2010

Sensitivity in Service Design for the Development of SOA Based Systems

Mohit Sud
University of Windsor

SENSITIVITY IN SERVICE DESIGN FOR THE DEVELOPMENT OF SOA BASED SYSTEMS

by
Mohit Sud

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2010

SENSITIVITY IN SERVICE DESIGN FOR THE DEVELOPMENT OF SOA BASED SYSTEMS

by

Mohit Sud

APPROVED BY:

Dr. Ahmed Azab, External reader
Department of Industrial and Manufacturing Systems Engineering

---

Dr. Scott Goodwin, Internal reader
School of Computer Science

Dr. Xiaobu Yuan, Advisor
School of Computer Science

Dr. Xiao Jun Chen, Chair of Defense
School of Computer Science

16 Sept. 2010

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Service Oriented Architecture has proven itself to be a beneficial approach to software development. One of the most identifiable challenges of the SOA model is performance evaluation and service selection. To ensure the continuing success of SOA, service requestors require a technique to evaluate existing services to identify and select the best service available for their needs. Furthermore, service providers require a similar method to evaluate the services they create to ensure the consistency, and performance of their services. A technique of sensitivity analysis addresses these concerns by evaluating the effects of factor variation on system performance in a quantitative manner. An algorithm is produced to identify which factors are sensitive to factor variation in a software service. An experiment is performed to demonstrate the effects of sensitivity analysis as it applies to SOA systems. The experiment successfully shows that sensitivity analysis is a successful approach of evaluating a services performance and resolving issues surrounding service selection.

To my dear parents for the encouragement and support of a lifetime.

Acknowledgements

It is near impossible to overstate my gratitude and appreciation I have to my M.Sc supervisor Dr. Xiaobu Yuan. Through his great efforts to explain things clearly and simply, he helped pave the road for me to walk. Throughout my thesis-writing period, Dr. Yuan has provided me with encouragement, advice, excellent tutelage, and many great ideas. Thank you Dr. Yuan.

I wish to thank my Internal and External Reader, Dr. Goodwin and Dr. Azab, for their excellent suggestions that helped to improve the experiment and overall quality of the thesis.

I wish to thank my entire family and friends for providing a loving environment for me to work. This success could not have been achieved without all of your support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope of Service Oriented Architecture (SOA)

The introduction of the Service Oriented Architecture (SOA) software model has proven itself to be a beneficial approach to software development. Its growth and popularity are constantly spreading throughout the industry, prompting businesses to change their current methods to incorporate this new development technique. SOA involves separating logic into small components that can be reused across multiple applications and multiple projects. Although the modern approach of SOA includes many benefits that could revolutionize the software industry, like any new technique, SOA includes limitations and drawbacks that may affect its acceptance. One of the most identifiable challenges of the SOA model is software performance.

Software performance is critical to the success of any software model and is an area that the SOA community is actively working on investigating. Key areas related to performance are performance evaluation and service selection[10]. One of the benefits of SOA is that multiple services may exist that aim to achieve the same objective. This creates

competition among services, thus reducing cost and providing service requestors with options when selecting a service. With the growing number of available services on the Web, service selection has become a key problem in the SOA research area. In order to ensure the continuing success of SOA, service requestors require a technique to evaluate existing services to identify and select the best service available for their needs. Furthermore, service providers require a similar method to evaluate the services they create to ensure the consistency, and optimality of their software services as a way to increase the desirability of their services.

1.2 Sensitivity Analysis

Typically, the SOA based applications include a set of components. Each component may have several parameters commonly referred to as input variables, or factors. Factors are input variables to a software system whose changes in value affect the performance of the system. Sensitivity analysis is the process of evaluating the performance of a software system with respect to their sensitivity to factor variations, and to identify the effects of factor variations on a systems performance. Through applying sensitivity analysis to SOA, this will help resolve many of the issues related to performance from the view point of both the service requestors and service providers. Service requestors will have a guideline to identify and select the most optimal service available for their application needs. Also, service providers can evaluate and enhance their services to make them more desirable.

1.3 Structure of Document

This document explores the rapidly growing and popular subject of SOA. A literature review is performed in regards to SOA. Some of the issues and drawbacks associated with SOA are identified. The drawbacks pertain to the development of services and service selection in regards to performance based upon sensitivity to factor variation. A solution is proposed and an algorithm is created to perform sensitivity analysis upon a service to determine the effects of factor variation on performance. An experiment has been designed that will evaluate the application of applying sensitivity analysis to SOA to identify and select services that yield better performance. The experiment also demonstrates how service developers can use this information to improve their services and make them more desirable.

Chapter 2

Review of Literature

2.1 Overview

Service Oriented Architecture (SOA) is a software model in which automation logic is separated into smaller, distinct units of logic. Together, these units compose a larger piece of business automation logic. Individually, these units may be distributed. Each unit of logic is loosely coupled and may be reused across multiple applications, as well as aide in creating several different project objectives [4].

The SOA software model contains three basic components; Service Provider, Service Registry, and Service Requestor, as seen in Fig 2.1 [17].

- *Service provider* - responsible for creating and publishing a service to a registry and makes it available through the Internet.
- *Service requestor* - performs service discovery operations on the service registry to find the needed service, then accesses that service.
- *Service registry* - aides service providers and requestors to find each other by acting

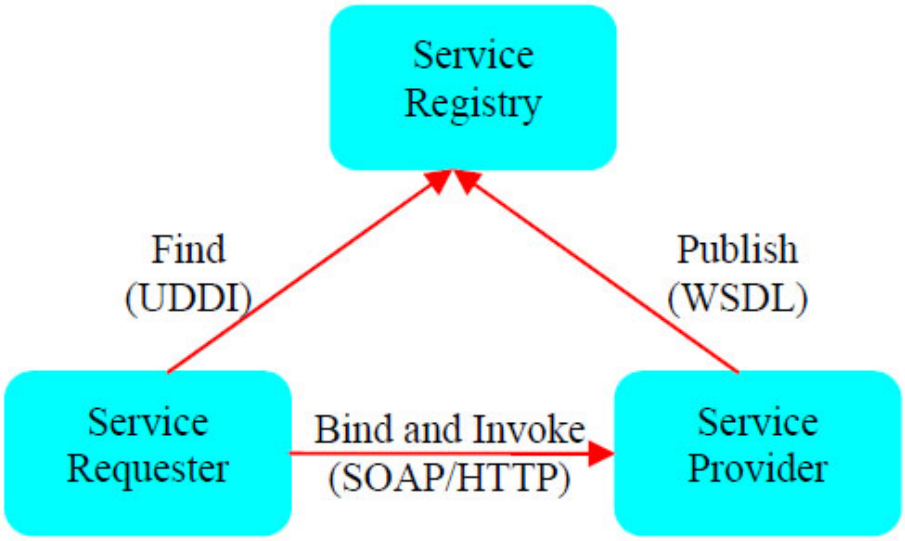

Figure 2.1: The Basic Components of SOA

as the registry of services.

Before a unit of logic can be considered part of the SOA model, it must first conform to a basic set of principles and standards that will allow them to grow independently while maintaining its ability to work conjointly with other units of logic. Each unit of logic is composed of three core components; services, descriptions, and messages [5].

- *Services* - Services encapsulate a single, or multiple units of logic. It allows developers to utilize the unit of logic from within their own applications. Services are essentially the communication structure between applications and the unit of logic.

- *Descriptions* - Service description must at the very least specify the name and location of a service, as well as data-exchange requirements. Typically, this information is included in a WSDL file (Web Services Description Language).

- *Messages* - Messages are passed to or received from a service as an 'independent unit of communication'. The messages are autonomous and contain enough intelligence to self govern their parts of the processing logic. This is because once a message is sent, the service loses control of what may happen to it.

2.2 Application Scope

The uses and application of SOA extends far beyond its typical uses today, however as it is an emerging technology, we are limited in its current implementation. The most current and effective implementation of the SOA software model may be seen through the use of web services. SOA is essentially a software design principle, whereas web services is a SOA based interface definition standard.

The popularity of SOA is driven by the momentum created by web services. Through the use of SOA, web services have fundamentally shifted the way applications are built, and involve businesses rethinking the role applications play in their enterprise [15].

As SOA is still in its early stages, the future use of the SOA software model has yet to be conceived and is constantly changing. There are multiple ideas (or dreams) about the practical and future use of SOA. One of the most noteworthy however is how SOA may aide in implementing the semantic web.

The semantic web is a concept that represents the idealization of having a vast amount of information linked in such a way that it is easily understand and interpreted by machines on a global scale. It can be perceived as an efficient way of representing data on the web, or as a globally linked database. Although the notion of the semantic web is revolutionary and can potentially change the way we use the World Wide Web, the development of the semantic web is still in its early stages.

The logic behind the semantic web is that valuable information is generally hidden away in HTML pages and may be difficult to find and use on a large scale. This is because there is no global system for publishing data that may be easily processed by everyone. The semantic web hopes to change that and make publishing data easier and in a more reusable form. This will create a ripple effect that more people will want to publish their data, because more people can access it easier, thus, reaching a broader audience.

