# Service Oriented Architecture (SOA) *and* Web services

Mark Lindquist

Enterprise Software Architecture Integrative Project

Instructor

Dr. Kris Jamsa

Date

12/17/2010

## Abstract

Service Oriented Architecture (SOA) can be described as an architectural style of software development whereby enterprise goals and objectives are encapsulated into unique functionality in the form of available "services." These services are developed as independent pieces of business logic- unlike traditional Object Oriented (OO) architectures in which the components are all interrelated. "Independence of the various interacting organizations, or at least of the services comprising an SOA, is fundamental to the SOA vision" (Taylor 2010). The idea is that the service interface or behavior, representing some business functionality, becomes available to other services and consumers in hopes of providing something unique to the enterprise. Web services are a popular service for enterprises trying to establish their presence online. Two are used in this project as is a java GUI desktop application to run the services.

Table of Contents

List of Figures



.

## Executive Summary

A service can be thought of as a component in a software system that is deployed independently from other components on a network while serving a particular task for the enterprise. Client programs, or other services, can then interact with the service.

If these independent services adhere to certain service oriented design principles; loose coupling, autonomy, abstraction, statelessness, reusability, discoverability, interoperability and composability then you have services that serve the enterprise in various ways. You instill these principles into the services in hopes of realizing certain enterprise goals. By far the most common type of service in today's business environment is the Web service.

# Chapter One: Introduction

SOA arose out of a need to respond to changing markets, initiated by an increasingly changing public. IT systems were being developed to keep up. Organizations who meet this need today are considered agile and smart. They see that IT must reflect in their design and development of software this changing nature of business, brought on by a changing clientele.

Consequently, within an organization business and IT need to align their processes so that each can change synergistically (together to support a goal). IT departments have to design into their software the change proposed in the business processes they model. These business processes should further reflect the changing nature of the market. What this does is allow for an organization to become agile, a desired quality for the successful enterprise. Agility reflects the ability for an organization to change in a constantly changing and ever increasing smart world.

Within SOA initiatives this agility is in fact a cornerstone and considered by many as the reason for the vision that SOA proposes. SOA promotes enterprise agility. How? Services in an SOA are designed to accept change. They accept this change by aligning with the changing initiatives of the enterprise.

## Chapter Breakdown

This final paper about the SOA shows what an SOA is and how it works. The introduction starting above in chapter one first describes business's interest in SOA. It goes on below to describe a precursor to SOA, Software as a Service (SaaS). Chapter two explains what the accompanying service oriented design principles mean to the service architecture. Chapter two is also about the service and service orientation. It describes the Web service, a

common type of service implementation, what it is and how it is made up. The Web service has become the service of choice for all but a few organizations on the Web.

In chapter three the project takes a turn as it relates SOA in a different context. It explains why cyber technology is different from everyday technologies we readily recognize. It goes on to describe why SOA as a cyber technology has ethical issues involved with its design and development. Privacy and the exchange of data within an SOA are discussed.

Chapter four is about the design and development environment of an SOA. Software architecture, the early design decisions made about a system, are discussed. The SDLC methodology Extreme Programming (XP) that guides its development in some instances is described. Chapter four also relates on the data layer and the management of the people and technology of SOA projects.

Chapter five is about creating a MailingList Web service while also integrating another Web service for database insertions. The MailingList Web service writes a member's information to a file on the Web server's hard drive. This might be useful if clients avail themselves of the service to subscribe to a company's mailing list. The Web server would be owned by the company. Any client, from a simple user to a large organization will have access to the Web service as it independently waits for requests to sign up customers.

The second Web service inserts the info that the MailingList service used to enlist customers with, into a database for permanent storage. This Web service was created in a previous project, hence the name CustomerPurchaseOrder, and serves as a demonstration of how a service becomes reusable. With a few modifications of the CustomerPurchaseOrder source code it became reusable in this project.

## Software as a Service (SaaS)

SOA didn't happen on its own one day where organizations woke up and saw a new vision that would change the way they deliver software. As an early form of service oriented architecture, SaaS was a new way of providing functionality to a changing enterprise.

### Understanding SaaS

SaaS, first appearing on the scene in 2001 in an article titled "Strategic Backgrounder: Software as a Service," is a software delivery mechanism that positions software application functionality on the internet with which other organizations may avail through a subscription, pay-as-you-go, or free of charge basis. The idea is that a software system provided by company "A" exposes system behavior "a" to companies B, C, and D over the Internet for a fee.

The providing organization is responsible for the licensing of the software saving the subscribing organization from having to do this. The providing organization is also responsible for designing, developing, and managing the software freeing the subscribing organization of all the related resources, activities, time, and expenses.

To gain a better feeling of how SaaS works it can be compared to traditional Object Oriented (OO) development. The important distinction and the primary benefit is that services within SaaS are single instance, multi-tenant (client) entities that have a centralized nature. Traditional component style development is instead multi-instance single-tenant and is de-centralized.

To understand this concept SaaS can be thought of as like going to a movie theatre to see a movie (a movie represents the service and a movie patron represents a user of that service). The movie house presents the movie as a centralized service available to any user

who comes to watch and pay a fee. This is analogous to the service sitting in the cloud waiting for and receiving requests from customers- “Cloud” can be thought of as the Internet.

The movie is multi-tenant meaning many people can take part. Likewise the service is multi-tenant allowing anyone to participate. This is accomplished because it is a single centralized instance instead of many instances where, like a movie, different clients watch instances from different homes.

## Properties of SaaS

SaaS applications are defined by three characteristics; software licensing, software location, and software management. In traditional localized applications software is generally licensed once and is good for the life of the product. On the other end of the spectrum are SaaS applications in which the software is subject to a metered (subscription) licensing procedure by consumers of the service.

Secondly, in on-premise systems the organization must design and create the software themselves adding enormous effort, cost, time, and resources to its agenda. In a SaaS application this management of the project is handled by the providing organization freeing up manpower to concentrate on other important enterprise objectives.

The third characteristic, location, simply refers to where the service is located. The providing organization hosts the software in a SaaS environment whereas in an on-premise application the IT department is the host.

# Chapter Two: What is an SOA?

An SOA can be thought of as an architectural style of software development in which the service, representing some business initiative, makes itself available over a network to other service(s) or consumers (clients) for the purpose of transporting data or processing information. This service within the SOA is uniquely identified by smart organizations as having certain design characteristics that serve them. Theses characteristics are discussed next.

## SOA Design Principles

An SOA, one that aligns with strategic goals and objectives of an enterprise, is more that just the technologies that make it up. With an SOA "...it isn't just the use of XML and Web services, and it's a good deal more than a development methodology" (Russell 2004).

We can begin now to look at the distinction that enables a service to become part of an SOA that serves the enterprise. The important distinction is the service oriented design principles discussed below. By instilling these design principles into service design and development the SOA grows to become a visionary SOA, one that serves the enterprise by isolating key business logic.

Each design principle effects the organization as it is designed into services. For example the design principle that states a service should have a service contract means a service interface must be designed and implemented with the idea that it will be exposed to other services via this contract. This means considerations must be taken into account when designing services.

