudioDesignService">
 <soap:binding style="document"
 transport="http://schemas.xmlsoap.org/soap/http"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
 <wsdl:operation name="DesignRequest">
 <soap:operation soapAction="" style="document"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  <wsdl:input>
 <soap:body use="literal"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  </wsdl:input>
  <wsdl:output>
 <soap:body use="literal"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
  </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
```

This binding defines an operation called DesignRequest, that will receive an incoming message and respond with an outgoing message. The formats of these messages are defined in other segments of the WSDL. For example, an input message called "DesignRequest" would include three parts when defined as follows:

```
<wsdl:message name="DesignRequest">
   <wsdl:part name="RequestDescription"
 type="xsd:string"/>
 <wsdl:part name="AudioOutFileName"
 type="xsd:string"/>
<wsdl:part name="DataInFileName" type="xsd:string"/>
 </wsdl:message>
```

1.5. Service Orchestration

Service Orchestration is a technology that links services together to create workflows which exhibit greater functionality

[1]See http://uddi.microsoft.com/about/FAQshutdown.htm for a description of the decision to shut down the public test UDDI servers.

when working with a suite of services than when working with individual services. Electronic business might utilize orchestration to implement a workflow that finds products from vendors, gets a vendor selection from a user, does electronic billing, generates purchase orders, and contacts a shipper for pickup and delivery of the product.

Orchestration technology is described in the BPEL4WS (Business Process Execution Language for Web Services) specification. The summary of the specification from the OASIS technical committee is as follows:

> BPEL4WS is an XML-based language enabling users to describe business process activities as Web services and define how they can be connected to accomplish specific tasks. WS-BPEL is designed to specify business processes that are both composed of and exposed as web services.

In 2007 IBM, SAP, Oracle, BEA, Adobe and Active Endpoints formally submitted the BPEL4People extension and WS-HumanTask specification to OASIS for approval. WS-HumanTask and BPEL4People are standards proposed to expand on BPEL's specification. They allow BPEL to completely model a business process, including complex human tasks and interactions. BPEL4People is a specific set of extensions to the BPEL (WS-BPEL 2.0) language that outlines a human task much like a call to a service. It allows service orchestrators to include human interactions at the orchestration level (BPEL4People). WS-HumanTask is the web services specification that describes the human tasks, assignments of people to tasks, and task parameters such as human role, task assignment (to a person) and task deadline. With this extension of the BPEL language, orchestration of workflows moves beyond machine-to-machine and specifically includes human activities.

1.6. Service Security

Web services typically will require some type of security to restrict the use of the services, and the data that these produce, to authorized users. As web services have been adopted widely for business, service security is an established technology. Several types of security are available. On the wire security is provided by the use of secure socket layer (SSL) transport. This is simpler to implement than one might expect. It requires SSL certificates that are digital signatures. These can be self-signed or can be obtained from government or private organizations. With certificates in hand, they are loaded into the appropriate keystores, and the web server is configured for SSL operation. User authorization requires a second level of security employing username and passwords to authenticate users. A moderate level of security can be obtained by requiring a username/password to gain access to a web site, but typically these are html forms that employ the *INPUT type=password* which simply echoes "*" when the user types the password. The actual password is not secured and is passed back from the browser as clear text.

A stronger method to restrict access is to employ username/password as described by the SAML specification. This encrypts the username and password in the header of the web service call. Currently, skilled Java programming is required to implement this capability. But as web services become more established, out of the box implementations will become common, especially for open sourced, Linux-based systems.

1.7. Content Discovery

Search capability is currently driving the economy of the web, most clearly demonstrated by the success of Google. The commercial search companies succeed by producing, indexing, and owning metadata about web content. But web sites often have internal search capability, demonstrating that content discovery can be implemented on any database. The U.S. Department of Defense invested in and demonstrated the feasibility of federated search technology. This is a design that uses a central query service to receive user input, parse it, and then transmit the queries to data providers through a search web service (SWS) specification. The data providers employ their own method of searching their data bases, and return messages with hits and URL links to the data itself. A similar design is behind Amazon's Open search specification.

A major limitation of content discovery capability is that it does not function well for media, unless the files have been tagged. However, by incorporating content analysis services, such as audio classifiers, into a search workflow, a functional search of web audio content could be achieved. The content analyzers would be designed to expose their capability as a web service. The workflow would apply these services to databases, and return the results back to the user.