2.3 Web Services

2.3.1 Overview

Web services are the most current implementation of the SOA software model. It utilizes SOA's design principles to create reusable software components that use a standardized messaging system, which is built within the scope of the internet. Through web services, different kinds of platforms and systems are able to communicate with each other in a common language, without the need for custom interfaces or wrappers. Web services include a standardized method for supporting machine-to-machine interaction over a network. Typically, a web service is a loose coupled unit of logic (or application) that includes an API that may be accessed over a network (typically the internet). SOA and web services provide complete transparency of the application, allowing developers to utilize the features of another application without needing to know the underlying implementation details behind it.

Web services are designed to support machine to machine communication over a given network. This is achieved through the adherence of XML based standards including SOAP, WSDL, and UDDI. Combined, these standards provide a method for locating, publishing,

and using web services. The web service stack is a combination of these standards and protocols that support communication over a network. Fig 2.1 illustrates the web service stack.

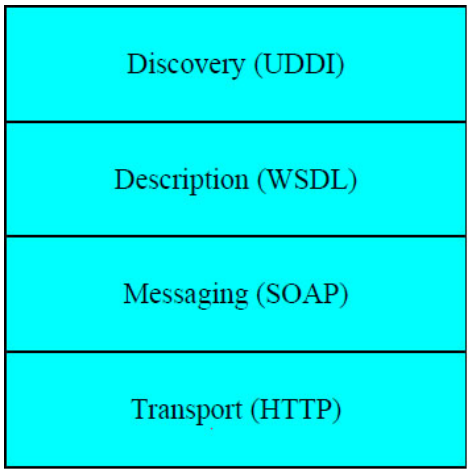

Figure 2.2: Four Layer Model of the Web Service Stack

- *Transport Protocol* - has the responsibility of passing messages between network protocols. Typically, HTTP is used; however web services are not restricted to any specific transport protocol.

- *Messaging Protocol* - has the responsibility of encoding messages in XML to ensure they are understood by the recipient. SOAP (Simple Object Access Protocol) is a common format for exchanging web service data over HTTP.

- *Description Protocol* - used to describe the interface for a specific web service. Typically, a WSDL (Web Service Definition Language) file is used to describe this information.

- *Discovery Protocol* - centralizes services into a common registry. UDDI (Universal

Description, Discovery, and Interaction) is a specification used by service providers to advertise the existence of their services to requestors in a service registry.

2.3.2 Implementation Methods

There are many ways to implement the messaging portion of a web service. Not all implementation methods are suitable for all projects, but the most noteworthy and widely used are: SOAP, XML-RPC, and REST [7].

Simple Object Access Protocol (SOAP)

SOAP is the most widely used web service technologies available. It is primarily an XML based protocol, and is supported by the W3C (World Wide Web Consortium). SOAP is platform and language independent and is considered to be the most widely supported web service implementation because of it. Unlike other web service technologies, SOAP is not restricted to a specific transportation protocol such as HTTP. Although SOAP may be considered to be the most dominant web service technology due to its many features and fewer limitations, it is also considered to be more complex, and slower than other web service technologies.

Extensible Markup Language - Remote Procedure Call (XML-RPC)

XML-RPC is a protocol that allows software running on different operating systems, and different environments to make remote procedural calls over the internet. Its remote procedural calls utilize HTTP as the transport layer and XML as the encoding. XML-RPC is a simple, lightweight implementation method for SOA. It allows complex data structures to be transmitted, processed, and returned. XML-RPC supports rapid development while maintain clean design and simplicity.

Representational State Transfer (REST)

Rest is not a standard, a language, or a protocol but a term used to refer to an architectural technique that works with the existing technologies of the web and its protocols (e.g. HTTP and XML) and exploits them to achieve new and different functionality. REST is said to be simpler to use than SOAP (although not as powerful) since it does not require the need for both a client program and a server program. REST has a different design approach when compared to other web service technologies. Unlike SOAP and XML-RPC that use RPC (remote procedure calls), REST utilizes the aspects of a resource that defines its content types. The largest advantage of REST over other Web Service technologies is that it may utilizes many of the Web protocols and is not limited to HTTP. Through its use of many Web protocols, REST can fully incorporate caching into its architecture, thus improving the overall performance of the service [16].

2.4 Advantages of SOA

There are many advantages to implementing an SOA based application [13] [1]. These advantages include:

- An SOA based application may be accessed from anywhere through the use of the World Wide Web. There is complete location independence.
- SOA based applications are completely reusable. Loosely coupled services allows for units of logic to be reused in many different applications.
- An SOA based application is completely platform and language independent.
- SOA allows for the service and connectivity to other applications to be done dynamically.

- SOA provides a real time decision making environment.
- SOA allows developers to implement new tools and services without needing to know the underlying services implementation details.
- SOA provides a data bridge between otherwise incompatible technologies.
- SOA decreases development time through service reuse.

Chapter 3

Performance Evaluation of SOA

3.1 Overview

SOA is an emerging technology and like any rising star, there are noteworthy drawbacks and concerns pertaining to its implementation. The most widely discussed issues pertain to the performance of SOA based applications. This section will highlight some of the most discussed aspects with the performance of SOA based applications, including quality of service, memory implications, and service sensitivity.

3.2 Quality of Service

It is important to realize that client processing power has grown considerably over the years, however network speeds have remained steady. This leads to the question of how to ensure Quality of Service (QoS) on applications that are heavily dependent on services scattered across a network. Since it is unlikely that we will be able to improve the overall transfer speed of a given network, it is important for developers to optimize their applications in

order to meet their QoS requirements.

It is extremely difficult for an organization to determine their QoS requirement at the start of application development. Furthermore, one of the most severe issues is the scalability of the application during peak usage times. The poor performance is largely attributed to overheads of delivery, parsing, validation, serialization of the XML data, as well as 3rd party service integration.

There are a series of methods and techniques that exist that may aide in improving the quality and responsiveness of an application, as well as ensure only a minimal amount of messages and data are transferred [9]. These techniques include:

- Utilizing caching to reduce bi-directional data transfer over a given network.
- Implement client side validation to reduce data transfer
- Lazy load required data to improve the user's perceived responsiveness. Lazy loading is a technique that involves pre-fetching information from the server, even though the client has not requested it yet.
- Shift certain requirements and responsibilities over from the server to the client.
- Incorporates Virtualization to handle resource sharing dynamically across systems and platforms.

3.3 Memory

For majority of all SOA based application there are two major components, a server component and a client component. The server component is responsible for interacting with all of the clients, and processing any required unit of logic that may be required. Since the

server component may be responsible for potentially millions of transactions at any point in time, it is important to consider the memory footprint on the server as the number of clients grows. The number of clients may grow as the application gains popularity, as well as during peak usage times.

It is important to consider the memory implications on such an application. As the memory demand increases to a point where the server is unable to keep up and manage all of the requests and resources sufficiently, it may result in the server overloading, crashing, and software outages.

Various techniques have been developed to help cope with the memory footprint problem [9]. These solutions include:

- Running performance tests to identify and optimize transactions that may be resource intensive and responsible for application sluggishness
- Optimize, eliminate, or reduce the state information that may be recorded on the server. This may drastically reduce the memory footprint.
- Incorporate server farms that you may expand as the demand increases (a costly solution).
- Cache application data on the client and not on the server.

3.4 Service Sensitivity

One of the greatest benefits of SOA based systems is service reuse. A service provider can create a service and allow it to be used by any number of service requestors in any number of applications. However, not all services are unique. There may be countless services

available that aim to achieve the same objective. This results in services competing against each other to be selected for various application uses. Furthermore, service requestors may have unorthodox or various input value requirements for the services they wish to use. It is strongly important for service requestors to be able to identify and select the most optimal service available that suits their changes in input value needs. Through this assurance of service selection, service requestors can ensure the successful operation of their application. The process of service selection is an active performance related issue in the SOA model that has no common solution to aide in its resolution.

The process of evaluating the performance of the output of a software system based on changes to its input values is known as sensitivity analysis. The importance behind sensitivity analysis as it applies to an SOA based service is that it assesses the output a software services performance based upon changes to its input values, thus allowing service requestors to identify and select the most suitable service for their needs as well as allowing service providers to evaluate, and improve upon their existing services. The objective of this thesis is to address a performance issue in the SOA model that pertains to performance evaluation and service selection. This endeavour will benefit both service requestors and service providers, as well as continue the success growth of the SOA model.

Chapter 4

Sensitivity Analysis

4.1 Introduction

The investigation of how changes to the values of the parameters of a given model affect the result is known as sensitivity analysis. Through this procedure we can determine how 'sensitive' a model is to changes in the value of its parameters and to changes in the structure of the model.

Through demonstrating how the behaviour of a model responds to changes in its parameters values, sensitivity analysis is a useful tool in model construction and model evaluation. Through studying the uncertainties that may be associated with parameters in a model, sensitivity analysis aims to develop assurances that the model will perform accordingly for various input sizes [2].

