1-1-2011

Effective Quality Of Service Browsing For Web Service Selection

Shilpi Verma
Ryerson University

EFFECTIVE QUALITY OF SERVICE BROWSING FOR WEB SERVICE SELECTION

by

Shilpi Verma

B.Tech – Computer Science and Engineering,

National Institute of Technology (NIT), India, 2008

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2011

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

SHILPI VERMA

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

SHILPI VERMA

EFFECTIVE QUALITY OF SERVICE BROWSING FOR WEB SERVICE SELECTION

Master of Science, 2011

Shilpi Verma

Computer Science

Ryerson University

ABSTRACT

The growing number of Services on the Web has made locating desired Web Services a sizeable challenge. Web Service requestors deem a Quality of Service (QoS) based Web Service selection important in terms of providing a relevant and user centric service selection experience. In this thesis an interactive QoS based Web Service browsing mechanism is proposed, which makes use of three clustering algorithms including vector-based, preference-based and weighted clustering. We use symbolic interval data as the principle representation of QoS attributes. The browsing mechanism which was implemented as part of this research allows service requestors to prioritize their search by hierarchically clustering their web services. This is done in the order of their preferences and also by attaching a weight to each QoS attribute, which is a beneficial compromise between performance-high preference-based clustering and time-efficient vector-based clustering. Along with several extensive experiments, a user study was conducted in order to test the usability of this browsing mechanism and to test the overall efficiency and performance of the three clustering algorithms in comparison. The result of the experiment led to evidences that preference-based browsing approach was the most efficient one when compared to vector-based or weighted clustering approaches.

ACKNOWLEDGEMENTS

It gives me great pleasure to be able to express my heartfelt gratitude to all the people who have helped me complete this significant milestone. Firstly, I am humbly indebted to my supervisor Dr. Chen (Cherie) Ding, whose support, patience and guidance has made a monumental contribution to my thesis. She has steered me in the right direction by giving me her valuable ideas, her time and constant guidance. Her intellectual thoughts and innovative ideas have always inspired and driven me to accomplish challenging tasks. I will always be deeply obliged to Dr. Ding for making this possible and staying by my side, helping me achieve this milestone.

I would like to sincerely thank my Co-supervisor Dr. Isaac Woungang for his support and his valuable time in reviewing my thesis. I would also like to thank the members of my thesis defence committee, Dr. Vojislav Misic, Dr. Abdolreza Abhari and Dr. Eric Harley for their valuable time, support and suggestions.

I am grateful to all the professors at the Department of Computer Science in Ryerson University, for dedicating their time in keeping me inspired and educated.

I am very thankful to Alex Yakobovich for his immense help, guidance and encouragement in helping me with the implementation of my ideas. I'd also like to take this opportunity to thank some key people who helped me stay on track with my goals and steered me towards the right path and they are: Sonal, Alexey, Kian, Lev, Patrick, Peter and Preethy. Finally and most importantly, I would like to thank my parents Ms. Meera Verma and Mr. Chandra Shekhar Verma, and my older sister Nidhi, for always being there for me and supporting me through every walk of my life. It is their dedication, encouragement, selflessness, and prayers that keep me going.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

GLOSSARY OF ACRONYMS

W3C	World Wide Web Consortium
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
HTTP	Hyper Text Transfer Protocol
WSDL	Web Service Description Language
RPC	Remote Procedure Call
XML	Extensible Mark-up Language
UDDI	Universal Description, Discovery and Integration
QoS	Quality of Service
MCDM	Multiple Criteria Decision Making
WSM	Weighted Sum Model
GUI	Graphical User Interface
SLA	Service Level Agreement
WSLA	Web Service Level Agreement
UML	Unified Modeling Language
WSCE	Web Service Crawler Engine
QWS	Quality of Web Service
CPC	Cost Per Click

CHAPTER 1

INTRODUCTION

1.1 Background

The World Wide Web today has evolved into a dynamic mesh of integral Websites, Web applications, and Web Services. The W3C (World Wide Web Consortium, 1991) defines Web Services as "a software system designed to support interoperable machine-to-machine interaction over a network" (Brugger, 2010). Web Services are based on SOA (Service Oriented Architecture) which is basically a set of principles that are followed when designing software systems, with characteristics such as reusability, interoperability, autonomy and platform independency, etc.

With SOA as a building block, a set of protocols called SOAP (Simple Object Access Protocol) was evolved. Web Service use the SOAP as an envelope to exchange all their messages. Since SOAP is an XML-based protocol which uses RPC (Remote Procedure Call) dialogues and HTTP (Hyper Text Transfer Protocol) as its transfer protocol, it is fairly versatile and supports interoperability. The behaviour of the Web Service however, is left to be described by WSDL (Web Service Description Language). The WSDL specification provides an XML format for documents that define services as an assortment of ports and messages that are bound together in a reusable manner. A WSDL document provides any client an overview of all the operations available on the Server.

In order to link services from providers to clientele, UDDI (Universal Description, Discovery and Integration) was created. UDDI specifies the standard for a service where Web

Service providers list themselves over the Internet enabling clients to find them and also keep track of all the services available. Over the years, UDDI has become stagnant, tedious with obsolete metadata structure. Several service providers like IBM, SAP and Microsoft have unlisted their listings from the public UDDI nodes (SAP NEWS DESK, December 18, 2005).

1.2Motivation

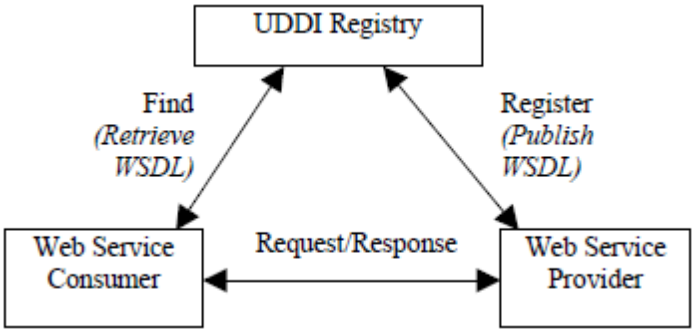

Figure 1 Web Service publish-bind-find model (Ran, 2003)

The Web Service publish-bind-find model (Figure 1) is largely dependent on the UDDI registries which often contain obsolete metadata, incorrect or broken links and incorrect information (Clark, 2001) and have a big drawback of based only on functional parameters. Even though several attempts are being made to advance from using UDDI, it is still considered to the central repository for all service specifications and therefore, a better search and discovery model is called for.

With fast moving technologies, the demand for interactive and adaptive Web Services has become preordained. Search engines like *Woogle* (Dong, Madhavan, & Halevy, Mining structures for semantics, 2004) and *Seekda* (Seekda corporate author, 2007) have come up with efficient crawls based on similarity searches and pattern matching. The biggest limitations with Web Service search engines however are not only the browsing experience but also the fact that

they do not address non-functional searches amongst other things. Non-functional requirements specify the operational aspects or the behaviour of a system, in other words, non-functional requirements indicate how the system should *be* as opposed to functional requirements which state what the system should *do*.

It was thus ascertained that QoS based Web Service selection and browsing mechanisms fall at the core of competent and powerful Web Service search. To address the challenges of an efficient QoS based search, clustering Web Services based on their attributes in an interactive browsing system was established to be an effective resolution. Clustering is a method used to group similar data together. It was adopted as a measure to group similar Web Services together based on their functionality and requesters' selection criteria.