2. WEB SERVICES FOR STREAMING CONTENT

The web services paradigm is a powerful one that can be applied to many different IT problems. There has yet to be much done with web services in the area of streaming audio content. This is expected, as the basic specifications for web services heavily emphasize text-based content. Nonetheless, the potential applications of a web services framework for streaming audio content are extensive. After overcoming certain hurdles inherent to conducting business with loosely coupled entities, one will be able to do many of the normal auditory processing tasks usually relegated to thick apps on workstations. All the advantages of using web services will follow. Potential applications are described in the following sections.

2.1. Sonification application

Imagine that a developer has invented an algorithm that creates a particularly useful sonification of data. Using this algorithm, a stock broker would like to construct a live sonification of stock watch data that is available in xml form from a web service. The resulting sonification would be available from a streaming server, and the broker has a webapp to listen to the audio stream from this server. If all these functions existed as a web service, one could easily orchestrate them together to get the desired effect. In addition, if you later decided to use a different data source or a different sonification algorithm, you could substitute those with ease. The architecture for this is illustrated in Fig 2.

2.2. Search application

Imagine that a sound designer is tasked with finding an archetype example of a particular voice utterance and has available an immense library of audio recordings from the radio

industry; however, the designer cannot listen to more than one recording at a time. A web service orchestration that combines a speech-to-text service and a search service would provide a solution. The speech-to-text service would be called to generate the text of the audio recordings, and its output would be searched by the search service. In practice, the audio recordings would probably be pre-indexed, just as the Google appliance is used to crawl the web searching for and indexing documents. Such a crawler would be an enterprise implementation of machine listening. The architecture for this application is illustrated in Fig 2, which happens to be the same architecture for the sonification application.

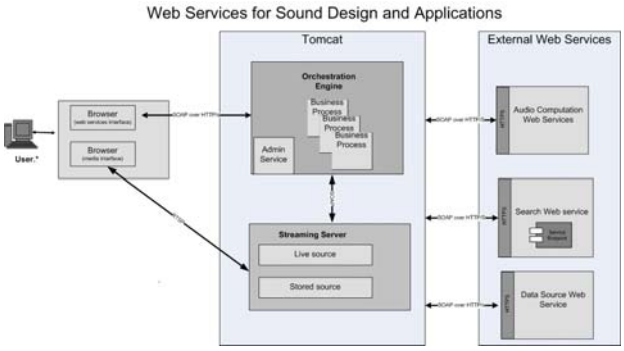

Figure 2. *Service Oriented Architecture to create, manipulate, render and deliver audio content.*

2.3. Sound design application

Imagine that a company develops and hosts musical sound editing software that is completely web based. This capability uses a set of web services that interface with sound generation services. All of the computation is performed on a centralized server. The user's computer has only a light-weight web application. This release is widely adopted by a global community of sound designers. The architecture for this is illustrated in Fig 2, an architecture that is also the basis of the previous two examples.

2.4. Architecture summary

These examples require an architecture with the following components:

- a streaming server that can be controlled by a publicly available web service;
- web services that can perform computations on audio content, including generating it under algorithmic control, searching audio files and classifying audio segments;
- orchestrations of publicly available web services including the service controlling the streaming server;
- web applications that can access these web services and receive streaming content.

The following section describes examples of the components illustrated in Fig 2.