To outline the need and importance behind sensitivity analysis, John Graham, a Risk Assessment professional from the University of Washington stated "Sensitivity analysis is particularly useful in narrowing the degree of uncertainty in the results" [18].

4.2 Advantages of Sensitivity Analysis

There are many benefits and advantages to sensitivity analysis. These benefits can be applied to various applications including SOA, risk analysis, software performance evaluation, etc. Some of the most noteworthy benefits to sensitivity analysis include [14]:

- Exploring the impact of varying input assumptions and scenarios
- Simplifying software models
- Investigating the robustness of the model predictions
- Provides quality assurance
- Identifies factors that mostly contribute to the output variability
- Identifies the optimal regions within the space of factors
- Identifies the effects of the interaction between factors

4.3 SOA and Sensitivity Analysis

SOA is a growing software model that is changing the way businesses develop their software systems. Its demand and popularity is increasing at a rapid pace. In order to ensure the continued growth and success of SOA, developers that are creating new services must ensure the quality of their services. As well, developers selecting a single service from multiple services available to use in their application must identify and select the most optimal service for their needs. Assurances are required to ensure that the services selected performance is maintained for various changes to its input values. These requirements are

not optional but are a necessary requirement for SOA to prevail and succeed in the future as a dominant software model.

These aims at addressing performance related issues in SOA are resolved through the application of sensitivity analysis as it applies to SOA. A methodology will be introduced to demonstrate how sensitivity analysis may be applied for the purpose of evaluating a services performance based on their sensitivity to factor variation. It is through these techniques that quality assurance can be provided, and service selection can be made after analyzing the results produced through sensitivity analysis.

4.4 Approaches to Sensitivity Analysis

There have been multiple approaches that are similar in nature to sensitivity analysis as it applies to various models, and are not restricted to the software domain. These approaches have not been applied to the SOA model as a means of effectively evaluating the performance of a service for the fulfillment of quality assurances and service selection in an effective manner.

Brute Force - Works only on small models that take a short amount of time to solve, change the initial data and solve the model again to see what results you'd get.

Classical Sensitivity Analysis - Applies to very large models that take a large amount of time to solve. The classical sensitivity method relies on the relationship between the initial table and any later table to quickly update the optimum solution when changes are made to the coefficients of the original table.

Computer Based Ranging - Simple information about how certain coefficients can change before the current optimum solution is fundamentally changed. John W. Chinneck, a professor at Carlton University, has done considerable research in regards to using sensitivity

analysis to identify which data has the most significant impact on results in a software system. He used a combination of linear programming and computer based ranging to establish this information.

Analysis of Variance (ANOVA) - The ANOVA approach is very common in statistics. It involves performing computations to determine if the mean of two given treatments are equivalent or not. Although this approach has proven successfully in evaluating various statistical models, its methods are best suited for two factor based sensitivity analysis [11].

Design of Experiment (DOE) - DOE experiments are not restricted to software systems and may be performed on a variety of things including; software systems, people, plants, animals, etc. DOE allows for observation and judgment on the significance to the output of input variables that may be acting alone, or in combination with one another. DOE may be considered to be Sensitivity Analysis earliest ancestry. Its approach has not been adapted to software service [8].

4.5 Previous Research

Sensitivity analysis is a practical optimization mechanic that has been used to establish confidence and performance optimization in many things including statistical research and software applications. There currently has not been a sufficient amount of research aimed in regards to applying sensitivity analysis to the success and improvement of SOA. The most identifiable research has been conducted at the University of Windsor with Dr. Xiaobu Yuan and a few former students [3].

Shangwei Duan, Tony Huang, and Dr. Xiaobu Yuan, University of Windsor, have demonstrated a technique that allows for two-factor based sensitivity analysis of SOA based services. There technique applies sensitivity analysis to SOA in an effort to evaluate and

predicate software quality, as well as to identify optimal configurations in software services with only two factors [19].

Chunjiao Ji and Dr. Xiaobu Yuan, University of Windsor, have outlined a methodology for multi factor based sensitivity analysis of SOA based systems. Their approach is used to identify which if any individual factors or joint factors have significant effects on the performance of a software system [20].

Chapter 5

Problem Statement

5.1 Statement of the Problem

Service Oriented Architecture (SOA) has gained an ever increasing popularity in both the academic and industrial communities in recent years. To ensure its continuing growth and success, concerns relating to its performance must be addressed. Two primary concerns surrounding SOA are performance evaluation to provide quality assurance, and service selection. This Master's thesis introduces a technique that offers quantized evaluation of service software based upon individual and interactive performance of services. In particular, an algorithm is developed to evaluate the performance of an SOA model with respect to changes in its service components for a specified range of parameters. This technique of sensitivity analysis provides a guideline for SOA developers to identify and select the most optimal software service, and for service providers to ensure the consistency, and optimality of the software services they aspire to create.

5.2 Purpose of the study

5.2.1 Overview

The approach to service oriented development involves having a set of loosely coupled components assembled together to create a new service with new functionality. With this technique, each component can be reused in a multitude of different projects with different objectives.

Several software services may be available that strive to achieve the same end objective as each other. Service requestors have a choice as to which services they choose to incorporate within their application. Although several software services may aim to achieve the same objective, they may produce different results for the same input parameters. Inaccuracies and errors in a software service may make it perform differently when compared to another software service with the same intent. With that in mind, each software service must provide quality assurances to remain as a competitive and desirable service.

With the introduction of sensitivity analysis as it applies to service oriented development, it provides a technique for developers to evaluate the performance of a software service with respect to its change in the input parameter values for its service components. Through demonstrating how the behaviour of a model responds to changes in its parameters values, sensitivity analysis is a useful tool in model construction and model evaluation. This plays a critical role at assisting service requestors with the selection of services and service providers of services.

5.2.2 Service Selection

For developers that are creating a SOA based application that is dependent on and is composed of several other services, developers require a technique to ensure that the services they select to be used in their application will perform accurately under the scope of changes that may occur to the input values of their SOA model.

With the introduction of sensitivity analysis to software service selection, developers can identify and select appropriate services as required for their application. With sensitivity analysis, developers can run performance tests to learn how any change in the input of a model will affect the output and whether that variation when compared to another service is greater, identifying the most optimal service to select.

5.2.3 Quality Assurance and Service Optimization

When a service provider is creating a software service that may be used in a wide scope of different applications, sensitivity analysis aims to develop assurances that the service model will perform accordingly for various input sizes. This enhances marketability for the service, as well as desirability for those who wish to use the service because they now have reassurance that the service will effectively meet their needs. In turn, this will improve the desirability of the service, thus increasing sales and usability.

5.3 My Contribution

It is evident that there are various performance related issues associated with the SOA model. A key concern is quality assurance and service selection. Service requestors require a method to evaluate software services based upon individual and interactive performance

of services. This technique can aide in the process of developing new services or selecting from existing services.

This Master's thesis aims at addressing this need for developers through the introduction of sensitivity analysis as it applies to software services. The objective is to evaluate software services in regard to their sensitivity to factor variations, and to determine the effects of factor variations on performance analysis in a quantitative manner.

Using a sensitivity analysis methodology, this technique will be applied to the analysis of software services, specifically an algorithm will be produced to automate the evaluation of a software service in a quantitative setting.

An experiment will be conducted to prove the effectiveness of sensitivity analysis. In the experiment, two software services will be available, each attempting to produce the same result. Through sensitivity analysis, it will be demonstrated that by utilizing the created algorithm, a developer may easily identify and select the most optimal solution among these services for use in their application. Also, service providers can evaluate the performance of their services to improve on its performance to provide quality assurance and increase its desirability.

5.4 Impact on the Industry

Through the introduction of sensitivity analysis into service-oriented software development, this will allow developers to produce better software services as they can now effectively measure the performance effects that factor variation may have upon their services. Using the information gathered through sensitivity analysis, a developer can then refine and enhance their services to provide further quality assurance to its users.

For service requestor wishing to use an existing software service, they now have a

method to evaluate a software service to identify whether the service is able to meet the demands of their application and can withstand the factor variations in their SOA model.

In the end, sensitivity analysis performance evaluations on software services will help to improve the software services industry and further promote SOA as a well-built, sturdy platform to build applications upon. Not only will service providers be able to select the best services to use, but superior services will continue to be developed and produced.

Chapter 6

Proposed Method

6.1 Methodology

6.1.1 Foundation for Proposed Method

The proposed method of addressing some of the performance related issues surrounding the SOA software model stem from a technique introduced in the early 1700's known as Design of Experiment (DOE). Design of Experiment was initially introduced as a method to identify what factors may trigger the onset of scurvy. It maintained the technique of observing and judging the significant of the output of input variables that act alone, in conjunction with one another, and at different values. Our proposed methodology applies this technique to modern day software services to effectively evaluate the performance of a software service. Its technique has been expanded to allow for the evaluation of factor variations on performance analysis in a quantitative manner.