Amongst one of the prime motivating factors was also to provide the users with the option to have a say in the service selection process. The promise user interaction, coupled with QoS awareness would make people more confident in the use of Web Service selection for critical tasks. This adds to the foundation of trustworthy service-oriented computing. The idea of using Multiple Criteria Decision Making (MCDM) (which has been addressed in the Preference-based model) and a weighted-sum model (WSM) (which has been reflected in the Weighted model) diversifies the users' prospects further.

Given the fact that QoS attributes are not always expressed in single numerical variables; and the fact that clustering is an evolved data analysis process capable of handling any objects, including symbolic variables in a vector form – it was promptly decided to opt for it for the purpose of simulating the browsing tool.

1.3Problem Statement

In the light of the current dynamic Web environment, it was ascertained that merely performing syntactical service search is not enough and it only adds a functionality filter to the user's search process. With increasing service populace over the Internet, there exist a number of non-trivial services with similar functionalities. Therefore, a QoS based service selection approach is much called for and even though there have been a number of research efforts on QoS aware Web service discovery and selection, much is to be said about the user's experience and involvement in the process.

Service requestors are usually known to have vague quality constraints and requirements due to which searching on fixed numerical values have posed a problem for several solutions proposed in the past. An important factor which is often overlooked is that the service requestors may not have the full knowledge of QoS attributes, or query forming skills which would match them to their desired Web Services. The need for a browsing tool which interacts with the user, as well as adapts to the user's fastidious decisions, has been made evident.

The next problem is that of the data representation of QoS attributes since they are non functional values which do not have any consistent representation. QoS attributes are best depicted symbolically, for which reason, interval variables were chosen because interval data is the most common type of symbolic data. Then the QoS data can be expressed in the form of a vector, where each vector consists of all the attributes of a particular Web Service, and every attribute value is represented as an interval. In order to extract patterns from these QoS vectors or organize them in an effective way, the clustering technique is considered.

Lastly, a selection system could support multiple QoS attributes; however, a single user may not be interested on all of them. It would be important for the system to provide a way for users to define their preferences so that the data analysis could emphasize on these preferred attributes.

1.4Proposed Solution

In order to resolve the problems stated in section 1.2, we propose an approach which would ease the process of browsing through the functionally similar services on their QoS values so as to facilitate the later service selection process.

Streamlining the user requirements and helping end-users understand and filter their constraints is addressed with the help of the browsing tool which, with the aid of a step-by-step interactive approach and a visualization feature, eases the process.

Since QoS attributes are best depicted symbolically and because Web Services are known to have a range of values, interval type was considered to be the best representation for each attribute.

Finally, since vector-based clustering techniques do not provide any flexibility and often yield only partially satisfactory results, a preference based and weight based mechanism is used to provide the user with a better browsing and selection experience.

1.5Objectives of the Thesis

The main purpose of this thesis is to design and implement a QoS browsing tool as a complement to the existing QoS-based searching and selection techniques. Due to the various data distribution patterns present in the QoS dataset, the popular vector-based clustering process

may not work well for all the datasets. Therefore we take into consideration another two clustering processes – preference-based clustering and weighted clustering. The former allows users to define their preferences on QoS attributes and takes a multi-levelled approach to cluster the dataset and present the result, so that the most preferred attribute values would be clustered first. Also by clustering on one or a small number of attributes at a time, it puts less cognitive overload on users to comprehend the presented information. The latter also allows the preference definition in the form of the attribute weights, and the clustering is done on the whole vector, however, with different weight on different dimensions. All three clustering processes are implemented and their effectiveness of dealing with different datasets are investigated and compared. In order to make a viable contribution in the domain of Web Services, a usability experiment was conducted to test the capability, performance and feasibility of the implemented tool. This experiment provided the research with a quality assurance and a tangible concept in the field of Web Service selection.

The browsing tool which was implemented has a Graphical User interface (GUI) for the user to navigate through and also has a visualization component where users can see the Web Service clusters and their distribution in the search space at every step. The GUI has built-in tracking mechanism for time taken at each step as well as a counter for the number of steps taken by the user to arrive at the final result.

The contributions made by this work, are listed as follows:

1. Web Service selection has been taken one step further by introducing QoS based browsing to functionally filtered Web Services. This allows for a more precise and user-oriented result which is one step over traditional service matching.

2. Three service selection browsing methods – vector-based, preference-based and weighted QoS browsing methods were investigated and implemented for comparison. Earlier, vector-based clustering method has already been investigated (Sambamoorthy, QoS Browsing for Web Service Selection, 2009) using a pre-packaged software called SODUS©. This work uses the base model of that work only for vector-based clustering approach.
    2.1 The vector-based browsing method, although exists in practice, was implemented so a comparison model could be developed to evaluate the performance of all three methods.
    2.2 The preference-based method breaks the constraints of the vector-based approach and allows the requestor to prioritize the order of the QoS attributes based on their personal preferences. This allows for an iteratively tapered process which leaves the requestor with cluster(s) of desired services.
    2.3 The weight-based approach is a compromised solution between vector and preference based methods and it takes advantages from both sides – allowing users to define their preferences on various QoS attributes as well as the time-efficiency of the vector-based method.
3. In order to provide the user with as near a real experience as possible, symbolic data was used to signify each attribute variable as opposed to single numeric data, thus adding another dimensionality of range to the data. It was confirmed that K-means clustering would be the most scalable clustering algorithm since it requires fewer passes on the entire dataset as opposed to other clustering algorithms.

To keep the experience as real as possible, the data sets used were all simulated and compiled based on inspirations from values of several real data and use-cases respectively. The use-cases, data distribution patterns and dataset values were all inspired from real time Web Service found on Web Service search engines like Seekda (Seekda corporate author, 2007).

1.6 Organization of the Thesis

The remainder of the thesis is organized as follows:

Chapter 2 – Literature Review: This chapter is dedicated to providing a narrative of all the relevant research efforts made in this field. It encompasses the various Web Service selection methods which are relevant to this study; the various features and characteristics of Web Services and their functional and non-functional categorization; and among other concepts, Data clustering, symbolic data types, and various clustering algorithms have also been discussed. Finally, all ends are tied together to the theory behind the three browsing mechanisms implemented in this thesis.

Chapter 3 – Different Comparative Clustering Processes for Web Service Selection: This chapter explains each of the algorithms for vector-based, preference-based and weighted K-means clustering which was implemented for the interval data. Here, the case of data clustering is stated with detailed explanations of K-means and the each of the subsequent algorithms.

Chapter 4 – Experimental analysis and performance evaluation: This chapter accounts the experiments and evaluations conducted in order to analyze the performance of each of the browsing methods in comparison with each other. Firstly, the data generation and the simulation scenarios are explained based on which the datasets were simulated for the experiments.

Next, the experiments are divided into three parts – the first one is to evaluate the accuracy of each of the three service selection methods with the help of certain datasets; the second experiment records the efficiency of the algorithms and their performance when certain metrics were tweaked and the third experiment is a usability study conducted over a small demographic to test the performance and viability of the browsing tool. The inferences of all the evaluations are then summed up as the result.