Below is a complete list of the service oriented design principles and their importance within the organization. The aim of each principle is to define what properties a service must posses to realize an SOA that aligns with the enterprise. I begin each principle with a quote

from Thomas Erl, the renowned author on SOA. A service architecture that follows these design principles will have more leverage in realizing an SOA for the enterprise that meets its needs as expected.

1. **Service Contract** – "Services express their purpose and capabilities via a service contract" (Erl 2009). In an SOA when two services communicate there must be a mutual understanding between processes about the information passed in the message. This understanding consists of descriptions of the service behavior (the operations) and its data. It also concerns the implementation of the operations themselves. This agreement between services is fundamental to the SOA.

   When Web services interact, for instance, it is the WSDL file that provides this understanding. The WSDL file represents an abstract interface definition and concrete implementation definition. These two definitions form the service description or contract between two services which must be mutually understood by both requestor and provider.

2. **Loose Coupling –** "Coupling refers to a connection or relationship between two things. A measure of coupling is comparable to a level of dependency" (Erl 2009). In service orientation this dependency must be of a limited nature as each service provides independent behavior that is separate from other services. The service logic does not rely and is not relied upon in a true loosely coupled service. This independent behavior serves the enterprise by allowing for reuse while promoting continued reliance on service functionality by consumers.

3. **Abstraction – "**On a fundamental level, this principle emphasizes the need to hide as much of the underlying details of a service as possible" (Erl 2009). When a consumer consumes a service the processing logic or service behavior is hidden away from the

consumer (representing a Black box). The complex details are hidden. This makes services more reusable. The only visible aspect of the service is the immediate need of the consumer.

4. **Reusability**- "Reuse is strongly emphasized within service-orientation; so much so, that it becomes a core part of typical service analysis and design processes..." (Erl 2009). A service has qualities making it reusable; autonomy, loose coupling etc. It becomes valuable to the whole enterprise as a result of being reusable. A traditional component class in a software system is tied or coupled to other components. Services are agnostic from other services. They employ their logic independently from other service logic, within some context, making them reusable.
5. **Autonomy –** "For services to carry out their capabilities consistently and reliably, their underlying solution logic needs to have a significant degree of control over its environment and resources" (Erl 2009). The solution logic of a service should serve a particular interest only for the enterprise. A service should not contain solution logic that is coupled to other services. It should serve its own purpose within some context of the enterprise.
6. **Statelessness –** "The management of excessive state information can compromise the availability of a service and undermine its scalability potential" (Erl 2009). If a service is tied to its state, or its representations of various conditions of a system, it becomes less available to consumers.
7. **Discoverability – "**For services to be positioned as IT assets with repeatable ROI they need to be easily identified and understood when opportunities for reuse present themselves" (Erl 2009). A service is best when it is reusable. A reusable asset gives a good ROI when it is designed and developed as being easily discovered. Quality of

communication (with other services) and its individual capabilities should be examined for this discoverability to materialize.

8. **Composability – "**As the sophistication of service-oriented solutions continues to grow, so does the complexity of underlying service composition configurations" (Erl 2009). Services form alliances with other services. Independence is the cornerstone for a service in a service oriented environment, but hooking into other services programmatically to serve the particular interest of the enterprise, is becoming more and more complex. Designing services so that they can easily enlist other services to satisfy an interest of the organization is the goal of this principle.
9. **Interoperability – "**interoperability is fundamental to every one of the principles we just described" (Erl 2009). Services by their very nature should be interoperable or able to operate with or hook into other systems easily. Designing services to be interoperable is part of the goal of all the previous design principles discussed.

## The Service

The service positions itself outside of the application scope meaning it's developed with a wider emphasis on enterprise needs than the component classes it works with. It represents some unique enterprise initiative and is discoverable by other services through a service contract. It is this service that sits as the enterprise's main asset when designed properly.

A service is similar to traditional components comprising a software system in that they provide independent functionality. Traditional component development creates behavior in code, in the form of a method or function, which is tightly integrated with other components. A service, on the other hand, serves its own interest within a network apart from the classes and components comprising the application.

It may enlist the help of component classes or other services, either internal or external, but these other entities just become a part of the service. A service can therefore be made up of other component classes and/or other services. This configuration of services and component classes to meet a central need for the enterprise is referred to as service orchestration.

Services interact with other services, the way objects in an object oriented system would. The service's solution logic would represent the object's methods that interact with other objects only with services you're interacting over a network, in a distributed environment, with other services.

Furthermore these available services represent some enterprise wide need as opposed to components that just serve the application environment. Services strategically then may serve as a commodity that an enterprise can situate on a network where each service is able to interact with consumers.

Thus the service solution logic becomes a vital asset, if properly designed, as it waits for requests from consumers on a network.

## Web Services

The most common type of service used in business today is the Web service. The Web service distinguishes itself by communicating data formatted as XML over network protocols, usually HTTP. The Web service typically adheres to the design principles as it is part of service oriented architecture.

Web Services are the service of choice for most organizations. The Web service technologies that underlie it are a collection of specifications customized for the purpose of communication between disparate pieces of business logic over a network to serve some enterprise wide initiative.

Simple Object Access Protocol (SOAP), Extensible Markup Language (XML), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI) are all specifications used to support the SOAP and XML based Web service. Depicted below are the relationships between these specifications as they enable communication between Web services. Each one is discussed further below.

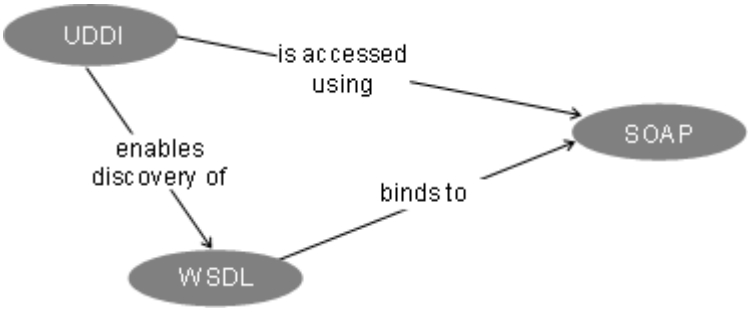


*Figure 1.* First generation SOA model. Source- SOABooks.com

For superior support, support for the enterprise and a realization of a portion of Web 2.0, one has to develop services that adhere to the Second Generation WS-* Web service standards. These standards are appended to the first generation Web service principles and provide to the Web service, and the organization, features not present but envisioned in the first generation of Web services. See Figure 2.