6.1.2 Overview of Proposed Method

This section outlines the underlying mechanics involved in sensitivity analysis for multi-factor based applications. A service based application is typically composed of several loosely coupled components. Each component may require several parameters as input variables, and these variables may be discrete or continuous in nature [2]. The input variables to a software system can be considered as sensitivity factors. Sensitivity factors are when changes in the input variable values affect the performance of the software system. Sensitivity analysis aims to determine the impact that a particular or combination of variables will have if it differs from what is previously assumed.

6.1.3 Multi-Factor Based Sensitivity Analysis

Suppose we have a multi-factor based component with m factors A(w) where $1 \leq$ w $\leq$ m. Each factor may accept values at different levels identified by i_v for $1 \leq i_w \leq a_w$. The analysis must consider the effects of each factor individually, as well as the interactive effects from two up to all m factors. Illustrated in Equation 1 is a model for a multi-factor based sensitivity analysis with m factors.

$$
\begin{aligned}
Y_{i_1 i_2 \ldots i_m l} = {} & \mu + A^{(1)}_{i_1} + A^{(2)}_{i_2} + \ldots + A^{(m)}_{i_m} \\
& + A^{(1)} A^{(2)}_{i_1 i_2} + A^{1} A^{(3)}_{i_1 i_3} + \ldots + A^{1} A^{(m)}_{i_1 i_m} \\
& + A^{2} A^{(3)}_{i_2 i_3} + A^{2} A^{(4)}_{i_2 i_4} + \ldots + A^{2} A^{(m)}_{i_2 i_m} \\
& + \ldots + \\
& + A^{(m-1)} A^{(m)}_{i_{m-1} i_m} \\
& + A^{(1)} A^{(2)} A^{(3)}_{i_1 i_2 i_3} + \ldots + A^{(1)} A^{(2)} A^{(m)}_{i_1 i_2 i_m} \\
& + A^{(2)} A^{(3)} A^{(4)}_{i_2 i_3 i_4} + \ldots + A^{(2)} A^{(3)} A^{(m)}_{i_2 i_3 i_m} \\
& + \ldots +
\end{aligned}
$$

$$+A^{(m-2)}A^{(m-1)}A^{(m)}_{i_{m-2}i_{m-1}i_m}$$
$$+...+$$
$$+A^{(1)}A^{(2)}...A^{(m)}_{i_1 i_{(2)}...i_{(m)}} + \varepsilon_{i_1 i_2 ... i_m l}$$

Equation 1 Model for Multi-Factor Based Sensitivity Analysis

The subsequent discussion illustrates a case where m = 4 factors, but the generalization may apply to multiple factors. For the case of four factors, the general model takes the form of Equation 2. A variance table is then constructed in Table I based upon this model.

$$y_{hijkl} = A_i + B_j + C_k + D_h + AB_{ij} + AC_{ik} + AD_{ih} + BC_{jk} + BD_{jh} + CD_{kh} + ABC_{ijk}$$
$$+ABD_{ijh} + ACD_{ikh} + BCD_{jkh} + ABCD_{ijkh} + \mu + \varepsilon_{ijkl}$$

Equation 2 Model for Four-Factor Based Sensitivity Analysis

In the model, μ (the population mean) represents the average of all possible observations for $1 \leq i \leq n$. A_i is the effect of the ith level of factor A, B_j is the effect of the jth level of factor B, C_k is the effect of the kth level of factor C, and D_h is the effect of the hth level of factor D. The joint effects of factors A, B, C, and D are AB_{ij}, AC_{ik}, AD_{ih}, BC_{jk}, BD_{jh}, CD_{kh}, ABC_{ijk}, ABD_{ijh}, ACD_{ikh}, BCD_{jkh}, $ABCD_{ijkh}$ through the interaction between A_i, B_j, C_k, and D_h. ε_{ijkhl} is a random error component.

There are a total of seven steps involved in determining the effects of each factor individually and jointly on system performance.

1. A total of a * b * c * d * n experiments are to be performed with A, B, C, and D each being set to different values. The number of samples is represented by n. The

observed response y_{ijkhl} from each experiment is an output of a performance metric during performance analysis when A, B, C, and D take values at different levels indexed at i for $1 \leq i \leq a$, j for $1 \leq j \leq b$, k for $1 \leq k \leq c$, and h for $1 \leq h \leq d$.

2. The mean of performance responses are calculated by keeping one factor constant while varying the levels of all other factors within their value ranges. This process will produce four group means $y_{i...}$, $y_{.j..}$, $y_{..k.}$, $y_{...h}$ where y_i represents the total of all experiment observations under the ith level of factor A, $y_{.j..}$ under the jth level of factor B, $y_{..k.}$ under the kth level of factor C, and $y_{...h}$ under the hth level of factor D. Equation 3 represent the marginal means $y_{i...}$, $y_{.j..}$, $y_{..k.}$, $y_{...h}$ for factor A, factor B, factor C, and factor D.

$$y_{i...} = \sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{i...} = \frac{y_{i...}}{bcdn}$$

$$y_{.j..} = \sum_{i=1}^{a}\sum_{k=1}^{c}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{.j..} = \frac{y_{.j..}}{acdn}$$

$$y_{..k.} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{..k.} = \frac{y_{..k.}}{abdn}$$

$$y_{...h} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{...h} = \frac{y_{...h}}{abcn}$$

Equation 3 Individual Mean of Performance Responses

3. The mean of the joint performance responses are calculated. This process will produce 11 group means between $y_{ij..}$, $y_{i.k.}$, $y_{i..h}$, $y_{.jk.}$, $y_{.j.h}$, $y_{..kh}$, $y_{ijk.}$, $y_{ij.h}$, $y_{i.kh}$, $y_{.jkh}$, y_{ijkh} for factors A, B, C, and D.

$$y_{ij..} = \sum_{k=1}^{c}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{ij..} = \frac{y_{ij..}}{cdn}$$

$$y_{i.k.} = \sum_{j=1}^{b}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{i.k.} = \frac{y_{i.k.}}{bdn}$$

$$y_{i..h} = \sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{i..h} = \frac{y_{i..h}}{bcn}$$

$$y_{.jk.} = \sum_{i=1}^{a}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{.jk.} = \frac{y_{.jk.}}{adn}$$

$$y_{.j.h} = \sum_{i=1}^{a}\sum_{k=1}^{c}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{.j.h} = \frac{y_{.j.h}}{acn}$$

$$y_{..kh} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{..kh} = \frac{y_{..kh}}{abn}$$

$$y_{ijk.} = \sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{ijk.} = \frac{y_{ijk.}}{dn}$$

$$y_{ij.h} = \sum_{k=1}^{c}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{ij.h} = \frac{y_{ij.h}}{cn}$$

$$y_{i.kh} = \sum_{j=1}^{b}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{i.kh} = \frac{y_{i.kh}}{bn}$$

$$y_{.jkh} = \sum_{i=1}^{a}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{.jkh} = \frac{y_{.jkh}}{an}$$

$$y_{ijkh} = \sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{ijkh} = \frac{y_{ijkh}}{n}$$

Equation 4 Join Mean of Performance Responses

4. The overall mean $y_{....}$ of performance responses is calculated.

$$y_{....} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl} \qquad \bar{y}_{....} = \frac{y_{i...}}{abcdn}$$

Equation 5 Overall Mean of Performance Responses

5. The sum of squares is calculated for each individual factor as well as for all the combinations of factors.

$$SS_A = \frac{1}{bcdn}\sum_{i=1}^{a} y_{i...}^2 - \frac{y^2....}{abcdn}$$

$$SS_B = \frac{1}{acdn}\sum_{j=1}^{b} y_{.j..}^2 - \frac{y^2....}{abcdn}$$

$$SS_C = \frac{1}{abdn}\sum_{k=1}^{c} y_{..k.}^2 - \frac{y^2....}{abcdn}$$

$$SS_D = \frac{1}{abcn}\sum_{h=1}^{d} y_{...h}^2 - \frac{y^2....}{abcdn}$$

$$SS_{AB} = \frac{1}{cdn}\sum_{i=1}^{a}\sum_{j=1}^{b} y_{ij..}^2 - \frac{y^2....}{abcdn} - SS_A - SS_B$$

$$SS_{AC} = \frac{1}{bdn}\sum_{i=1}^{a}\sum_{k=1}^{c} y_{i.k.}^2 - \frac{y^2....}{abcdn} - SS_A - SS_C$$

$$SS_{AD} = \frac{1}{bcn}\sum_{i=1}^{a}\sum_{h=1}^{d} y_{i..h}^2 - \frac{y^2....}{abcdn} - SS_A - SS_D$$

$$SS_{BC} = \frac{1}{adn}\sum_{j=1}^{b}\sum_{k=1}^{c} y_{.jk.}^2 - \frac{y^2....}{abcdn} - SS_B - SS_C$$