Chapter 5 – Conclusion: This chapter concludes by summarizing the key achievements of this research work with potential recommendations and future work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the various related works which were investigated for this research study have been discussed. Some of the different domains which will be tackled in this section are: QoS for Web Services, Data clustering, and Web Service discovery and selection.

2.1 QoS for Web Services

With the advent of increasingly large number of Web Services exchanging hands between service providers and subscribers, functional and syntactical service discovery methods yield qualitatively similar Web Services in significant numbers. The term Quality of Service refers to all non-functional features of a service that may be used to evaluate its quality and performance characteristics. With a number of services match-made to functionalities, a semantic solution based on QoS-based Web Service discovery is in high demand. Syntactical service discovery mechanisms make use of SOA, which is a major software framework which when used with standard protocols such as SOAP and WSDL help compose Web Services with same functionalities. However, various services may have different practical applications such as a banking system that requires services providing a high level of security as opposed to a health care information system requiring quicker response time (XQ, X.W., & C-J, 2011); yet another example would be a hotel-car rental system requiring high availability. Service requestors' global restrictions to functionally composite services may be met effectively by making use of QoS based service discovery and selection methods.

QoS attributes have been classified into separate domains in order to quantify and organize them and drawing from several research studies (Liu, Ngu, & Zeng, 2004), (Ran, 2003), the following are the four main classifications:

- Run time related: These QoS-based attributes are related to the run-time of the Web Services and include scalability, capacity, performance, reliability, avail-ability, robustness, exception handling, and accuracy.
- Transaction support related: These QoS-based attributes are related to the core characteristics of the Web Services and include integrity, atomicity, consistency, isolation and durability.
- Configuration management and cost related: These QoS-based attributes are based on the configurations and cost management of the Web Services and include regulatory features, supported standard, stability, cost, and completeness.
- Security related: These QoS-based attributes are based on the security parameters of the Web Services and include authentication, authorization, confidentiality, accountability, traceability, data encryption, and non-repudiation.

Composite Web Services are composed from a set of abstract Web Services using the SOA paradigm, from which a concrete service is selected and used. This ensures flexibility and loose coupling within this process. QoS parameters play a significant role in determining the success or failure of the composed application. Therefore, a Service Level Agreement (SLA) is often used as a contractual agreement between service providers and service users (Alrifai, Skoutas, & Risse, April 2010). Along the same lines, is the important matter in the subject of QoS is the service actually rendered to the consumers by service providers. In order to ensure the delivery and quality of the service, just as advertised in the WSDL of the Web Service, a

framework is put in place which caters to the terms, conditions and agreements between the service providers and the service requestors. This is known as the Web Service Level Agreement (WSLA) which not only monitors the SLA that binds the two parties but also specifies risk and failure mitigations. The WSLA essentially provides a runtime architecture and language for SLAs specification and also determines the logistics and negotiations throughout the SLA life-cycle. Apart from SLAs, service registries, service consumer feedback and third party monitoring agents also provide information about web services' quality attributes (Farhana & Patrick, 2011).

2.2 Data Definition

In order to perform well-rounded experiments in a QoS aware browsing environment, the data used had to be drawn from multiple sources verified by publishers, UDDI registry listings, verified third party engines and SLAs. For this reason, QoS data was examined from various sources such as *Woogle* (Dong, Halevy, Madhavan, Nemes, & Zhang, 2004), *Seekda* (Seekda corporate author, 2007), (Vu, Hauswirth, Porto, & Aberer, 2006) and Amazon (Amazon Web Services, 2006).

The processes of data representation and data generation conducted for the purpose of this research have been explained in section 3.3 and section 4.2 respectively. The software chosen for simulating these datasets was MATLAB® (MathWorks), which is a MathWorks™ product.

2.3 Data Clustering

Cluster analysis is an expansive domain which deals with various types of data and application areas. For this purpose of this research study, a survey on cluster analysis was carried

out by initiating on a broad base with fundamental concepts and algorithms existing currently. With a view of focussing solely on clustering semantic relationships and performing QoS based clustering for Web Service data, this survey provides an insight to the type of clustering methods finally employed along with its benefits and limitations.

Clustering is defined as the process of grouping together similar objects with an objective of increasing intra cluster similarity and inter cluster dissimilarity. Clustering is an important technique since it helps mine useful data from large databases and determines meaningful relationships between the data in various fields (Han & Kamber, 2001). The two main categorizations used for data clustering are partitioning clustering and hierarchical clustering method. Partitioning method allows for a dataset of n objects to be partitioned into k clusters, (where $k <= n$) and where each group contains at least one object belonging to exactly one group. An example of this heuristic approach is K-means clustering, where the each group or cluster is iteratively calculated based on the collective mean of the group.

Hierarchical methods create a hierarchical decomposition of the given data set either in an agglomerative approach of a divisive approach. In an agglomerative approach a bottom up technique is used where each object belongs to a group and eventually these groups are merged together based on their similarity. In a divisive approach all objects belong to the same group and eventually they are divided into sub groups based on their similarity measure.

K-means is a type of Partitioning clustering method where n objects (x_1, x_2,... x_n) can be partitioned into k clusters (S_1, S_2,... S_k) so as to increase the intra-cluster similarity. In order to proceed with clustering any given set of data using K-means, the initial prototypes (also known as mean and later known as centroid) must be determined. The simplest way to select the initial

prototypes is through a random selection. For *k* clusters, *k* number of prototypes must be selected (m_1, m_2... m_k). The following are the steps followed to perform K-means clustering:

Step 1: Every data object is assigned to any one mean or centroid located nearest to it. This distance is measured using any standard distance measurement depending upon the type of data (for example - Euclidean distance measurement, Manhattan distance measurement, Minkowski distance measurement etc).

Step 2: The centroid of each cluster is re-calculated by taking the mean of the distances between each object and the centroid for that cluster. The data objects are re-assigned to the centroid nearest to them as shown in the assignment step below:

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

(MacKay, 2003)

Once the new centroid has been calculated, the re-assignment is done as follows:

$$S_i^{(t)} = \{ x_j : \| x_j - m_i^{(t)} \| \leq \| x_j - m_{i^*}^{(t)} \| \text{ for all } i^* = 1,... ,k\}$$

(MacKay, 2003)

Step 3: Step 2 is repeated until the algorithm converges such that the centroid no longer changes from the previous iteration.

Since QoS attributes have a wide range of data types and the nature of the aggregated data is symbolic, it is was important to consider to a class of clustering algorithms which would be able to handle symbolic and interval data. Interval data is a collection of continues data values which represent a variable or a token. QoS attributes often have values which are better expressed as interval variables since it is not often accurate to take an average, mean or median

of variables and express them as single points of reference when they are better expressed as a series of values.

Interval data clustering (Carvalho, Brito, & Bock, Dynamic clustering for interval data based on L2 distance, 2006) (Carvalho, Souza, Chavent, & Lechevallier, 2006) (Peng & Li, 2006) has had several applications in dynamic and symbolic data clustering. However, since interval data which is symbolic makes use of similarity and dissimilarity measures which are based on location, range and content of the symbolic data type rather than only its numerical value.