2.5. IBM Streaming Server

The development of a suite of web services for audio design and its applications is notably advanced by IBM's Streaming Server (http://www.alphaworks.ibm.com/tech/streamingengine). This server can be hosted in Apache Tomcat, an open source web server. It has a WSDL that defines the web service calls that can be made, and can generate media streams from recorded or live sources using the Real Time Streaming Protocol (RSTP) for control and the Real-time Transport Protocol (RTP) for transport. RSTP is a tape-like control protocol (e.g., play, pause, record). The media streams that are currently supported are MP3 audio and MPEG-4 video. The WSDL defines 147 elements, 88 messages, and 88 operations (e.g., as start and stop the server, etc). One of the operations is createLiveAssetSession. This operation expects a message with the following elements:

```
<element name="createLiveAssetSession">
<complexType>
<sequence>
      <element name="name" type="xsd:string"/>
      <element name="srcType" type="xsd:int"/>
      <element name="srcHost" type="xsd:string"/>
      <element name="srcPort" type="xsd:int"/>
      <element name="srcName" type="xsd:string"/>
      <element name="mcast" type="xsd:boolean"/>
      <element name="archName" type="xsd:string"/>
      <element name="archDur" type="xsd:int"/>
      </sequence>
</complexType>
```

The response is simply a Boolean that the session was or was not created. This example from the WSDL illustrates the control that an application can have over this server, such as the URL source (srcHost is the URL of the live source; srcPort is the IP port at this URL). The WSDL definition of the message elements does not enumerate what is expected in the parts of this message, but these could be easily added. The streaming server has been tested and we have verified that it can be controlled through web service calls that conform to the WSDL specification[2].

2.6. Audio Computation

While a streaming engine can deliver content, and a web services streaming engine can be controlled through web service calls, the actual media content has to come from some other source. An example of audio computation technology that can be quickly demonstrated, but that also has impressive growth capability, is the python-based boodler system (http://sourceforge.net/projects/boodler/). This is a system to generate soundscapes, using agents that start, stop, and parametrically manipulate sound segments. It can produce raw sound files as well as MP-3 and it can be exposed as a web service using SOAPy (http://sourceforge.net/projects/soapy).

[2] After starting the server and verifying its operation, we loaded the WSDL into the eclipse Web Services Explorer, and successfully made calls to the published operations.

Another type of audio computation would be analysis technology that would take sound files (either recorded or live) and extract features about the sound such as its source event. For example, one of us has recently published source code for an audio classifier [5]. This classifier determines whether a sound might be classified as a propeller airplane. The algorithm uses statistical properties of the sound to make this classification, and the final decision algorithm could be modified for other types of events. This code could be converted to Java, and hosted as a web service. Alternatively, the windows executable could be exposed as a web service through the Java Runtime.exec() method. We have successfully implemented this option and confirmed that the classifier can be called as a web service and will correctly process the files that have been specified in the web service call.

2.7. Orchestration

A streaming server and audio computation services are critical components of this SOA, but their usage requires a development environment such as Eclipse as well as an experienced programmer. An alternative to these is the web services orchestration design and deployment environments offered by several vendors. ActiveBPEL Designer is an environment we have been using to construct web services workflows. The Designer produces BPEL code that is executed by an open source BPEL engine that is hosted in a web services container such as Apache Tomcat. An example of an orchestration design is illustrated in Fig 3. This workflow receives a request to create a live source, checks whether the server is started (and if not, starts the server), enables the Real Time Streaming Protocol (RTSP) on the server, copies the parameters for the live source into a web server message to the streaming server, and finally sends the message to create the service to the streaming server. The result of this request is returned to the user to complete the workflow. This design could be easily modified to call ancillary services such as those described in previous sections (e.g., create a soundscape).

2.8. Browser Applications

The last component in Fig. 2 is the web application, typically a browser, that can utilize the web services and receive the streaming content. Two issues associated with any browser application are real-time rendering and proprietary coding. It is well established that the protocol typically used for text content (e.g., HTTP) is not designed for streaming content. The RTSP and the Real-time Transport Protocol (RTP) were developed to support streaming applications. RTSP uses a tape-like control interface (e.g., play, pause, record). Streaming servers such as QuickTime and its open source version, Darwin, utilize RTP/RTSP as does the IBM streaming server described earlier. Utilization of these protocols can sometimes cause issues if firewalls have been configured to block their use or if the server is using the UDP transport layer and it is blocked.