$$SS_{BD} = \frac{1}{acn}\sum_{j=1}^{b}\sum_{h=1}^{d} y_{.j.h}^2 - \frac{y^2....}{abcdn} - SS_B - SS_D$$

$$SS_{CD} = \frac{1}{abn}\sum_{k=1}^{c}\sum_{h=1}^{d} y_{..kh}^2 - \frac{y^2....}{abcdn} - SS_C - SS_D$$

$$SS_{ABC} = \frac{1}{dn}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} y_{ijk.}^2 - \frac{y^2....}{abcdn} - SS_A - SS_B - SS_C - SS_{AB} - SS_{AC} - SS_{BC}$$

$$SS_{ABD} = \frac{1}{cn}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{h=1}^{d} y_{ij.h}^2 - \frac{y^2....}{abcdn} - SS_A - SS_B - SS_D - SS_{AB} - SS_{AD} - SS_{BD}$$

$$SS_{ACD} = \frac{1}{bn}\sum_{i=1}^{a}\sum_{k=1}^{c}\sum_{h=1}^{d} y_{i.kh}^2 - \frac{y^2....}{abcdn} - SS_A - SS_C - SS_D - SS_{AC} - SS_{AD} - SS_{CD}$$

$$SS_{BCD} = \frac{1}{an}\sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{h=1}^{d} y_{.jkh}^2 - \frac{y^2....}{abcdn} - SS_B - SS_C - SS_D - SS_{BC} - SS_{BD} - SS_{CD}$$

$$SS_{ABCD} = \frac{1}{n}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{h=1}^{d} y_{ijkh}^2 - \frac{y^2....}{abcdn} - SS_A - SS_B - SS_C - SS_D - SS_{AB} - SS_{AC} - SS_{AD} - SS_{BC} - SS_{BD} - SS_{CD} - SS_{ABC} - SS_{ABD} - SS_{ACD} - SS_{BCD}$$

$$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}\sum_{h=1}^{d}\sum_{l=1}^{n} y_{ijkhl}^2 - \frac{y^2....}{abcdn}$$

$$SS_E = SS_T - SS_A - SS_B - SS_C - SS_D - SS_{AB} - SS_{AC} - SS_{AD} - SS_{BC} - SS_{BD} - SS_{CD} - SS_{ABD} - SS_{ACD} - SS_{BCD} - SS_{ABCD}$$

Equation 6 Sum of Squares

6. Complete the variance table illustrated in Table 6 for each factor and their interactions with the calculated sum of squares and F distribution values. The degree of freedom (df) represents the number of independent variables for each associated sum of squares. Furthermore, a mean square (MS) is produced by dividing each sum of square by its degree of freedom. The final F distribution is obtained by dividing the mean square with the mean square error component.

Source	degree of freedom	Mean Square (MS)	F distribution
A	(a-1)	$MS_A = \frac{SS_A}{(a-1)}$	$F_0^A = \frac{MS_A}{MS_E}$
B	(b-1)	$MS_B = \frac{SS_B}{(b-1)}$	$F_0^B = \frac{MS_B}{MS_E}$
C	(c-1)	$MS_C = \frac{SS_C}{(c-1)}$	$F_0^C = \frac{MS_C}{MS_E}$
D	(d-1)	$MS_D = \frac{SS_D}{(d-1)}$	$F_0^D = \frac{MS_D}{MS_E}$
AxB	(a-1)(b-1)	$MS_{AB} = \frac{SS_{AB}}{(a-1)(b-1)}$	$F_0^{AB} = \frac{MS_{AB}}{MS_E}$
AxC	(a-1)(c-1)	$MS_{AC} = \frac{SS_{AC}}{(a-1)(c-1)}$	$F_0^{AC} = \frac{MS_{AC}}{MS_E}$
AxD	(a-1)(d-1)	$MS_{AD} = \frac{SS_{AD}}{(a-1)(d-1)}$	$F_0^{AD} = \frac{MS_{AD}}{MS_E}$
BxC	(b-1)(c-1)	$MS_{BC} = \frac{SS_{BC}}{(b-1)(c-1)}$	$F_0^{BC} = \frac{MS_{BC}}{MS_E}$
BxD	(b-1)(d-1)	$MS_{BD} = \frac{SS_{BD}}{(b-1)(d-1)}$	$F_0^{BD} = \frac{MS_{BD}}{MS_E}$
CxD	(c-1)(d-1)	$MS_{CD} = \frac{SS_{CD}}{(c-1)(d-1)}$	$F_0^{CD} = \frac{MS_{CD}}{MS_E}$
AxBxC	(a-1)(b-1)(c-1)	$MS_{ABC} = \frac{SS_{ABC}}{(a-1)(b-1)(c-1)}$	$F_0^{ABC} = \frac{MS_{ABC}}{MS_E}$
AxBxD	(a-1)(b-1)(d-1)	$MS_{ABD} = \frac{SS_{ABD}}{(a-1)(b-1)(d-1)}$	$F_0^{ABD} = \frac{MS_{ABD}}{MS_E}$
AxCxD	(a-1)(c-1)(d-1)	$MS_{ACD} = \frac{SS_{ACD}}{(a-1)(c-1)(d-1)}$	$F_0^{ACD} = \frac{MS_{ACD}}{MS_E}$
BxCxD	(b-1)(c-1)(d-1)	$MS_{BCD} = \frac{SS_{BCD}}{(b-1)(c-1)(d-1)}$	$F_0^{BCD} = \frac{MS_{BCD}}{MS_E}$
AxBxCxD	(a-1)(b-1)(c-1)(d-1)	$MS_{ABCD} = \frac{SS_{ABCD}}{(a-1)(b-1)(c-1)(d-1)}$	$F_0^{ABCD} = \frac{MS_{ABCD}}{MS_E}$
Error	abcd(n-1)	$MS_E = \frac{SS_E}{abcd(n-1)}$	
Total	abcdn-1	$MS_T = \frac{SS_T}{abcdn-1}$	

Table 6.1: Example of a Four-Factor Variance Table

7. The final step is to compare each F distribution value with the cumulative F distribution table value F_{α,df_1,df_2}. A significant effect on the performance of a software system can be determined if the individual factor effect or a joint group of several factors corresponding F distribution value exceeds F_{α,df_1,df_2}. The confidence interval is represented by ? and df1 and df2 are the degrees of freedom. The F-Distribution table is required at various confidence intervals in this step.

6.2 Sensitivity Analysis Web Service

6.2.1 Overview

The procedure to evaluate a software systems performance through the utilization of sensitivity analysis has been implemented into an algorithm. The algorithm is based upon the methodology for multi-factor based sensitivity analysis. The sensitivity analysis component accepts a sample from one or more web services with its performance responses at various factor values. The component uses this sample to generate a variance table that determines which factors (both individual and collaboratively) have the most significant effect on performance. This component has been extended into a service that is distributed over the Internet to allow any service provider to perform sensitivity analysis on their services to identify which factors have the greatest impact on their services output. Additionally, this component allows a service requestor to evaluate the performance of two services to identify which service is the most optimal to use based upon their sensitivity to factor variation.

6.2.2 Implementation Details

The sensitivity analysis service follows steps two to seven as outlined in our multi-factor based sensitivity analysis methodology. Step one is reserved as input for the service. The service uses the inputted dataset to create a variance table and calculate the F distribution.

The service exposes two modules, each with their own objective:

1. *getVarianceTable* - Generate a variance table that identifies the individual and joint effects of each factor based upon their sensitivity to factor variation.

2. *compareServices* - Evaluate the performance of two services and identify the less

sensitivity service as the most ideal service to use based upon their sensitivity to factor variation.

Input

The input for the getVarianceTable module is a collection of data from our software system or service. This data includes each factor and its value, as well as the response from the system. The data inputted into the algorithm can be observed and represented by Table 6.2.2.

Response	Factor 1	Factor 2	Factor 3	...	Factor m

Table 6.2: Sample Web Service Input

The input for the getVarianceTable module accepts a multi-dimensional array, A, where A[x][0] is the response from the system for 0 ¡= x ¡= length(A) and where A[x][i] is the value of each factor at 1 ? i ? n.

The input for the compareServices module that is responsible for performing a sensitivity analysis based comparison to identify the most optimal less sensitivity service has similar input requirements as the getVarianceTable. It accepts two multi-dimensional arrays, each with a sample of data gathered from each service. The structure of the array is identical to the one used in the getVarianceTable module.

Output

The sensitivity analysis operation getVarianceTable generates a variance table with the calculated F distribution for each individual and joint factors. This information is returned to

the user in the form a multi-dimensional array. The end user may use this information to identify the role and effect that each factor (and their combinations) have on the systems performance.

The second operation, compareServices, calculates each services sensitivity to factor variation and identifying the less sensitive service as the most ideal service to use by a service requestor. The operation simply returns the statement ”Service A is less sensitivity”, or ”Service B is less sensitivity”, where Service A is the first input and Service B is the second input to the compareServices operation.