One of the dynamic clustering algorithms explored (Souza & Carvalho, 2004) makes use of interval data for clustering using a two-step relocation process. Once the initial prototypes have been identified and the data objects have been assigned allocated to the cluster centers based on their proximity to the prototypes, the algorithm iteratively locally optimizes the cluster allocations until the adequacy function converges. The distance between data objects and the cluster center is measured by two adaptive versions of the city-block distance. In another paper (Chavent, Carvalho, Lechevallier, & Verde, 2006), the dynamic clustering algorithm is used with Hausdorff distance measure and the two-component dissimilarity measure. These distance measures have been known to perform exceedingly better when compared to other forms of distance measurement which better handle numerical or other data types (Marie & Lechevallier, 2002).

2.4 Web Service Discovery and Selection

With prolific growth in the service discovery and selection area, choosing the most suitable method given the number of techniques based on the structural and semantic information

of services has become a daunting task. Service requestors are often faced with information overload or tedious syntactical mechanisms or rudimentary search engine crawls.

The domain of Web Service discovery and selection is ever evolving but may be classified with the following sub-domains (Abramowicz, Haniewicz, Kaczmarek, & Zyskowsk, 2007) (Ran, 2003) (L.Vu, Hauswirth, Porto, & Aberer, 2006) (Wang & Stroulia, 2007) (Xu Z. , Martin, W.Powley, & Zulkernine, 2007):

- Functionality Based Selection Methods
- QoS-based Selection Methods
- Trust and Reputation Based Methods

Functionality based methods refer to search and discovery processes that deal with syntactical and match-making service discovery based on the functionality offered by the Web Service. QoS-based methods focus on discovering Web Services on a more qualitative scale by finding services based on their non-functional, performance related attributes. Trust is a subjective entity which refers to a personalized single-user score or rating on the performance of a service (Wang & Vassileva, 2007); reputation on the other hand is the public opinion aggregated on a service gathered by the collective scores from individuals about a particular service (Keller & Ludwig, March 2003).

There are several research works and studies that have been carried out in the field of preference-based Web Service selection and weight-based algorithms. One such study (Fan, Zhang, Shen, & Wang, 2010) focuses on selecting Web Services based on the user's perspective of the qualitative and quantitative constraints. In this paper, the authors have suggested that although the user may know which qualitative factor is important for them, they may not know

what quantitative values these factors should have. Thus, two methods have been explored where subjective weights are assigned to the qualitative factors of the Web Service using the indifference curve of Cobb-Glass method of preference (Cobb & Douglas, 1928). The service selection model is based on a cloud model. The important aspect about this paper was the focus on selecting Web Services based on the QoS factors and then assigning weights based on user-preference.

Another QoS-based model based on user preference (Cao, Huang, Wang, & Gu, 2009) employs Q-WSEM (QoS-based Web Service Evaluation Model) which arrives at suitable Web Services by making use of Web Service evaluation center strategically placed in the middle of a three layer framework. The first and second layer of service evaluations narrow the constraint based results on the basis of QoS attributes by using normalization and multi-criterion optimization. Here, "the preference weight algorithm takes the relative importance judgment of users about QoS attributes as input parameter." A decision matrix is used to organize the multi-attribute decision making and to facilitate the third layer of the service selection model.

Multi criteria decision making (MCDM) is another important aspect which is important to consider when dealing with multiple QoS attributes and user preference. MCDM is a decision supporting discipline which strategizes arriving at decisions based on user preferences with the help of a decision matrix when several criterions exist. There are some research studies that have been carried out based on the principle of MCDM (Li-Li & Yan, 2009) where the QoS ontology is transformed into the OWL-S standards and multi criteria decision matrices help extend weights assigned to QoS attributes based on user preferences into generating suitable Web Services. The values of the quality parameters are normalized here, and a comparison study proves that higher numbers reflect higher levels of service. Another technique (Herssens, Jureta,

& Faulkner, 2009) uses a UML (Unified Markup Language) based model to gather and represent the users' preference and priority relationships between QoS parameters. A fuzzy MCDM technique is then used to build a set of references, which when assigned with weights provides a ranking and comparison model.

Another important contribution in the field of selection Web Services based on QoS parameters is the usage of Top-k Skyline algorithm (Borzsonyi, Kossmann, & Stocker, 2001). The authors of the research study in consideration (Alrifai, Skoutas, & Risse, April 2010) have proposed a way to reduce the search space of the candidate services from which to select the most suitable Web Service; the Skyline method has then been used to break down the end-to-end QoS constraints specified in the SLA and a hybrid approach is used to locally select only the most desirable QoS levels of each class. This is achieved by first representing a skyline of the services in each service class, and then uses K-means clustering to recursively cluster the services from which one service is chosen as a representative and presented to the user. This paper also focuses on which service needs attention on which QoS level so as not ot get dominated by other services.

Another research work which was proposed in the field of QoS based Web Service selection (Sambamoorthy, QoS Browsing for Web Service Selection, 2009) is considered to be an important background material for this research. In this study, the author proposed a browsing method for QoS based Web Service selection using symbolic data analysis and the SODUS software (Diday & Noirhomme-Fraiture, 2008). In this study, the nature of data related to Web Services is considered to be symbolic and therefore dealt with as vector-based sets of data representing the various types of QoS values. Allowing the array of values Web Services can have, interval based data is used for which purpose the dynamic clustering algorithm based on k-

means, called SCLUST, within SODUS software is used for clustering Web Services. This method also goes on to determining the optimal K-value by making use of certain statistical indices called C-H index, C index and Γ-index (Hardy & Baune, 2007), (Mali & Mitra, 2003). This work, although considered vital, had its limitations in that the datasets chosen were drawn from a single source and were neither drawn from real QoS agents, search engines or SLAs nor simulated to reflect real scenarios and use cases. Due to the nature of the dynamic clustering algorithm SCLUST in the SODUS software, only vector-based data was considered which coupled with the lack of a visual interface yielded only partially useful results. The lack of a multi-level approach and any user interaction further isolated the results.

2.5 Summary

In this chapter, the various related works that were studied as background material and considered precedential were accounted. The four main sections in which this chapter was divided were QoS for Web Services, Data definition, Data Clustering and Web Service discovery and selection. The motivation giving way to semantic service discovery and the various non-functional QoS domains have been explain in the first section, following which the various works in the field of data clustering and symbolic data analysis relevant to this study were explained. The research leading to the simulation of data sets and the implementation of a graphical browsing system was explained in the next two sections where various studies conducted in the field of QoS based Web Service selection as well as in user-centric, preference and weight-based techniques were divulged.

CHAPTER 3

CLUSTERING PROCESSES FOR WEB SERVICES

This chapter focuses on three different clustering processes for implementing QoS browsing on Web Services. It starts from a few motivating examples showing the necessity of using different clustering algorithms for the effective QoS browsing. In order to better understand the QoS browsing and the purpose behind its mechanism, certain areas such as QoS parameters and the attributes chosen for the purpose of this study; clustering, its types and specifically k-means clustering algorithm; data definition, representation and its distribution, will be discussed.

3.1 Motivating Scenarios

Oftentimes, service requestors may not have a set constraint or requirement; they may not know how to form search queries or have only vague pre-requisites. Furthermore, there is little chance that the user would know and understand which QoS attributes should have which values. This indicates towards the need for a browsing tool which would interact with the user to help narrow down their requirements and help locate desired service(s).