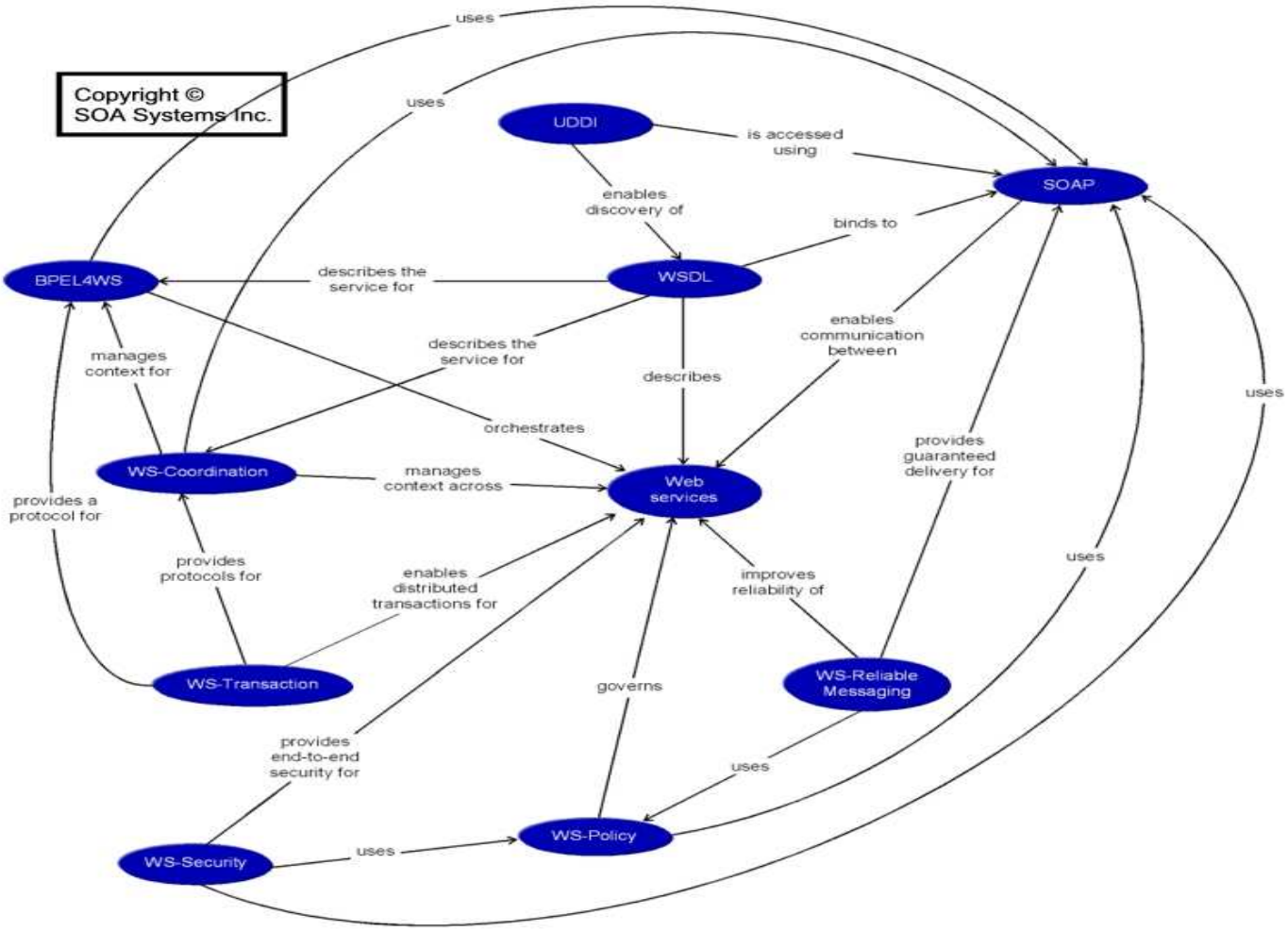


*Figure 2.* Second generation WS-* Web services model.

**Extensible Markup Language (XML)**

XML is the tool in the Web service arena. It is only a tool in this respect. Each specification mentioned in this section utilizes XML to perform some function. XML, on the other hand, does not utilize any other specification. It is used by but does not use.

XML formats data (marks-up), which can then be transported via SOAP over a network. Because XML is text based Web services are supported over many platforms, programming languages and operating systems. WSDL and SOAP as well as UDDI rely on it.

If you know HTML and its pre-defined tags you can understand the markup in an XML document. An XML documents marks-up content using customized tags that serve a particular interest as opposed to HTML tags that have pre-determined meaning.

**Web Services Description Languages (WSDL)**

The Web Service Description Language is an XML document and describes the Web service to service consumers. The WSDL document, formatted in XML, provides a description of a Web service.

Specifically the WSDL document describes an abstract definition of the service interface (behavior). This amounts to a high level description of what the service does. There is also a concrete, implementation definition that defines the operations and data used in the transport. This would be the low level description of the service operations (methods) and data to be transported.

Together these abstract high level and concrete low level definitions make up the contract between services- the service description. Each Web service and Web service consumer has this service description or one similar so they can communicate. In fact any service in an SOA shares a contract about the service. This is a vital principle of the service oriented architecture which is listed in the Design Principles section. The WSDL customized tags for these service definitions, both abstract and concrete, are mentioned now.

The abstract interface definition in a WSDL document is defined by the XML customized tags DEFINITON, INTERFACE, and OPERATION while concrete implementation definitions of the operations and data are defined with the XML tags SERVICE, BINDING, and OPERATION. Together these tags provide to the consumers of Web services a means of understanding and accessing a Web service.

These XML tags serve in their entirety as the service description or WSDL document. They abstractly define what the service behavior does while also supplying the concrete implementation behavior as well.

**Simple Object Access Protocol (SOAP)**

We now know that XML is the tool for Web services, WSDL provides a description, but how do you access the content of a provider in one application from another. The communication channel that allows for the transfer of information from consumer to provider and from provider to consumer is provided by SOAP.

As an XML based specification SOAP provides a set of XML customized tags of its own. The purpose of these tags is to send and receive messages formatted and described in XML between the consumer and service. ENVELOPE and BODY tags are required. The HEADER tag is optional and is used to carry information such as the WS-* next generation Web service standards features.

A SOAP node- a node is used to describe the SOAP roles in this context - is that of a sender and a receiver. Corresponding to the service requestor and provider roles of the service message, a SOAP *sender* sends the message request to the SOAP *receiver*, also a service provider.

In essence you have a SOAP node (See Figure 3.), the sender, wrapped around the message request on one end and a SOAP node, the receiver, wrapped around the message response at the other end. The transfer of data is then allowable through XML and executed by SOAP.

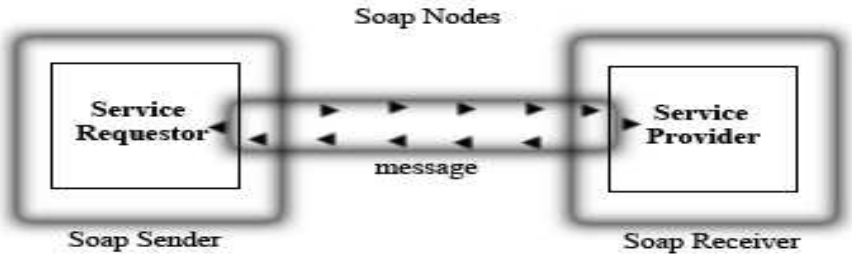


*Figure 3.* Sending and Receiving service data.

**Universal Description and Discover Language (UDDI)**

This specification serves as a repository for publicly available services in which other services may avail. The repository can be public, as available to many organizations, or private, available within a single organization.

Web services thus contain the necessary ingredients to discover a service, define a service, and transport data between services. They adhere to the design principles discussed earlier making them an ideal service for an SOA that serves enterprise wide needs.