6.2.3 Design

The sensitivity analysis service was developed using the Java programming language. The java programming language was selected as the development language because it is platform independent and it supports most major web service implementation techniques. The algorithm was initially developed to runs as a standalone java class, however it has been extended into a web service so that it may be distributed over the internet and used in a wide array of software applications. The service was developed using the SOAP implementation technique because of SOAP's wide acceptance for supporting higher end services.

There are two components to the sensitivity analysis service that aide in generating the variance table and performing the comparison. The first component is a combination generator. The combination generator is responsible for generating all 2n possible combinations for any number of factors [6]. The second component is responsible for performing sensitivity analysis through the use of the steps outlined in the methodology.

Combination Generator

The algorithm in Fig 6.1 generates 2n combinations for a given set in lexicographic order.

It was provided by Rosen in his textbook Discrete Mathematics and its Applications, 2007 [12].

Algorithm 5: Generate k-combinations of $\{1, \ldots, n\}$ in lexicographic order.

$a_1 a_2 \ldots a_k := 1\,2 \ldots k$ {first combination in lexicographic order}
while $a_1 a_2 \ldots a_k \neq n{-}k{+}1\ n{-}k{+}2\ \ldots\ n$
  $m :=$ the rightmost location among $1, \ldots, k$ such that a number larger than a_m but smaller than n is not in the combination
  $a'_1 a'_2 \ldots a'_{m-1} := a_1 a_2 \ldots a_{m-1}$ {retain everything to the left of a_m}
  $a'_m := a_m + 1$ {increase a_m by 1}
  $a'_{m+1}\, a'_{m+2} \ldots a'_k := a_m{+}2\ a_m{+}3 \ldots a_m{+}k{-}m{+}1$ {continue consecutively}
  $a_1 a_2 \ldots a_k := a'_1 a'_2 \ldots a'_k$
output $a_1 a_2 \ldots a_k$ {the members of each k-combination are given in ascending order}

Figure 6.1: Combination Generator Algorithm

Sensitivity Analysis

After all combinations are identified, the steps outlined in the methodology are executed with the input array. These steps involve:

- Calculating the mean of performance responses for each individual factor.
- Calculating the mean of performance responses for each combination of factors.
- Calculating the overall mean of performance responses.
- Calculating the sum of squares for each factor.
- Calculating the degrees of freedom.
- Calculating the F distribution.

```
procedure sensitivityAnalysis
   For i = 0 to factorCombinations.length
         fDistribution = calculateFDistribution(factorCombinations[i])

procedure calculateVarianceTable(int i)
   performanceMean[i] = calculatePerformanceMean(i)
   sumOfSquares[i] = calculateSumOfSquares(i)
   meanSquare[i] = calculateMeanSquare(i)
   degreesOfFreedom[i] = calculateDegreesOfFreedom(i)
   fDistribution[i] = calculateFDistribution[i]
```

Figure 6.2: Sensitivity Analysis Algorithm

Performing these steps will generate a variance table for the software service in question. The compareServices operation has an additional step that involves comparing the F distribution for the combination of all factors with F_{α,df_1,df_2} where α is our confidence level, and df is our degrees of freedom to determine whether all factors in combination have a significant impact on performance, and which of the two input services are less sensitive to factor variation. Performance is affected if the F distribution value exceeds F_{α,df_1,df_2}.

6.2.4 Time Complexity

The algorithm contains two main components; the combination generator and sensitivity analysis. The combination generator is responsible for identifying all individual and joint factor combinations. The sensitivity analysis algorithm calculates the mean square, sum of squares, degrees of freedom, and F-distribution.

Suppose there are n factors, then there are 2n individual and joint factors we must consider, as generated by the combination generator. The algorithm iterates through all 2n combinations and calculates the F distribution as seen in Fig 6.3.

```
For i = 0 to factorCombinations.length

        fDistribution[i] = calculateFDistribution( factorCombinations[i] )
```

Figure 6.3: Algorithm Time Complexity Analysis

The calculateFDistribution action is executed a total of 2n times since the length of factorCombinations is 2n. Thus, the sensitivity analysis algorithm has a running time of T(n) = O(2n) and is exponential.

Chapter 7

Experiment And Analysis

7.1 Experiment Analysis

7.1.1 Overview

To evaluate the performance of the sensitivity analysis service, an experiment is designed to demonstrate the advantages of using sensitivity analysis for service evaluation and service selection. Through this experiment, it is demonstrated how sensitivity analysis can address performance issues and improve the SOA software model from the standpoint of both the service providers, and the service requestors. Two web services have been created, Service A and Service B. Both web services have the same objective of helping the user select the most ideal vacation spot based on proximity to their current location, and desired destination temperature. Both Service A and Service B are composed of several smaller services to achieve their objective. Each service is unique which plays a critical role for the adjustment of relationships between services. The sensitivity analysis service that is created will be applied to both software services. Based upon the results, the service that

is less sensitivity to factor variation would be chosen as the most optimal service to use by a service requestor. The service that is most sensitivity to factor variation will then be modified and improved in hopes of reducing its individual and joint factor sensitivity. The sensitivity analysis comparison service is then performed again to show that the initial service that was more sensitive to factor variation is now improved and is less sensitive then the alternative service, thus is ultimately selected as the most optimal service. As demonstrated by the first part of the experiment, applying sensitivity analysis to SOA resolves performance issues related to service selection by helping service requestors to select the best service available for them to use. The second part of the experiment demonstrates that through the application of sensitivity analysis, a service provider can identify which factor and which combination of factors are most sensitivity to change, and improve their service accordingly to make them more desirable. This will ensure the production of high quality services.

7.1.2 Experiment Requirements

In order to successfully deploy the experiment involving two services and sensitivity analysis, various hardware and software requirements must be met. Table 7.1.2 illustrates the hardware requirements for the experiment. Table 7.1.2 outlines the software requirements for the experiment.

Product	Description	Specifications
Personal Computer	A computer is required to act as a host for all software services. As well, the PC is required to execute the experiment with the sensitivity analysis service.	Sufficient specifications are required to meet the demand of the selected operating system, Windows 7.

Table 7.1: Experiment Hardware Requirements

Software Title	Description	Availability
Microsoft Windows 7	The chosen operating system.	Available from Microsoft.
Eclipse IDE	The selected IDE for development.	http://www.eclipse.org
Java SDK	The development language software development kit (version 1.6)	http://www.sun.com
Service A	A software service composed of several smaller services. Service A has the same objective as Service B.	Created for the purpose of the experiment.
Service B	A software service composed of several smaller services. Service B has the same objective as service A.	Created for the purpose of the experiment.
Sensitivity Analysis Service	A software service created out of the sensitivity analysis methodology.	Created for the purpose of the experiment.

Table 7.2: Experiment Software Requirements

7.1.3 Sample Services

The experiment requires that two web services with similar end objectives be used to perform the sensitivity analysis experiment. To fulfill this requirement, two services, Service A and Service B have been created. The objective of both services is to identify potential

vacation destinations based upon the desired destination temperature and the destinations proximity to their current location. There are a total of thirty destinations available, each being a popular city located in the U.S.A.

Both services are composes of several smaller services to achieve their end objective. A total of three services are used in both Service A and Service B. This will allow us to identify the individual and joint effect each service has on the services performance.

The implementation method for both Service A and Service B is based on the SOAP model. The SOAP model was chosen because it provides the most flexibility and greatest compatibility across platforms. The Java development language was used during development.

The services consist of several loosely coupled components operating independently as web services. These components are connected through the control interface. The interface executes each individual service sequentially, collecting the results from one service and passing that information on to the following service. During the creation of Service A and Service B, the following sub-services are used as represented in Table 7.1.3 and Table 7.1.3:

Service Name	Description	Availability
GlobalWeather	Retrieves the weather for a desired city.	http://www.webservicex.net
CityZipService	Determines the zip code in the desired cities.	http://www.ecocoma.com/
DistanceService	Calculates the distance between two zip codes.	http://www.ecocoma.com/

Table 7.3: Service A Composition

Service Name	Description	Availability
USA Weather Forecast	Retrieves the weather for a desired city.	http://www.webservicex.net
USA Zip code Information	Determines the zip code in the desired cities.	http://www.webservicex.net
LocationByZip	Calculates the distance between two zip codes.	http://www.flash-db.com

Table 7.4: Service B Composition

7.2 Case Study

A total of three factors are chosen for this study. Factor A is the number of cities to use as potential vacation spots in the U.S.A. Factor B is the minimum desired destination temperature (degrees Celsius), and factor C is the ISP (Internet Service Provider) of the current internet connection. Each factor has variations at three fixed levels, and a total of two experiments are performed with factors A, B, and C. Factors A, B, and C are set to different values indexed respectively at i for $1 \leq i \leq 3$, j for $1 \leq j \leq 3$, and k for $1 \leq k \leq 3$. Through the use of the multi-factor based sensitivity analysis service, the data sample for Service A and Service B is collected and displayed in Table 7.3.1 and Table 7.3.1. The variance table for each service is created through the service and displayed in Table 7.2 and Table 7.2.