Let us consider an example where a user is interested in subscribing to a Web Service which provides him/her with the daily weather forecast of his/her city. The requestor would use keywords to describe the functional requirement on the service by searching on a web service search engine such as Seekda (Seekda corporate author, 2007). When it comes to searching on non-functional constraints such as its QoS values, the following are some plausible scenarios:

a) The user may not know the range of the QoS attributes (i.e. the cost range of subscription for weather related services is from $40 to $100, however the user puts down the

requirement as “cost < $30”). In the event that the user does have some idea of what the ranges of the service QoS values are like, if they provide a query with multiple requirements, (e.g. “cost < $100” and “response time < 20ms” and “reliability > 95%”), the result may be null because no service fulfills all the constraints in rigid terms. However, if the user was exposed to the minimum and maximum ranges of the values, they may have been able to make an intelligent query with desirable results.

b) If the user were to form a query for a Web Service with the value of reliability >=99% and cost < $100, the result may be null because all services have a reliability of 97% or less. This alone is a justification for providing the users with a clustered result so if their query was to view results with services having reliability >95%, they would see all the services between 95% and 100% reliability and thus be able to pick one.

c) An important drawback of using vector-based clustering would be the fact that it ties the user to only one attribute of choice. In vector-based clustering, should the user desire attribute values which fall into different clusters, the user would have to choose only one. However, if the user is provided with the clusters on an attribute by attribute basis, in the order of the most preferred attribute(s), the user will get their pick every step of the way. For example, if there are three clusters as shown in Table 1, the user would not be able to choose the desired values as marked below:

Cluster	Cost	Response Time	Reliability
Cluster 1	100-199	30-39	60-69
Cluster 2	200-299	20-29	70-79
Cluster 3	300-399	40-49	80-89

Table 1 - Example of drawbacks of vector-based clustering

However, as denoted in Table 1, if the user was presented every cluster for every attribute separately, in the order of preference, it would allow them to choose their desired value for every attribute. Alternately, to be more time-efficient, if a weight was attached to each attribute signifying the preference order, the user would be able to get a customized vector-based clustering clustered in the desired priority.

d) It can also be inferred from table one that if a user wished to have different number of clusters to choose from for each attribute, it would not be possible in vector-based clustering. However, a step-by-step multi-level approach provisioned by preference-based clustering does allow users to vary the number of cluster they desire for each attribute.

e) Another aspect to consider here is the tendency to trigger an information overload on service requestors. A query result with a list of Web Services having multi dimensional vectors with a number of different values could overburden the requestor and lead to ill-formed decisions.

A lot of research has been done in this field and other clustering based, Web Service discovery tools have been proposed (Sambamoorthy, Interactive QoS bowsing for web service selection, 2009). However, the need for a unique browsing system which exceeded the limitations of integer data type, vector based clustering and yet be effective and efficient in terms of CPU expense, time and customer satisfaction were the key driving elements for this study.

3.2 Quality of Service and Web Service Discovery

The term “Quality of Service” is an expression used to state non-functional requirements for different areas such as network research community and in real time issues (Cruz, 1995) (CLARK, 1992). For the purpose of this study, Quality of Service (QoS) is defined as a set of

non-functional attributes which depict the quality delivered by a Web Service. QoS of web services can be organized into several categories such as run-time related, security related or transaction related [reference]; some of the most popular ones (and the ones used for data simulations in this study) are listed as follows:

1. Cost – Cost describes the price of the service listed by the service provider. More often than not, this price is determined by the number of invocations or simply for a flat time period of subscription. For the sake of simplicity, this attribute will be measured in U.S. Dollars ($).
2. Response Time – Response time is the time taken by a Web Service to complete a request made by a user. This time is either the guaranteed max or the average time required to complete a service task (Gunther, 1998). This attribute is measured in milliseconds (ms).
3. Reliability – The ability of a Web Service to deliver successfully as stated in its WSLA (Web Service Level Agreement) is known as its reliability. It may be measured by: "*Mean time between failure (MTBF), Mean time to Failure (MTF), and Mean time to transition (MTTT)*". It is closely related to the availability of a Web Service (Gunther, 1998). It is measured in percentage (%).
4. Availability: The probability of a system to be up and accessible is known as availability. It may be measured by:

$$A = \frac{< upTime >}{< totalTime >} = \frac{< uptime >}{(< uptime > + < downtime >)} \qquad (1)$$

where:

<*upTime*> is the total time the system has been up during the measurement period.

<*downTime*> is the total time the system has been down during the measurement period.

<*totalTime*> is the total measurement time, is the sum of <*upTime*> and <*downTime*> (Gunther, 1998). It is measured in percentage (%).

5. Accessibility: Accessibility may be defined as the probability of successful reach and installation by service requestors at a given point of time. It is measured in percentage (%).
6. Security: Security may be defined as the level of privacy a web service provides to its subscribers. These may include confidentiality measures, third party access to user information, message encryption and providing access control. This attribute may be measured on a scale of 100, 100 being the most secure, private and rigid in terms of access control and 1 being the most lenient.
7. Compliance: Compliance relates to the successful accordance of the service i.e. the percentage of times the users' requests have been understood and complied with successfully. It is measured in percentage (%).
8. Latency: Latency is measured as the delay time of the service itself, or the amount of first-response time taken by the service to acknowledge the receipt of a request (it is different from response time, as response time is the actual time taken by the service to complete the users' request). It is measured in milliseconds (ms).
9. Flexibility: Flexibility of a Web Service is the degree to which it functions correctly even in the presence of invalid or conflicting inputs (Gunther, 1998). It is measured in percentage (%).

QoS attributes also function as a benchmark which distinguishes one service provider from the other and build its trust and reputation. Research efforts have been made where users would be allowed to submit a rating or feedback on their experiences of service qualities or third

party agents would perform non-biased surveys and monitoring on service qualities in order to avoid service providers from promoting incorrect and conflicting QoS values.

In order to follow industry best practises and to achieve a standard for these QoS attributes, major efforts such as Web Service Level Agreements (WSLA) [(IBM Corporation, 2003), (Keller & Ludwig, March 2003), (Ludwig, A., Dan, & King, March, 2003)] and Web Service Policy Framework (Bajaj, 2006) were put in place. These standards represent a complex framework focusing not only on QoS specifications, but also on a complete set of aspects related to Web Services. WSLA documents refer to contracts signed between service providers and subscribers upon agreeing to the terms of service with respect to quality. The challenge however remained in the assurance of compliance of such contracts and agreements. Monitoring agencies and third party agents are therefore used as policing agents who account for QoS deliveries and their accuracy levels.

3.3 QoS data representation

It has been established that QoS data has been categorized and defined to cater to the service providers and requestors. These attributes, by their very nature, are dynamic and in order to describe and represent them correctly, any single valued data variable is unsuitable and inappropriate.