# Chapter Three: Ethical and Legal Considerations

## Cyber Technology

We all recognize in our lives certain new technologies that come along and with them rules, regulations, policies, and social norms that guide their development and use. These non cyber technologies perform a particular task only, lacking the characteristics of cyber technologies. A DVD player can not microwave a meal. A television remote control can not make coffee. Cyber technologies, on the other hand, *"....can be shaped and molded to perform a variety of functions*" (Tavani 2007). Cyber technologies are not tangible. They live in a virtual world. They can change readily.

This concept about cyber changing technology is what James Moor likes to call "Logically Malleable" (Blackwell Publishing n.d.). Because computer technologies can take on many forms *i.e.* email client, text editor, developer's Integrated Development Environment (IDE), cyber technology is malleable.

As users begin to use these malleable cyber technologies it opens up new areas of exploration for newer technologies. As each new technology then follows, rules, regulations, and policies are initially absent about its existence and a "Policy Vacuum" (Blackwell Publishing n.d.) exists.

Moor further explains that it is not so easy to just set some policies to oversee new technology uses because certain vagueness and ambiguities arise about their existence. Thus a "Conceptual Muddle" (Blackwell Publishing n.d.) exists. An example will demonstrate the point made by Moor regarding cyber technology being logically malleable, full of policy vacuum and having conceptual muddle.

File sharing, the ‘ripping’ or duplicating of MP3 files onto a user’s hard drive, which today we have all heard of from sites like Napster, is now considered illegal. However file sharing some fifteen years ago was free and uninterrupted. Indeed some programmers, creating software programs, freely exchanged their products source code on a regular basis.

The advent of a new technology, file sharing or the exchange of copyrighted material over the internet, brought with it no policies for freely exchanging these files or to prevent it. The internet was new and free information was a welcome novelty.

Then a policy vacuum emerged as proprietary ownership began to take control within cyberspace and other sites such as Gnutella. Morpheus, and KaZaA began to swap information as well. The internet began to move from free information to shared information to outright ownership of information. A policy vacuum was the result of this corralling of information as it was unclear weather information should be free or private.

When certain rules and regulations were discussed to alleviate concerns about ownership there was no clear consensus. Thus conceptual muddle existed. File sharing technology, being a malleable technology, brought with it a policy vacuum and conceptual muddle about how to instill policies. There is still a lot of muddle concerning this area of cyber technology today.

To demonstrate just how much confusion existed with the free exchange of data over the internet just look at this ruling outcome between MGM Studios inc. v. Grokster, Ltd case in which conceptual muddle existed about file sharing. MGM sought to block the free exchange of copyrighted material through a Peer to Peer program owned by Grokster. The supreme court of the United States eventually ruled against Grokster and determined that the proprietary sharing of copyrighted material over the Internet was illegal. The basis of this decision was a division of two principles; “...the need to “protect new technologies” (Tavani

2007) (such as p2P networks) and the need to provide "remedies against copyright infringement" (Tavani 2007).

## SOA - A Cyber Technology

SOA has been defined, explained, and illustrated in many publications but it is also an excellent example of Moor's above three concepts, logical malleability, policy vacuum, and conceptual muddle.

When SOA entered the mainstream, when it began to garner attention because of its vision for business computing, it was a new cyber technology in the process of evolving. It took on various forms and enlisted many standards and technologies which eventually realized it as part of the new Web 2.0. This logical malleability of the early SOA, its ability to change and become a newer, more feature-ridden technology, brought with it a policy vacuum and conceptual muddle.

It went through many changes thus becoming logically malleable. It saw no policies to guide its use and development initially, thus a policy vacuum existed. And it saw conceptual muddle as new policies tried to intervene.

For instance SOA didn't start out as the new vision for businesses. It was first conceptualized in the delivering of shrink wrap software over the Internet in the 1990s. A crude form but nevertheless service oriented architecture. This type of software was developed de-centrally by proprietary organizations and then delivered to hosts that ran the software for a fee. The problem was the de-centralized nature of the software. This meant it lacked characteristics of a centralized service, as in SOA.

SaaS then garnered attention as services became available centrally, and with more enterprise wide characteristics. It wasn't until later however that SOA, with all its principles of design, began to stir up excitement.

What this evolution of SOA means is that SOA was a logically malleable cyber technology coming into its own that was consequently absent of policy. This policy vacuum brought with it much discontent as to how SOA should evolve so a conceptual muddle existed.

## SOA and Privacy

With this view of SOA as a cyber technology we can, from an ethical perspective, mention SOA and privacy. Privacy of information that travels across the internet is a must. Privacy of information exchanged over public Web service architecture is especially vital. Securing data, especially personal data, is a type of privacy referred to as informational privacy, defined below. But what is privacy?

One view of privacy is that it has both intrinsic value and instrumental value meaning it can provide worth for its own sake, for example happiness, or can be a means to some end, such as security. ".... though privacy is instrumental, it is not merely instrumental. ....unlike most instrumental values that are simply one means among others, privacy is also essential, necessary to achieve important human ends, such as trust and friendship" (Tavani 2007).

Moor's view of privacy, from a different perspective, assumes that privacy is not an intrinsic value but a means to some other end- instrumental. "Privacy is the articulation or "expression" of the "core value" security" (Moor 2004). Security within software systems is "The protection afforded to an automated information system in order to obtain the applicable objectives of preserving the integrity, availability, and confidentiality of information resources" (Taylor 2010). Of course security is a vital aspect of an SOA. Understanding that privacy is a pre-cursor to security, an instrumental view, only emphasizes the need for privacy in cyberspace.

As a Web service sits in the cloud (the Internet) exchanging information it can be subject to invasion of privacy as information (data) in the SOAP message becomes available to third parties unbeknownst to the requester or provider. The collection of information obtained from one company can be used by other companies for proprietary reasons. The information can be used in Data Merging, Data Matching, and Data Mining to customize third party selling practices, again without the knowledge of the user.

The ethical issue of privacy and cyber technology is a new phenomenon and coincides with the two other views of privacy, accessibility privacy and decisional privacy. Accessibility privacy is "... freedom from unwarranted intrusion" (Tavani 2007) on one's person or possessions. Decisional privacy is the "... freedom from interference in one's personal choices, plans, and decisions" (Tavani 2007).

As mentioned with cyber technology, including SOAs, we are concerned with the third type of privacy, informational privacy. This is due to four factors, listed below, that differentiate cyber technology from traditional technologies in terms of the need for private data. (Tavani 2007)

1. The amount of personal information that can be gathered
2. The speed at which personal information can be transmitted
3. The duration of time that the information can be retained
4. The kind of information that can be transmitted

Informational privacy can be defined then as ".... control over the flow of one's personal information..." (Tavani 2007), and is what we are concerned with in an SOA as we try to clear up conceptual muddles and establish policies.

But is the privacy of data over the internet, a theme that reverberates throughout cyberspace, what is best for the future of the Internet? As already mentioned the internet in its early years saw information as free and unrestricted. There were no rules to constrain the amount of free information one could accumulate. Even today the internet is considered a repository of publicly available free information.

If we now, comparatively, look at society in general and consider the way in which we are all allowed to gather in a public place, say a park, and freely exchange information we get an idea behind the information available over the internet. Indeed much of the public's understanding of the internet is that of a free and unique invention with information available upon request.