Combo	Sum of Squares	Degrees of Freedom	Mean Square	F Distribution
A	2.047	2	1.023	134.985
B	1.995	2	9.979	13.156
C	3.568	2	1.784	23.518
AB	4.371	4	1.092	1.44
AC	5.127	4	1.281	1.689
BC	5.459	4	1.364	1.799
ABC	9.97	8	1.246	1.642
Error	2.048	27	7585801	
Total	3.058	53	5.77	

Table 7.5: Service A Variance Table

Combo	Sum of Squares	Degrees of Freedom	Mean Square	F Distribution
A	1.646	2	8.231	402.741
B	4.97	2	2.485	12.159
C	2.125	2	1.062	5.2
AB	2.74	4	6.851	3.352
AC	1.606	4	4.015	19.647
BC	8.66	4	2.165	10.592
ABC	8.451	8	1.056	5.168
Error	5.518	27	2.043	
Total	2.131	53	4.021	

Table 7.6: Service B Variance Table

7.3 Analysis and Discussion

7.3.1 Initial Results

For Service A, at a confidence level of 1%, factors A, B, and C have a significant impact on system response time due to the fact that F_0^A at 134.985, F_0^B at 13.156, and F_0^C at 23.518 exceed the cumulative F distribution table value $F_{0.01,2,27}$ which is 5.49. However, the combination of factors AB, AC, BC, and ABC show little factor sensitivity since F_0^{AB} at

1.440 F_0^{AC} at 1.689, F_0^{BC} at 1.799, and F_0^{ABC} at 1.642 are below the cumulative F distribution table value of 5.49. Changing the confidence level to 5% so $F_{0.05,2,27}$ at 3.35 shows no change as the individual factors A, B, and C all have a significant impact on performance, and the combination of factors AB, AC, BC, and ABC do not have a significant effect on performance.

For Service B, at a confidence level of 1%, factors A and B have a significant effect on system response time due to the fact that both F_0^A at 402.741, and F_0^B at 12.159 exceed the cumulative F distribution table value $F_{0.01,2,27}$ which is 5.49. In addition, the combination of factors F_0^{AC} at 19.647, and F_0^BC at 10.592 have a significant effect on performance, exceeding the cumulative F distribution table at 5.49. However, by changing the confidence level to 5%, factor C shows significant impact as F_0^C at 5.200 exceeds $F_{0.05,2,27}$ which is 3.35.

A: Num of Cities	B: Min. Temp.	C: Bandwidth		
		U. of Windsor	Cogeco	Rogers ISP
10	15	12968	15375	18406
		8438	10391	28047
	25	7812	10172	12125
		7891	10000	13687
	35	7188	8937	9203
		7156	20047	9344
20	15	17625	21609	22531
		17078	22610	21297
	25	15875	20407	19813
		15953	22734	22343
	35	14437	17641	22954
		14390	16719	16875
30	15	27344	27891	39360
		25265	26422	30843
	25	23735	26375	31969
		25406	25375	34438
	35	22547	22891	27625
		22110	23187	25687

Table 7.7: Service A Data Sample

A: Num of Cities	B: Min. Temp.	C: Bandwidth		
		U. of Windsor	Cogeco	Rogers ISP
10	15	26250	29891	29843
		23657	28063	25734
	25	28860	29281	26985
		23547	28844	35703
	35	22422	32594	32719
		24969	40735	35266
20	15	45406	23812	52047
		44188	15438	64609
	25	45125	17375	47890
		49562	40437	51531
	35	44609	54500	48641
		44032	55859	51578
30	15	66094	85890	74672
		63532	83594	68735
	25	59844	67578	66360
		62797	62766	68094
	35	74735	89984	64594
		72422	89578	63672

Table 7.8: Service B Data Sample

Using this information, it is evident that Service B's performance is affected by all three factors, with the greatest influence being placed on the number of cities, and minimum destination temperature. The samples from both Service A, and Service B are then used as inputs for the comparison module in the sensitivity analysis service. The comparison module uses a sample from two services to identify which service is the most optimal based upon their sensitivity to factor variation. The comparison service looks at the combination of all factors A, B, and C and selects the service which is least sensitivity to changes in the combination of all factors. The results are that Service A is less sensitivity to factor variation, and thus is a more reliable and consistent service when compared to Service B. This demonstrates that if we are to use sensitivity analysis as a means of service selection,

Service A would be the most optimal solution available to use.

Using the information gathered through the sensitivity analysis service, a service requestor is provided with quality assurance that Service A is less sensitivity to factor variation. Thus, the service requestor is able to confidently select Service A as the most optimal service to use in their application. This demonstrates that sensitivity analysis is a valuable technique in service selection, and is a suitable approach to address concerns related to service selection in the SOA software model.

The service provider of Service B is able to review the results of sensitivity analysis, and see that their service is highly sensitivity to factor variation. The service provider can see that all three factors; the number of cities, desired destination temperature, and internet connection have a large impact on performance. Using this information, the service provider can make certain modifications to their service to improve its performance and make them more desirable.

For the case of this experiment, Service B will be modified in an effort to reduce its sensitivity to factor variation, make it more desirable, and ultimately have it selected as the ideal service to use when the sensitivity analysis comparative web service is run. There are three approaches that will be taken to improve upon Service B's performance to reduce its sensitivity to factor variation. These approaches include:

1. Modifying Service B to optimize and improve its performance.

2. Swap the web service in Service B that is responsible for obtaining the current weather in each city.

3. Both, modify Service B to optimize and improve its performance, as well as swap the web service responsible for obtaining the current weather in each city.

7.3.2 Approach 1: Modify Service B to Improve Performance

For the first approach of modifying Service B in an effort to optimize and improve its performance, various optimization techniques were performed. Since the service is largely composed of several smaller web services, modifications to how the web services calls were made. The web service calls are now made synchronously to reduce the time required to retrieve the requested data. Additionally, the sorting algorithm has been modified to use the merge sort algorithm, as oppose to the previous bubble sort. Through these various techniques, the following data sample and variance table is created, as displayed in Table 11 and Table 12.

A: Num of Cities	B: Min. Temp.	C: Bandwidth		
		U. of Windsor	Cogeco	Rogers ISP
10	15	9323	11125	9640
		8459	11516	10015
	25	9321	13281	9531
		8343	12031	10750
	35	7834	7453	7609
		6023	8391	8203
20	15	14323	14890	17844
		16032	18875	16672
	25	12343	21390	15844
		16233	13250	14469
	35	14532	16906	13687
		16434	13172	16453
30	15	22323	38968	23875
		20232	19000	21343
	25	21032	15844	18828
		19833	17688	21223
	35	18434	27625	20121
		16954	25844	18534

Table 7.9: Approach 1: Service B Optimized Data Sample

Combo	Sum of Squares	Degrees of Freedom	Mean Square	F Distribution
A	1.331	2	6.657	63.492
B	5.134	2	2.567	2.448
C	6.971	2	3.485	3.324
AB	6.567	4	1.641	1.565
AC	1.815	4	4538683	0.432
BC	1.517	4	3794454	0.361
ABC	1.111	8	1.389	1.325
Error	2.83	27		
Total	1.945	53		

Table 7.10: Approach 1: Service B Optimized Variance Table

Using this information, at a confidence level of 1%, factor A is the only factor, to have a significant effect on the system, as shown by F_0^A at 63.492 exceeding the cumulative F distribution table value $F_{0.01,2,27}$ at 5.49. Changing the confidence level to 5% yields the same results. Observing the combination of factors ABC, the factor sensitivity has been significantly reduced. Service B previously had an F distribution of F_0^{ABC} at 5.168, and has since been reduced to 1.325.

7.3.3 Approach 2: Replace Web Services in Service B

The second approach involves replacing the web service responsible for retrieving the current weather in each city. The new web service is titled US Weather and is provided by WebServiceX. Through replacing this web service, the information in Table 7.3.3 and Table 7.3.3 is obtained.

A: Num of Cities	B: Min. Temp.	C: Bandwidth		
		U. of Windsor	Cogeco	Rogers ISP
10	15	13323	16687	13047
		13589	15531	14511
	25	13321	12391	11331
		12983	13468	12434
	35	12834	13515	12111
		12094	11266	11454
20	15	14392	16281	14232
		14201	18891	15343
	25	13999	14532	15333
		12934	16781	14053
	35	12832	19000	15555
		13584	22203	15011
30	15	17043	23437	17933
		16454	25342	17101
	25	15393	23485	16989
		14599	21328	16787
	35	14923	20734	16953
		14403	23500	16535

Table 7.11: Approach 2: Service B Replace Web Services Data Sample

Combo	Sum of Squares	Degrees of Freedom	Mean Square	F Distribution
A	2.626	2	1.313	123.976
B	1.907	2	9537235	9.003
C	1.794	2	8.97	84.688
AB	1.726	4	4315160	4.073
AC	7.195	4	1.798	16.982
BC	6608950	4	1652238	1.559
ABC	1.766	8	2207625	2.084
Error	2.86	27	1059269	
Total	6.032	53	1.138	

Table 7.12: Approach 2: Service B Replace Web Services Table

Analyzing the results of replacing one of the web services in Service B shows that factors A, B, and C all have a significant impact on system performance. This is shown by F_0^A at 123.976 , F_0^B at 9.00359, and F_0^C at 84.688 all exceed the cumulative F distribution table value $F_{0.01,2,27}$ at 5.49. The combination of factors AC is shown to have high factor sensitivity with F_0^{AC} at 16.982 exceeding the cumulative F distribution table value of 5.49. Additionally, changing the confidence level to 5% shows that factor AB also has a significant effect on performance with F_0^{AB} at 4.073 exceeding $F_{0.05,2,27}$ at 3.35.