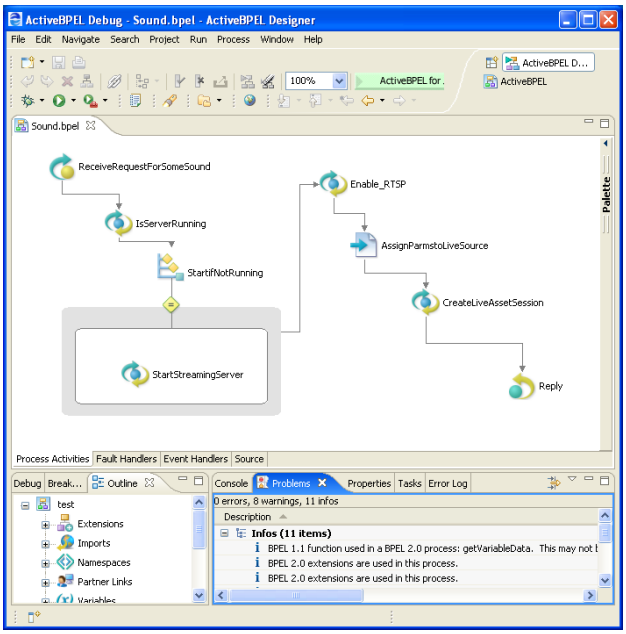

Figure 3. *BPEL workflow to control an instance of the IBM Streaming Server, starting it if necessary, enabling RTSP, and creating a live source. This visual design is saved as a deployable file that is executed by a BPEL engine.*

The second issue associated with the browser is proprietary coding, such as that utilized by Adobe Flash. In order to improve real-time performance, streaming content is often compressed and encoded. While proprietary coding might provide some features that are not available in an open coding scheme, its use in a SOA enterprise will require all the web services to use that proprietary format. Unless the commercial vendor offers a decoder, services that require access to the raw waveform could not be developed. For example, a service that would generate metadata for the audio content would need access to the raw audio. The widely used MP3, which is the audio layer of MPEG-1, is a proprietary format, and its use must be licensed for commercial applications (see www.mp3licensing.com). Open source coders/decoders are available for applications that need access to the raw audio. Finally, the browser will need a player application that is capable of handling the protocols, coding, and the computer's audio hardware. Alternatively, IBM's alphaworks has another suite of technologies, IBM Presenter and IBM Viewer, that also could be incorporated into a web services system (http://www.alphaworks.ibm.com/tech/personalpresenter).

Besides hosting a player, the browser would host the interfaces to other web services that are illustrated in Fig 2. Once a suite of services is available, workflows could be composed and the BPEL orchestration can include BPEL4People capability that would add user control and feedback to a workflow process.

3. CONCLUSION

The payoff of SOA lies in the potential to incorporate data sources, search engines, user-facing services (user interfaces), computational services and other computer capabilities hosted locally but available globally. Data from financial, government, or entertainment sources could be tied to sonification algorithms which would be generating audio content for the streaming server. Sound designs could be collaborative enterprises with individuals hosting disparate services such as filters, effects, generators, mixers, and renders. A SOA that is focused on sound and its design would be an enabling framework for global collaboration.

4. ACKNOWLEDGMENTS

The preparation of this paper was sponsored by the U.S. Naval Research Laboratory. The views and conclusions expressed in this paper are those of the authors and do not reflect the official policies, either expressed or implied, of the U.S. Navy.

5. REFERENCES

[1] D. Jones, K. Kerr, R. Carr, J. Olsonbaker, J. Cook, T. Tsui, D. Brown, J. Ballas, J. Stroup, K. Wauchope, & L. Aker (2005). Environmental Visualization and Horizontal Fusion. *Proceedings of the Battlespace Atmospheric and Cloud Impacts on Military Operations (BACIMO) Conference*, October12-14, Monterey, CA.

[2] J. Nevitt and D. Brown. *A Search Relevance Algorithm for Weather Effects Products*. Memorandum Dec 2005-Oct 2006, Naval Research Laboratory, Washington, DC, 2006.

[3] A. Michlmayr, F. Rosenberg, C. Platzer, M. Treiber, and S. Dustdar, Towards recovering the broken SOA triangle: a software engineering perspective. In *2nd international Workshop on Service Oriented Software Engineering: in Conjunction with the 6th ESEC/FSE Joint Meeting*, Dubrovnik, Croatia, September 03 - 03, 2007, IW-SOSWE '07. ACM, New York, NY, pp. 22-28.

[4] M. Juric, B. Mathew, and S. Poornachandra. *Business Process Execution Language for Web Services*. Birmingham, AL: Packt, 2006.

[5] J. A. Ballas, D. Brock, and H. Fouad, An Introduction to Sound Classification. In K. Greenebaum & R. Barzel, *Audio Anecdotes III: Tools, Tips, and Techniques for Digital Audio*, Natick, MA: A K Peters, 2007.