Thus the internet can be considered a public place, or intellectual commons area, in which free information exchange unrestricted by unforeseen forces is available as a free commodity. It becomes a virtual public arena in which information is globally available.

This concept of free information over the internet clashes with the rise of e-commerce which establishes data as one's own possession. With the rise of new and more sophisticated technologies we began to realize that the internet can be used as a means to profit. This profiting over the internet through technological means comes with it a need to privatize an organization's data.

Within an SOA the flow of data is the primary means of satisfying enterprise needs. Should this data be exposed to the intellectual commons area (the Internet) freely or should data be controlled more and more by policies and regulations, if not further exposed for profit as it increasingly is.

We looked at two concepts as it relates to SOA and Web services in this chapter, what constitutes a cyber technology and why privacy is vital within the SOA. The two

concepts are only related in that SOA is a cyber technology and that cyber technology, or the data it comes in contact with, must be kept private.

# Chapter Four: Project Related Considerations

## Software Architecture

As in SOA, Software architecture, the early principle design decisions made about a system, can be compared to the architecture of building a house. There are many processes and activities in building various structures for people to reside in but it is the building's architecture, or blueprint, that instructs and serves as a guide for all of the stakeholders, much more than just processes and activities. The processes and activities only serve to reinforce the architecture. The heart of a project is the architecture.

And so it is with architecting software, except that software is malleable and can exist in many forms, but for our discussion building structures or software is analogous. All Software has architecture to guide its design and development. It serves as the foundation that supports, but is also fed by, each phase of the SDLC. The architecture of a software system is not a phase or process however.

Software architecture is the view of a system from all aspects of its design and development. It provides to all stakeholders of a system, from the business community to the implementers, an abstract view of a system's components and their relationships. It helps stakeholders comprehend a system in which they can then use to proceed further in their activities. It is the core of a system's design and development used for guidance and comprehension while aiding in feedback processes as well.

Furthermore each phase of the SDLC, requirements analysis to maintenance, builds upon the architecture as the project moves forward, making the architecture stronger and more instructive for the next phase. Reciprocally, the architecture guides each phase.

Specifically, each phase of the SDLC maps to the prescriptive architecture (early design decisions) to keep the project on track.

Early design decisions, representing the architecture, are known as the prescriptive architecture of a system. It says what the system ought to be. The descriptive architecture is the final outcome or what is realized in the final solution. It is the goal of the development team to adhere to the prescriptive architecture as each phase progresses. That way you are adhering to the initial requirements or decisions made about a system and the customer's request.

Mapping phase activities to the architecture as discussed is part of what's referred to as architecture-centric development. Each early design decision made about a system should show in the result from some activity of that SDLC phase. For instance, if the early design decision or architecture states a system should take into account security, as would be the case in an SOA, then within the implementation phase each class interface would define security into their processes. This mapping of prescriptive architecture to each phase within the SDLC keeps the project on track.

## The SOA Architecture

. Distributed computing up until now has been only assumed in describing the SOA and Web services. Distributed computing is a result of wanting to ease the performance issues and complexity associated with the early years of computing where a software program ran entirely within the confines of a single computer.

In a distributed environment so called "tiers" represent the layers of a system. In a typical distributed system the presentation layer, business logic layer, and data layer are distributed over different computers on a network. The business logic layer interacts with the

user through the presentation layer to display information, usually as a result of manipulating data in the data layer.

When introducing a service interface into this distributed environment (see figure 4), the service interface, or the business logic representing some enterprise wide need, is exposed to the presentation layer and the business layer. The business layer might modify this service behavior before it renders the output to the presentation layer. Notice the new integration layer a service introduces.

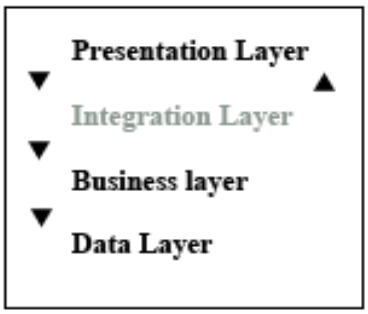


*Figure 4.* Distributed application environment

An SOA normally is a distributed, n-tier system, where the service because of its unique location adds an integration tier. This integration layer is where the service endpoints or service logic is exposed to consumers.

Software architecture, like architecting a building, is something that starts out as a few design principles but grows and supports development as the SDLC moves forward. Architecture, including the architecture of service orientation, provides the stakeholders of a particular project, all those involved with the project design and development, a clear picture of their roles, responsibilities, and activities. In a distributed environment a service within an SOA adds a separate independent layer to interact with the software.

## Managing the People and Technology

As Information technology goes it can become quite technical. Establishing an SOA in an enterprise is one such example. The teams enlisted in developing an SOA for an organization are not only everyday stakeholders with certain behavioral characteristics that reward them as they add input to a project.

At the heart of technology development are the knowledge workers who are rewarded, not by behavioral actions, but by characteristics such as thought and creativity. They are sometimes referred to as Geeks but their knowledge and skills show in their excellent work. They are rewarded by creative thoughts and actions and frown upon power and control that reward behavior instead of thought.

Supporting this type of creativity furthermore is an inherent quality we all have as human beings, trust. With trust a collaborative working environment can exist, and the knowledge workers can create to their hearts content. Communication becomes fluid as teams relate ideas without reservation.

This collaboration and communication are then two key supporting qualities of a development environment. Without trust the communication channels and trustworthy give and go that affect those working directly on the project cannot exist. Without trust neither can team members collaborate to exchange opinions and ideas. But what builds trust?

Communication, Performance, and Integrity are three such key ingredients which establish trust that in turn, enables a collaborative environment to exist.

- Communication builds trust in that one is trustworthy if a two directional information exchange can occur.

- Performance builds trust in that ones abilities are exposed as a task is completed which builds a sense of accomplishment, leading to a feeling of trust.
- Integrity adds confidence which aids in feeling trustworthy.

Thus trust is sought after within software design and development teams to enable collaboration and communication between members. This collaboration and communication, based on trust, feeds the knowledge worker in creating by thought and not by behavior, a characteristic of software development.

## The Project Methodologies

Software development is not an easy process. To bring software projects from inception to deployment requires a discipline that guides its design and development. There are numerous methodologies or techniques to choose from when trying to introduce a process that does this. Extreme Programming (XP), one of these techniques, is an agile method for software development.

Agile means “Marked by ready ability to move with quick easy grace. Having a quick resourceful and adaptable character” (Collegiate Dictionary). When it comes to Agility within the software world we can refer to development processes that are “more responsive to customer needs (agile)” (Wikipedia). These processes aid in the development of a software system.

A well known approach for developing software is known as the Agile methods. Agile methods are marked by speed of product development and the ability to adapt quickly to change. Agile methods use people and feedback, collaboration and ideas instead of documentation for completing a task. Taking the simple path or most direct path when developing the product is a quality of EXtreme Programming (XP) an Agile method.

XP is different from other methodologies in that it guides the SDLC by using a people-centric approach that delivers iterations fast with little documentation, little requirements gathering, little designing of the system, and not much testing. XP uses feedback over documentation, is highly capable of addressing changes in requirements late into the project, and can produce a system fast. The difference in XP from other methodologies for software development is its flexibility in the phases of the SDLC.