A tModel is a data structure defining a Web Service in the UDDI registry. It organizes the specifications of the Web Service and makes it available on the service registry. There have been studies [(Devis, Antonellis, & Melochiori, 2004), (Lamparter, Ankolekar, Studer, & Grimm, 2007), (Ran, 2003)] where the value of the QoS variable is represented as a single valued integer variable, for example, in the tModel given below:

```
<categoryBag>

<keyedReference

tModelKey="uddi:uddi.org:QoS:Price"

keyName="Price Per Transaction"

keyValue=" 0.01" />

<keyedReference

tModelKey="uddi:uddi.org:QoS:ResponseTime"

keyName="Average ResponseTime"

keyValue="0.05" />

<keyedReference

tModelKey="uddi:uddi.org:QoS:Availability"

keyName="Availability"

keyValue="99.99" />

<keyedReference

tModelKey="uddi:uddi.org:QoS:Throughput"

keyName=" Throughput"

keyValue="500" />

</categoryBag>

</tModel>
```

Figure 2 - Sample tModel (Xu Z. , Martin, Powley, & Zulkernine, 2007)

In the tModel shown in Figure 2, values for services have been simplified and thus compromised in their accuracy. Often, services may have more than one value for a particular constraint. Furthermore, users often have range requirements instead of a requirement of a fixed single value, e.g. a service with cost greater than $50 but less than $ 100 or a service with

reliability > 95%. A single value or measurement unit may not accurately identify the value of the service or its magnitude on a scale. Service providers often have to provide the average, mean or median of the actual deliverables, which sometimes completely deviates from the real picture. It would be more suitable from the requestor's perspective, to provide them with a range of values which indicate the promised assortment of values. This would not only increase the accuracy and assurance value of the quality but also align with the compliance clauses of the respective WSLA. Another positive outcome of adapting attributes with a range would be incorporating scenarios where publishers need only provide a maximum and minimum value of their services which would allow potential subscribers to gauge their minimum and maximum values (for example their performance or reliability highs and lows) and make an informed decision.

Since QoS attributes have a wide range of data types in which they can be expressed; such as Boolean, real, integer, enumeration, etc. [(Devis, Antonellis, & Melochiori, 2004), (Liu, Ngu, & Zeng, 2004)]. It can thus be ascertained that QoS attributes are best represented as symbolic data type, among which the interval type is the most common one. Interval data is described as "a group of variables, each of which contains a range of continuous values instead of the traditional single continuous or discrete values" (Peng & Li, 2006). An Interval data type would resolve the inaccuracy problem caused by representing QoS values with the average or mean value, and in the meantime, provide the user with the ability to define their requirements in unrestrictive or vague terms such as a service with "$40 < price < $100", a "response time < 300 milliseconds" and "reliability > 95%". Since WSLAs already have the provision of representing QoS attributes as interval values (Ludwig, A., Dan, & King, March, 2003), the datasets used to in this study have made use of interval variables of a symbolic data type which have eliminated

the need to normalize the data, thus adding the robustness other standardized, manipulated or normalized data may lack.

As discussed in previous works (Liu, Ngu, & Zeng, 2004), due to the dynamic nature of Web Services, it isn't feasible to maintain a single, static central repository for all QoS parameters, some of which may be domain specific applicable to certain Web Services and not the others. Therefore, a number of generic QoS criteria, such as cost, response time, reliability and other attributes already discussed and defined would be stored in a matrix of data where each row would represent a Web Service and each column a QoS attribute. Thus, each row would consist of a vector of QoS attribute values.

3.4 Clustering Processes for QoS data

3.4.1 Overview

As mentioned earlier, QoS data of web services can be represented as vectors of interval data. In normal clustering process, a data object (in our case a vector) is the unit for clustering. So the vector-based clustering is the first clustering process we are going to study. We have presented in Section 3.1 that in some use case scenarios, vector-based clustering would not work properly to reveal the true pattern in the dataset. And therefore, we propose another two clustering processes we could follow according to the data distribution patterns – preference-based clustering and weighted clustering. In preference-based clustering, a QoS vector will be segmented into a few parts based on user's preferences. Services will be clustered based on one or a few QoS attributes at a time, following their preference orders, starting from the most preferred ones to the least preferred ones. In this way, if a dataset has natural groupings on one or a few attributes instead of the whole attribute set, the pattern could be revealed from the clustering result. Since preference-based clustering may take longer time to achieve the final

result compared with the vector-based clustering, weighted clustering is considered as a compromised solution between the two clustering processes. In weighted clustering, weights will be assigned to different attributes based on user preferences, in the hope that the pattern on most important attributes could be revealed more clearly. Weighted clustering also forms clusters on the whole vectors, and thus its efficiency level should be comparable to the vector-based clustering.

3.4.2 K-means clustering for Interval data

K-means clustering is a type of partitioning clustering algorithm. It is probably the most well-known and commonly used clustering algorithm. It is chosen as the clustering algorithm for our vector-based clustering process due to its simplicity and its provision to allow users to fix the number of clusters a priori. Initially, *k* objects are chosen randomly as the centroids (cluster centres or cluster prototypes) of *k* clusters where each object is placed into any one of these *k* clusters depending on which centroid is closer in distance. Based on this distribution of objects, the algorithm then calculates new *k* centroids which are the bary-centres of the clusters from the previous step [(Han, Taehwan, & J.). The redistribution of objects within these new *k* centroids recurs iteratively until the centroids are stable and the algorithm has converged. In order to perform the distance measurements to determine the dissimilarity between interval data objects, the most commonly used distance measures are city block or Hausdorff distance measure.

The steps for K-means algorithm can be described as follows:

1. The number of required clusters *k* is determined where each data object is placed in its own cluster and random *k* values are chosen as the cluster centres of the dataset. These initial cluster centres are also known as cluster prototypes.

2. Based on an appropriate distance measure, city block or Hausdorff in case of interval-based vector data, the data objects are each assigned to the centroid nearest to them.
3. Once all the data objects have been assigned to a cluster, a recalculation of centroids occurs where cluster centres are now determined as the arithmetic-mean or average of all the points in the cluster.
4. The data objects are now reassigned to centroids based on the new distance calculations and the last two steps are repeated iteratively until the new centroids no longer change and are the same as the old centroids and the algorithm's convergence criteria is met.

Algorithm 1 - Algorithm depicting the steps for K-means clustering for QoS data

Since the input to the clustering algorithm is in the form of a vector, a set of *n* QoS vectors, $QS = \{Q_1, Q_2, \ldots, Q_n\}$ are described by *p* interval variables. Each QoS vector Q_i ($i=1, 2, \ldots, n$) is represented as $x_i = (x_i^1, x_i^2, \ldots, x_i^p)$ where, $x_i^y = [a_i^y, b_i^y]$ ($y=1,2,\ldots,p$) represent the start and end points of the interval values.

The convergence criterion is defined by:

$$W(P,G) = \sum_{k=1}^{K} \sum_{CQ_i \in C_k} D(CQ_i, G_k) \qquad (2)$$

where,

$P = (C_1 + C_2 + \ldots + C_k)$ of Qs in k clusters and a set of cluster prototypes $G = (G_1, G_2, \ldots G_K)$

$D(CQ_i, G_k)$ is the dissimilarity measure between $CQ_i \in C_k$ and the cluster centre G_k *of* C_k.

The city block distance is defined as the sum of the differences between the upper and the lower bounds of the interval which represent each attribute in the QoS vector. Hausdorff distance

is given by the maximum value of the upper and lower bound values of the interval for the two QoS vectors.

The city block distance and the Hausdorff distance are described as follows:

$$D_{CB}(Q_i, Q_j) = \sum_{h=1}^{p} (|q_{hs,i} - q_{hs,j}| + |q_{he,i} - q_{he,j}|) \qquad (3)$$

$$D_H(Q_i, Q_j) = \sum_{h=1}^{p} \max(|q_{hs,i} - q_{hs,j}|, |q_{he,i} - q_{he,j}|) \quad (4)$$

where, q_s and q_e refer to corresponding interval bounds of the QoS attribute of the i^{th} and j^{th} quality vectors.