7.3.4 Approach 3: Optimize and Swap Web Services in Service B

The final approach involves combining the previous two approaches. Service B is optimized to improve upon its performance through the use of synchronous web service calls, as well as through the use of the merge sort algorithm. Additionally, the web service responsible for determining the weather in each city has also been replaced. These modifications produced the data sample and variance table displayed in Table 7.3.4 and Table 7.3.4.

A: Num of Cities	B: Min. Temp.	C: Bandwidth		
		U. of Windsor	Cogeco	Rogers ISP
10	15	20593	26859	22859
		19584	26578	19578
	25	21443	34797	23797
		24303	31360	21360
	35	24343	29641	21641
		22543	26171	19171
20	15	26944	35078	25078
		27945	31468	23468
	25	26844	30234	24234
		25000	33297	23297
	35	28549	29781	23781
		27454	30125	21125
30	15	32012	44593	30593
		31053	46703	28703
	25	29857	45829	28829
		33593	43719	27719
	35	29545	42391	28391
		28733	41360	27360

Table 7.13: Approach 3: Optimize and Replace Web Services in Service

Combo	Sum of Squares	Degrees of Freedom	Mean Square	F Distribution
A	9.897	2	4.948	206.834
B	2.142	2	1.071	4.476
C	1.104	2	5.524	230.888
AB	4.671	4	1.167	4.881
AC	1.646	4	4.115	17.2
BC	2.025	4	5064438	2.116
ABC	2.163	8	2703816	1.13
Error	6.46	27	2392677	
Total	2.433	53	4.592	

Table 7.14: Approach 3: Optimize and Replace Web Services in Service

Analyzing the information in the variance table for the final approach of Service B that includes both optimized and service replacement techniques, shows it is evident that factors A, and C have a significant impact on the system performance since at a confidence level of 1%, F_0^A at 206.834, and F_0^C at 230.888 exceed the cumulative F distribution table value $F_{0.01,2,27}$ at 5.49. For the combination and interaction of factors, factor AC has a significant effect on performance with and F_0^{AC} at 17.200 exceeding $F_{0.01,2,27}$ at 5.49. Comparing the results of the interaction of all three factors A, B, and C shows that F_0^{ABC} has little impact on the systems performance.

7.4 Final Result

Performing the comparison service again with the modified approach 3 of Service B, and the initial version of Service A shows that Service B is less sensitive to factor variation than Service A, and is ultimately chosen as the most ideal service to use. Thus, the optimized version of Service B is now the more desirable service. This technique demonstrates that through the use of sensitivity analysis, a service provider is able to analyze the performance of their services, and take the necessary actions to improve upon their performance to make

them increasingly desirable by service requestors. Additionally, service requestors are provided with a technique to identify and select the most ideal service for their needs based upon sensitivity to factor variation.

Chapter 8

Conclusion And Future Work

Service Oriented Architecture (SOA) is a modern software model that is rapidly increasing in popularity. It's loosely coupled, language and platform independent, reusable approach has businesses rethinking the way they develop applications in their enterprise software. As most new technologies do, SOA has a few obstacles and drawbacks that are limiting its potential. One of the more noteworthy issues with SOA is performance. More importantly, how well does a service perform in regards to its sensitivity to factor variation. Sensitivity analysis aims to evaluate a software system to determine the effects of individual and joint factors on the system. Through sensitivity analysis the issues surrounding SOA's performance, specifically quality assurance and service selection, are addressed.

The impact of sensitivity analysis on a SOA based system is that service requestors have a guideline to identify and select the most ideal software services for their application, and service provides have a technique to ensure the reliability, efficiency, and optimality of the software services they aspire to create in an effort to increase desirability.

This Master's thesis proposed a technique to address the need for both service providers and service requestors through the introduction of sensitivity analysis as it applies to soft-

ware services. The objective is to evaluate software services in regard to their sensitivity to factor variations, and to determine the effects of factor variations on performance analysis in a quantitative manner. A sensitivity analysis service is produced to automate the evaluation of a software system in regards to factor sensitivity. An experiment was conducted to demonstrate the effectiveness of sensitivity analysis.

The results of the experiment have identified that the use of multi-factor based sensitivity analysis is a useful tool in performance analysis of SOA based system. Through the experiment, it was demonstrated that sensitivity analysis can aide service requestors in the process of identifying and selecting the most optimal service available for their needs. Furthermore, the experiment shows how service providers can use the results of sensitivity analysis to optimize their services to make them more desirable. This technique of sensitivity analysis proved useful in resolving the performance related issues of quality assurance and service selection. Further research may be beneficial to improve the current method of sensitivity to factor variation by considering the effects of uncontrollable factors.

In the end, through the introduction of sensitivity analysis on SOA based systems, this will ensure the continued growth and acceptance of SOA as a highly capable software model that is adapted to today's needs.

Bibliography

[1] The Benefits of Service-Oriented Architecture (SOA). *avail: http://www2. roguewave.com/support/docs/leif/leif/html/leifintroug/2-4. html*, 2005.

[2] L. Breierova and M. Choudhari. An introduction to sensitivity analysis. *Prepared for the MIT System Dynamics in Education Project*, 1996.

[3] J.W. Chinneck. Practical optimization: a gentle introduction. *Electronic document: http://www. sce. carleton. ca/faculty/chinneck/po. html*, 2004.

[4] A. Dan, R.D. Johnson, and T. Carrato. SOA service reuse by design. In *Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 25–28. ACM New York, NY, USA, 2008.

[5] T. Erl. *Service-oriented architecture: concepts, technology, and design*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[6] M. Gilleland. Combination Generator. *avail: http://www. merriampark. com/comb. htm*, 2007.

[7] M.D. Hansen. SOA using Java web services. 2002.

[8] E. Honour. Design of Experiments as Applied to Systems Engineering Return on Investment. In *Proceedings of the Conference on Systems Engineering Research, Hoboken, NJ*, 2007.

[9] N. Kanakalata, U. Banerjee, and S. Kumar. Performance optimization of SOA based AJAX application. In *Proceeding of the 2nd annual conference on India software engineering conference*, pages 89–94. ACM, 2009.

[10] M. Li, J.P. Huai, and H.P. Guo. An Adaptive Web Services Selection Method Based on the QoS Prediction Mechanism. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 395–402. IEEE Computer Society, 2009.

[11] W. Mendenhall, R.J. Beaver, and B.M. Beaver. *Introduction to probability and statistics*. Brooks/Cole, 2008.

[12] J.G. Michaels and K.H. Rosen. *Applications of discrete mathematics*. McGraw-Hill, 1991.

[13] Y. Natis. Service-oriented architecture scenario. *Gartner, Inc., Stamford*, 2003.

[14] D.J. Pannell. Sensitivity analysis: strategies, methods, concepts, examples.

[15] J. Phelps and B. Busby. Service-Oriented Architecture-What Is It, and How Do We Get One? *Educause Quarterly*, 30(3):56, 2007.

[16] L. Richardson and S. Ruby. *RESTful web services*. O'Reilly Media, Inc., 2007.

[17] K. Sahina and M. Gumusay. SERVICE ORIENTED ARCHITECTURE (SOA) BASED WEB SERVICES FOR GEOGRAPHIC INFORMATION SYSTEMS. 2008.

[18] A. Saltelli, K. Chan, E.M. Scott, et al. *Sensitivity analysis*. Wiley New York, 2004.

[19] X. Yuan, S. Duan, and T. Huang. Analysis of service-oriented architectures with sensitivity analysis. In *2006 IEEE International Conference on Electro/information Technology*, pages 372–376, 2006.

[20] X. Yuan and C. Ji. Performance analysis of service-oriented architectures with multi-factor sensitivity analysis. In *2007 IEEE International Conference on Electro/Information Technology*, pages 198–203, 2007.

Vita Auctoris

NAME:	Mohit Sud
PLACE OF BIRTH:	Toronto, Ontario
YEAR OF BIRTH:	1985

EDUCATION:

1999-2003	Erindale Secondary School, Mississauga, Ontario
2003-2008 B.Sc.	University of Windsor, Windsor, Ontario
2008-2010 M.Sc.	University of Windsor, Windsor, Ontario