XP is able to deliver iterations that are readily changeable as the customer's requirements are changing. The XP process is malleable. The phases are not written in stone with XP. The idea is to produce a system fast and with more flexibility using collaboration and feedback amongst developers and customer. Thus the phases are flexible as XP uses slimmed down requirements before moving on to coding and deployment.

## The Data Layer

Early computing saw static Web pages delivered as cutting edge technology. With the rise of technologies to describe the data contained within a Web page (images, text, etc.), namely the Data Definition Language (DDL), Web page data became dynamic. The data now had meaning, being described by its DDL. But what is data?

Data are stored representations of objects or events that have meaning in the user's environment. Data, arranged in a certain way as to increase the user's knowledge, is called information. Data that flows through today's ubiquitous distributed software systems require relevant data and secure data structures. It requires a realization that an organization's data is its most valuable asset and that this asset should remain up to date and secure.

An organization, in order to produce a competitive advantage, must use its data and subsequent information in a way that meets goals and objectives. This relationship between data and organizational goals is gaining widespread consensus throughout today's online

business environment. It has become clear that data, used throughout the organization, are the element(s) that most importantly support the organization's goals.

As data has become the deciding factor in an organization's success it becomes necessary then to manage this data through a data management program. After all who wants data that is out of date, duplicated, or irrelevant? Within an organization I would therefore –

- Set up an overall Data Management Program to oversee the activities needed to keep the data managed.
- The use of data within an organization requires a Database Management System (DBMS).
- A committee to oversee the relevance of data (keeping the data current and useful to the organization) would be set up.
- Stemming from this committee would be managers that oversee the data, making sure the data is independent, that it reduces data dependencies, improves data sharing, improves data quality, and other data characteristics.
- Finally it would be appropriate for SOA initiatives to have an SOA governance board to, amongst other things, see to the alignment of changing business processes within IT application processes.

## The Database (DB)

A DB is simply a structured collection of logically related data. Within the software world the DB stores and allows manipulation (through a Database Management System) of data to and from computer programs, mostly for use in organizations. "You can think of a database as an electronic filing system" (Webopedia n.d.).

A DB is made up of fields, records, and files. A field is a piece of information, like a name category or address category in a phonebook. A record is one set of fields, such as John, Smith, Concord, Mass., Phone# etc. A file is a collection of records. The most common type of DB used for online activity is the Relational database.

In a relational DB, like one that is used in this project, there are “entities” stemming from organizational needs that become “tables” that have relations with other tables through a unique field within a record. In a relational DB each record has a Primary Key (PK) field that makes it unique. In the record above the Phone field value would uniquely identify John Smith. Usually an ID number is used as the PK however as it is more unique.

This relationship between tables supports the use of data from all sectors of the organization, or as many as needed for a healthy solution to materialize. Data from table one, representing some enterprise wide need can use data from table two which can then interact with table three before presenting the desired results to the user. Each table represents one sector or need of the organization. This wide use of data within an organization is another example of an enterprise’s primary asset- its data.

## Chapter Five: Implementing SOA with Web services

This paper has covered many aspects of the SOA and Web services but has yet to demonstrate exactly how one would actually establish a presence on the Internet in the form of a Web service.

Within the Java programming language, used to create the two services for this project, there are two APIs (Application Programming Interfaces) that facilitate creation of Web services. A JAX-WS API Web service requires the service contract (discussed on Page 11) in the form of the WSDL file as described in the section WSDL in chapter two. It is a SOAP based Web services meaning it transports data between services within a SOAP envelope. It was used in creating both Web services for this project. It is the most common type of service used in an SOA.

The second API is called JAX-RS API for a REST (REpresentational State Transfer) based Web service. The Service architectures of JAX-WS and JAX-RS differ in that a RESTful Web service does not require a contract between consumer and provider. It transports and manipulates data using HTTP methods.

The request and response are sent over HTTP in the URL using the GET, PUT, POST, DELETE, and HEAD methods of the HTTP protocol, mostly GET and PUT. JAX-RS is an easier to use Web service API making it popular to many developers. However it doesn't offer the capabilities of the SOAP based service.

Interestingly REST stands for REpresentational State Transfer. It is the architecture of the World Wide Web. To understand and to describe the architecture of a RESTful Web service and to understand the architecture of the World Wide Web, look at the words in the title; Representational, State, and Transfer.

The Internet can be thought of as a collection of resources, Web pages for instance, which can be retrieved through their URLs and rendered to a user's browser. If one could imagine the hyperlink to the resource as 'representing' the resource you have the REpresentational aspect. The hyperlink represents a resource on the Web, the Web page.

State refers to the currently displayed resource as it is displayed in the user's browser. After clicking the representation (Link) the current state or Web page is transferred into the desired resource. The Web page is said to transfer state.

## The NetBeans IDE

It is very simple to create a Web service within the Netbeans Integrated Development Environment (IDE). Using the Java language, in this case, and with a few clicks of the mouse you can have a Web service running over cyberspace in no time. But what is made easy in the NetBeans environment only hides the complexity that Web services introduce as one could attest from reading the Web services section in chapter two. Luckily the NetBeans IDE automates most of the complexities for you.

Specifically for this project was created a SOAP based Web service in NetBeans named MailingList. Another SOAP based Web service named CustomerPurchaseOrder created also in NetBeans was developed in a previous project but is highly reusable.

The MailingList Web service accepts three parameters from a client and writes that information to a file on the server hard drive. It then offers to the client that the record be inserted into a database permanently by referencing the CustomerPurchaseOrder Web service that performs the insert.

Lastly I create a client desktop application, a consumer of the Web services, which runs the services. This desktop application was developed in short time with little effort but is

a major part of this project. It serves as a demonstration of what was discussed under the section Web services.

Each Web service provides to applications such as this desktop application the wsdl file. The client program creates a Proxy object from this file that, one, references the Web service, and two, is then usable to interact with the service. Consequently my desktop application, which happens to be named SubscribeClient, has access to two Web services.

# Emerging Trends

## Web Oriented Architecture (WOA)

One rising trend that meets the need of the enterprise in a changing market is WOA. WOA can be considered a light version of SOA on the web. It is an extension of SOA. In an SOA organizations present their services as network based processes that are centralized allowing company data to pass.

In WOA this architecture aligns more with the demands of a changing public as it is located on the Web. The service within SOA is replaced with the resource in WOA, a representation of a company's data and information. This Web resource, which contains its own unique Uniform Resource Identifier (URI), is used in a WOA by the network protocol's methods- typically HTTP and methods GET, POST, PUT, and DELETE whenever a request is made by a consumer of the resource. Below is a table of qualities that helps define WOA.

*What is WOA? The Basic Tenets-*

- While data and information are the central points of emphasis on a network for businesses today, WOA represents this data in the form of a resource, accessible through the network protocol's methods via a unique Uniform Resource Identifier (URI).
- Every resource can be located in this manner globally through this URI.
- The accessibility and manipulation of the resource are handled through the network's protocol, specifically its methods. In a WOA this is typically HTTP and methods GET, PUT, POST, and DELETE. The actual manipulation of the resource by these methods in this way follows the REST architecture described in chapter five. It is a

simpler design approach and is gaining widespread acceptance by organizations such as Amazon, EBay, Microsoft, and others.