3.4.3 Three Comparative Clustering approaches

In previous works done in this field where vector-based clustering has been used to browse and select Web Services based on non-functional parameters, some of the key drawbacks have been the inability to cluster and browse uninhibitedly between attributes, choosing and select any clusters in desired order to preference. Information overload on users and the inability to reveal true data patterns have also been significant limitations.

In order to understand the preference-based and weighted clustering processes on QoS data, it is important to understand the schema of vector-based clustering process. The following are the steps followed in an interactive vector-based clustering:

1. Let N = Number of functionally similar services.
2. Let $QS = \{Q_1, Q_2, \ldots, Q_N\}$ be a set of N QoS vectors and p the number of QoS attributes in QS_i for (i=1,2,…N).
3. Input QS to the interval clustering algorithm. Present the clustering results of k clusters to requestors as i) G_k given by ([$gq_{1s,k}$, $gq_{1e,k}$], ([$gq_{2s,k}$, $gq_{2e,k}$], …, [$gq_{ps,k}$, $gq_{pe,k}$]) prototypes in (C_1, C_2, …, C_k) clusters ii) n; the size of cluster C_j for (j = 1…, k) and iii) Range [$q_{maxs,i}$, $q_{maxe,i}$] for entire partition P(k) and (i = 1,2…,p).
4. Input *condn;* (where *condn*- requestor's input)
5. While (*condn* = yes)
6. Requester selects (k*); where k* is the requestor's selection from K clusters based on QoS attribute values.
7. Repeat steps 3-4 for QS_{k*}
8. End browsing.

Algorithm 2 - Steps for Vector-based clustering on QoS data

In vector-based algorithm, each vector (which is made up of several interval attribute values) is treated as a whole and clustered using K-means. The above steps are similar to the dynamic clustering algorithm for QoS data implemented in the SODUS software package as seen in similar research efforts previously (Sambamoorthy, Interactive QoS bowsing for web service selection, 2009). However, several drawbacks in that study apart from the drawbacks of vector-clustering itself were the absence of any user interaction, the ambiguity in cluster distinction due to the lack of clear visualization and the lack of control over the service selection decisions.

In comparison with vector-based clustering, preference-based and weighted clustering methods have a more open approach to interaction and multi-level selection process. The steps for preference-based clustering will now be explained which will allow a better understanding of

the framework that has been put in place for the service selection tool. The general scheme of the algorithm is as follows:

---

1. Let $QS = \{Q_1, Q_2, \ldots, Q_n\}$ be a set of n QoS vectors described by p interval variables.
2. Each QoS vector Q_i $(i=1, 2, \ldots, n)$ is represented as $x_i = (x_i^1, x_i^2, \ldots, x_i^p)$

   where:

   $x_i^y = [a_i^y, b_i^y]$ $(y=1,2,\ldots,p)$ represent the start and end points of the interval values.

   - If **preference-based clustering** method is chosen,

     then

     The requestor is asked to order the QoS attributes according to his preference from *1* to z $(1<=z_1<=z_2<=\ldots<=z)$ where $(z <= p)$
   - Set $j=1$ for the attribute with first priority and set $QS_j = QS$.
3. The user is asked to input the number of clusters k.
4. Random k mean values (centroids) are chosen from the set QS $(m_1, m_2, \ldots, m_k)$.
5. Each interval object is assigned to its closest mean using city block distance measurement for symbolic interval objects.

   $S_i^{(t)} = \{ x_j \mid\mid d(x_j, m_r^{(t)}) \mid\mid <= \mid\mid d(x_j, m_{r*}^{(t)}) \}$

   For all $r* = 1, 2, \ldots, k$.

   Where, $d(x_j, m_r) = \sum_{y=1}^{p} \varphi(x_i^y, m_r^y)$ such that;

   $\varphi(x_i^y, m_r^y) = | a_j^y - \alpha_j^y | + | b_j^y - \beta_j^y |$
6. Now, the new mean of the centroid of the observations in the cluster is calculated:

   $m_r(t+1) = 1 / (S_i(t) \sum x(j) \in S_i(t) x_j$
7. Step 6 and step 7 are repeated until convergence has reached and all objects belong to a certain cluster.

8. The result of S_i clusters is presented to the requestor with related information such as size and prototype.
9. The requestor may then make an informed decision and choose *h (0 < h <=S_i)* clusters and then continue with step 12. However, if *j > 1*, the user may choose to go back to the previous level to change a previous decision. In this case, *j* is set to *j*-- and the user is taken back to step 5.
10. *11: j* is set to *j*++ and QS_j = {Q_i | Q_i € the *h* selected clusters). As long as *j* <= p_1 and QS_j is not empty, steps are repeated from step 4 to step 11.
11. End.

Algorithm 3 - Algorithm explaining the steps involved in preference-based clustering for QoS data

For preference-based clustering, each attribute in the vector is independently controlled based on the preference value entered by the user. Once the user has prioritized this order, the attributes get clustered in that order iteratively. Each time the results of clustering is presented to the user, they have the option of selecting one or more clusters which will form the basis for the search space for the next attribute in order.

3.4.4 Weighted Clustering approach

As seen in section 3.4.3, the user ends up with a set of desired services which fit their bill of requirements. This process, although construed as an intuitive method with time-efficient performance, was more time consuming on the user's end due to its multi-level nature. Weight based clustering on the other hand, with a few minor compromises, proved to an effective solder between vector-based and preference-based clustering. The steps followed in the weighted clustering are as follows:

1. Let $QS = \{Q_1, Q_2, \ldots, Q_n\}$ be a set of n QoS vectors described by p interval variables.

2. Each QoS vector Q_i $(i=1, 2, \ldots, n)$ is represented as $x_i = (x_i^1, x_i^2, \ldots, x_i^p)$

   where,

   $x_i^y = [a_i^y, b_i^y]$ $(j=1,2,\ldots,p)$ represent the start and end points of the interval values.

3. If **weighted clustering** method is chosen,

   then

   A weight w_b is assigned to each attribute where $0 < w < 1$ and $b = (1,2,\ldots, n)$ correspond to each attribute.

4. The user is asked to input the number of clusters k.

5. Random k mean values are chosen in the set QS $(m_1, m_2, \ldots, m_k)$.

6. Each interval object is assigned to its closest mean using city block distance measurement for symbolic interval objects.

   $S_i^{(t)} = \{ x_j \mid\mid d(x_j, m_r^{(t)}) \mid\mid <= \mid\mid d(x_j, m_{r*}^{(t)}) \}$

   For all $r* = 1, 2, \ldots, k$.

   Where, $d(x_j, m_r) = \sum_{y=1}^{p} \varphi (x_i^y , m_r^y)$ such that,

   For **weighted clustering** method:

   $\varphi (x_i^y , m_r^y) = |w_1 (a_j^y) - w_2 (\alpha_j^y)| + | w_1(b_j^y) - w_2 (\beta_j^y) |$

   assuming w_1 is the weight assigned to x_j i.e. aj,bj and w_2 is the weight assigned to m_r i.e. α_j,β_j.