- It is typically a Web based design where only network components (browsers and servers) are capable of manipulating the resource.
- Access and manipulation of the resources requires no more than local knowledge of the network.
- Components only, interact with the resource. They must understand the representations of the resource as well as the state the representations transfer to.
- In SOAP based Web services the wsdl file represents the explicit contract between services. In a WOA this contract is implicit in the resources representations of other resources.
- WOA resources contain embedded URIs that build a larger network of granular representative state (i.e. order resources contain URLs to inventory resources).
- The design principles described in chapter two are of equal importance in WOA as in its precursor SOA.

# Conclusion

It is the hope of the author that the reader has enjoyed reading this paper and understands the SOA and Web services more than he did before beginning. The SOA and Web services platform promises to be a solid investment for organizations for the foreseeable future and beyond.

Service orientation arose out of the Object Oriented paradigm of software design and development with an emphasis on reusable, independent, customizable services that meet some need of the enterprise. By combining traditional object oriented system architectures with the service architecture concepts discussed in this paper, and by further instilling the design principles discussed in chapter two into the services, you have a system that goes beyond meeting organizational needs. You have a new vision for your organization.

The vision that has emerged in the online business world is in the form of Web services. Web services are services that communicate via HTTP over the internet. They also send and receive XML formatted data.

However building Web services for your organization will not solidify the vision SOA offers. Many organizations build Web services and think they will see the new light. Web services are one aspect of SOA and are only one type of service. An SOA that meets the needs of the organization to produce a competitive advantage adheres to certain design principles that will position the enterprise as a leader in the world of IT and automated resource availability.

# References

Blackwell publishing website (n.d.) *what is Computer Ethics* Retrieved October 2010 from http://www.blackwellpublishing.com/content/BPL_Images/Content_store/Sample_chapter/9781855548442/CEAC01.pdf

Erl T. (2009) *Service Oriented Architecture, a field guide to integrating XML and Web services* Chapters 3

Erl T. (2009) *Service Oriented Architecture, an introduction to the service-oriented design principles* Retrieved November 2010 from http://soaprinciples.com/default.php

Merriam-Webster's Collegiate Dictionary (2005) *Merriam-Webster's Collegiate Dictionary* Eleventh edition

Russell K. (2006) *SOA* Retrieved October 2010 from http://web.ebscohost.com.library.capella.edu/ehost/detail?vid=3&hid=109&sid=90b61c7f-5a3d-4294-ade1-13e7d6fb2cd0%40sessionmgr104&bdata=JnNpdGU9ZWhvc3QtbGl2ZSZzY29wZT1zaXRl#db=bth&AN=12484689

Taylor R. N., Medvidovic N., Dashofy E. M. (2010) Software Architecture- foundations theory and practice, Chapter 11 p. 433

# Appendix A

**Member Table Data Definition Language (DDL):**

```
  CREATE TABLE "ML1088809"."MEMBERS"
   (                                    "FNAME" VARCHAR2(20 BYTE),
                                        "LNAME" VARCHAR2(20 BYTE),
                                        "EMAIL" VARCHAR2(20 BYTE) NOT NULL
ENABLE,
                                         CONSTRAINT "MEMBERS_PK" PRIMARY KEY
("EMAIL")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "ML1088809"  ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "ML1088809" ;
```

# Appendix B

## The Service Design (Diagrams)

### Enterprise Application Diagram (The Web service application environment)

The diagram below depicts the structure of my project solution. The purpose of the diagram is to show the de-centralized nature of the Web services from traditional OO system development environments.

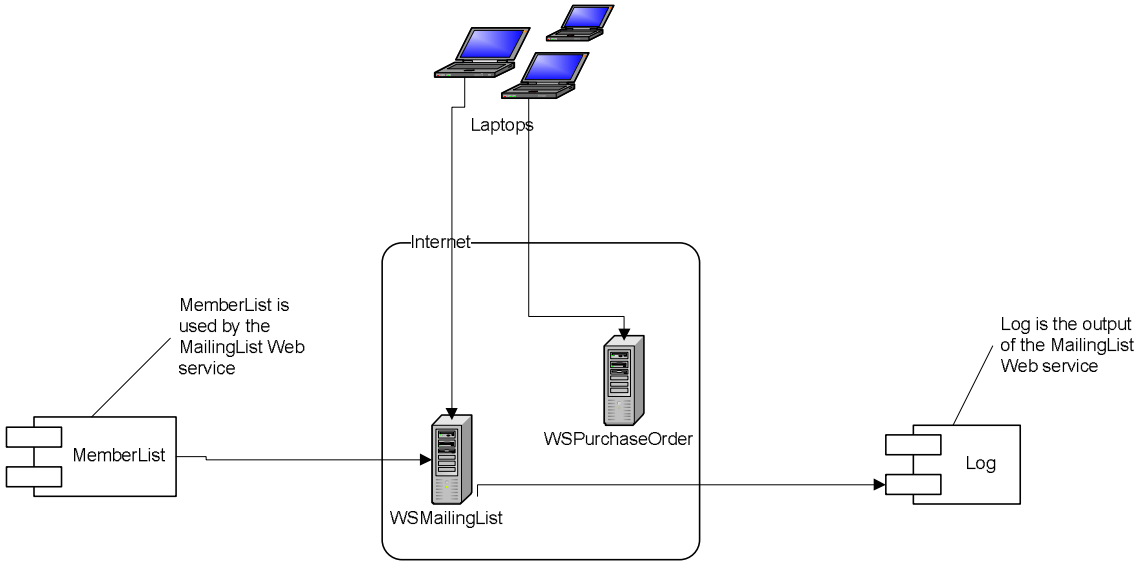


*Figure 5.* Enterprise Application Diagram- Each Web service sits on a network waiting for requests and delivering responses

## The Use Case Diagram

### Scope Statement

A user or another service can subscribe to a mailing list for future email discounts and promotions. He has the option of inserting the subscription information into a database via the WSPurchaseOrder Web service.

A *user* or *service* **subscribes** to a mailing list. A *user* or *service* **inserts** a record into the database.

**NOUNS**: user, service, mailing list, record

**VERBS**: subscribes, inserts

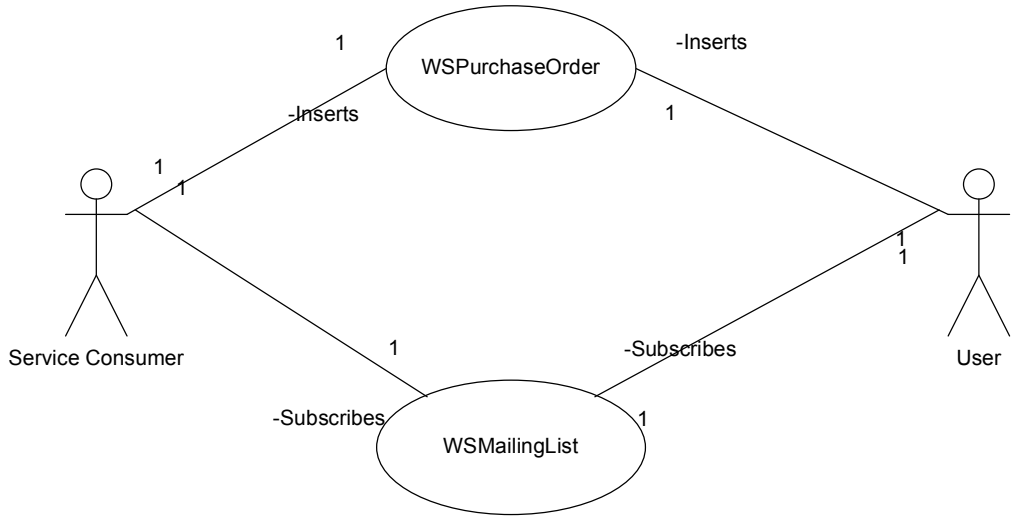


*Figure 6.* Use Case Diagram- Consumers consuming two Web services

**Class Diagram**

Each class in an Object Oriented (OO) system represents the blueprint of an object. It defines an object's state and behavior at runtime. It is what an object is and does. In the simple class diagram below, a consumer of the MailingList service, either a user or other service provides on a one to one basis, parameters to the MailingList Web service. The Web service is dependent on the MemberList component class. It also has an association with the Log component class. The user or service then has an option to insert the information into an Oracle 11g database using Web service WSPurchaseOrder.

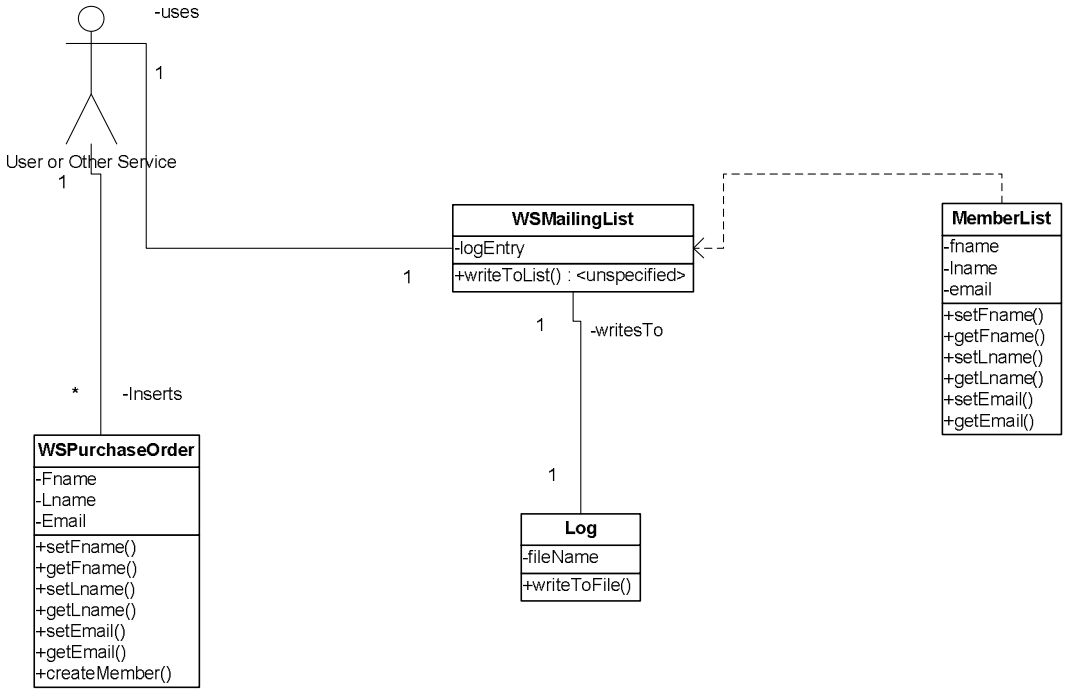


*Figure 7.* Class Diagram- A user writes to a file his information via method writeToList(). He can also store the info in a database via Web service WSPurchaseOrder's createMember() method

**Sequence Diagram**

Below is a Sequence diagram depicting the flow of the application components.

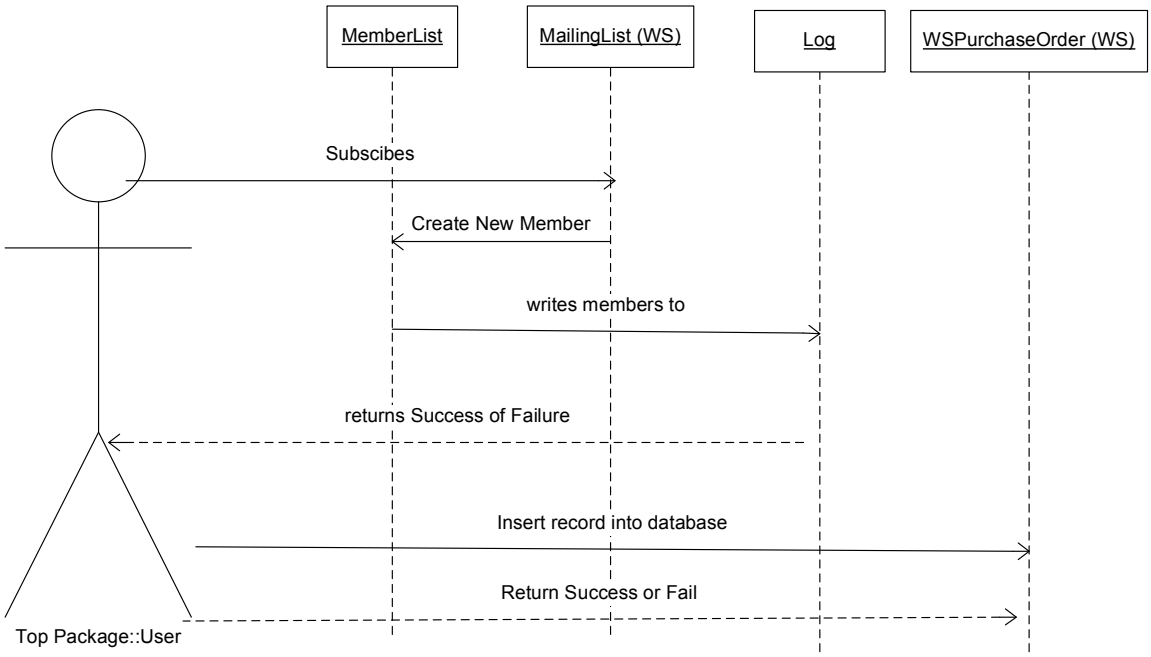


*Figure 8.* Sequence Diagram- User writes to a file, a new member, with an option to permanently store the information in a database.

# Appendix C

## The Service Implementation

### The MailingList WSDL file:

http://vle6.capella.edu:9038/MailingList3/MailingListService?wsdl

### The CustomerPurchaseOrder WSDL file:

http://vle6.capella.edu:9038/WSPurchaseOrder/CustomerPurchaseOrderService?wsdl

### The Web services and component classes source code:

http://www.superiorwebdesign.org/CapellaPortfolio/Assignments/TS5950/TS5950.htm