7. Now, the new mean of the centroid of the observations in the cluster is calculated:

   $m_r(t+1) = 1 / (S_i(t) \sum x(j) \in S_i (t) x_j$

8. Step 6 and step 7 are repeated until convergence has reached and all objects belong to a certain cluster.

9. The result of S_i clusters is presented to the requestor with related information such as size and prototype.

10. The requestor may then make an informed decision and choose *h (0 < h <=S_i)* clusters and then continue with step 12. However, if *j > 1*, the user may choose to go back to the previous level to change a previous decision. In this case, *j* is set to *j*-- and the user is taken back to step 5.

11. *j* is set to *j*++ and QS_j = {Q_i | Q_i € the *h* selected clusters). As long as QS_j is not empty, steps are repeated from step 4 to step 11.

12. End.

Algorithm 4 - Algorithm explaining the steps involved in weighted clustering for QoS data

In the weighted clustering process, the user is asked to assign a weight to each attribute according to its importance level. Although the clustering is performed on the whole vector, compared with the vector-based clustering, the natural groupings on those important attributes are more likely to be revealed due to the higher weights assigned to them. It keeps the same level of efficiency as the vector-based clustering process, whereas avoid the problem of its inability of identifying the pattern for indistinguishable dataset.

3.5 Chapter Summary

In conclusion, this chapter presents the framework of the Web Service selection tool implemented; the algorithms which substantiate its functions, namely – vector-based clustering approach, preference-based clustering approach and weighted clustering approach. Having established that K-means was the most adaptive algorithm to use with QoS-based, symbolic interval data type, use cases were described demonstrating the need to augment from vector-based approach to more interactive and efficient methods like preference-based and weighted

clustering. On the basis of the algorithms described, the service selection tool was implemented complete with a dataset repository and a visualization tool. The chapter was concluded with screen captures of the service selection tool to point indicate the steps and workings of the browsing tool.

CHAPTER 4

EXPERIMENTS AND PERFORMANCE EVALUATION

4.1 Overview

This chapter illustrates the experiments and the usability studies carried out to analyse the performance of the browsing mechanism proposed and evaluate its viability. Before the experiments are carried out, the datasets which were generated for the purpose of this study will be explained in detail along with the data distribution and cases used to simulate these datasets. Following this, the implementation of the tool and the programming sequence will be explained.

The evaluation of the QoS based browsing tool was conducted in three segments – evaluation of the accuracy of three clustering processes, evaluation of their efficiencies and evaluation based on the usability study. The evaluation on and the system efficiency demonstrates the time efficacy and the performance of each of clustering approaches in comparison with each other based on certain metrics. The evaluation based on the simulation scenarios and use cases was conducted to demonstrate the ease and clarity with which preference-based and weight-based approaches trump over the vector-based approach in terms of identifying relevant services. Finally, a user study was conducted to test the usability of the implemented browsing mechanism and to evaluate the real-time results recorded.

4.2 QoS Data Generation

The QWS (Quality of Web Service) dataset is one of the more well known datasets which are used by Web Service researchers to conduct experiments for their research efforts. There are approximately 5000 Web Services collected for this dataset with a subset of 365 real Web Service implementations as of March 2008 (Al-Masri & Mahmoud, 2008). These services were

collected using the Web Service Crawler Engine (WSCE) which crawls over a variety of public sources including the UDDI registries, search engines, and service portals. This dataset formed the basis of the experiments conducted in Interactive QoS browsing for Web Services (Sambamoorthy, QoS Browsing for Web Service Selection, 2009).

A search engine like *Seekda* (Seekda corporate author, 2007) is also a popular portal for searching for real Web Services and relevant measures attached to them. However, it was concluded that neither of the existing sources met with all the dataset requirements of the experiments conducted in this thesis since they consisted of no real data patterns. For the purpose of this thesis it was considered important to simulate datasets which reflected real datasets and also incorporated various use cases and scenarios. Since this is a comparative study for various types of Web Service clustering methods, it was considered important to have a variety of datasets for a keener performance evaluation.

In order to simulate these datasets, MATLAB® (MATrix LABoratory) (MathWorks), which is a MathWorks ™ product, was used. MATLAB is a widely recognized tool for numerical computing and statistical analysis. The functions used for data simulation were – *normrnd (mu, sigma, m, n)* and *sort (x, y)*; where, *normrnd* generates random numbers following the normal distribution with mean parameter *mu (μ)* and standard deviation parameter *sigma (σ)* in an *mxn* array; *sort* is a function used to used scalar *x* in *y* columns in ascending or descending order. These functions were chosen because the datasets were determined to best have a multivariate normal distribution.

There were a total of 24 datasets which were officially used to the purpose of performance analysis and usability study. The datasets were generated in MATLAB and

exported to a Microsoft Excel file from where it was saved as a tab delimited text file which was directly fed into the Web Service selection program.

4.2.1 Data simulation scenarios and Input Parameters

The following are the simulation scenarios and the use-cases considered when generating the datasets with the following cluster distributions:

- Distinct clusters: These datasets were generated keeping in mind data where the cluster would be clearly distinct and far apart from each other, with a large difference in mean values and relatively smaller deviations.

  Datasets with distinct cluster groupings were generated with variations like different data size (with 300, 3000 or 30,000 objects in each), different number of attributes (such as 3, 4, 6 or 9 attributes) and different number of sub-cluster groupings within each cluster grouping (such as 1, 3, 4, 6 etc).
- Indistinct clusters: These datasets were generated with indistinct, overlapping cluster groupings with smaller difference in mean values and larger deviations. These datasets also had variations with respect to size and number of attributes.
  - Distinct clusters with indistinct sub-clusters: These sample datasets were generated with three main clusters within which overlapping sub clusters were created with large standard deviations.
  - Partially overlapping clusters: These were datasets where some of the attributes had overlapping cluster groupings while others had distinctly aligned cluster groupings.
  - Completely overlapping clusters: These datasets were generated with completely overlapping clusters with almost continuous mean-values and large deviations.

- Redundant clusters: These datasets was generated with the view that more than one dataset could contain one or more attributes having duplicate values present in more than one QoS vectors which may fall under different clusters and produce different results for different clustering methods.
- Different number of clusters for different attributes: These datasets were simulated with different number of clusters within each attribute.

4.3 Implementation of service selection browsing tool

The implementation of this tool was done in C# language using Microsoft Visual Studio. It was chosen as the most suitable platform due to its object oriented support and graphical user interface (GUI) design support. Along with implementing each of the algorithms cohesively into a sound GUI, it was also considered important to implement a visualization feature to present the user with a visual of the distribution of clusters and to simply the process of choosing the cluster(s) significantly.

The structure of the tool maybe well explained with the help of a few screen captures of the tool. For the purpose of this representation, the dataset chosen had a three attributes – Cost, Response Time and Reliability with pre-defined cluster groupings of 3 distinct clusters and 3 distinct sub-clusters each.

1. A user is first presented with the data selection menu where the user can browse all the datasets present in the repository which matches the required format of the dataset.
2. If a user wishes to cluster the services using the vector-based clustering approach, the desired dataset is selected and the preference order of the attributes is, by default, uniformed to 1.

3. If a user wishes to cluster the services using the preference-based clustering approach, the desired dataset is selected and the order of preference is chosen from the drop down menu as required.
4. If a user wishes to cluster the services using the weighted clustering approach, the desired dataset is selected and, under the weighted clustering tab, the order of weights may be adjusted as desired such that the sum total of the weights is 1.

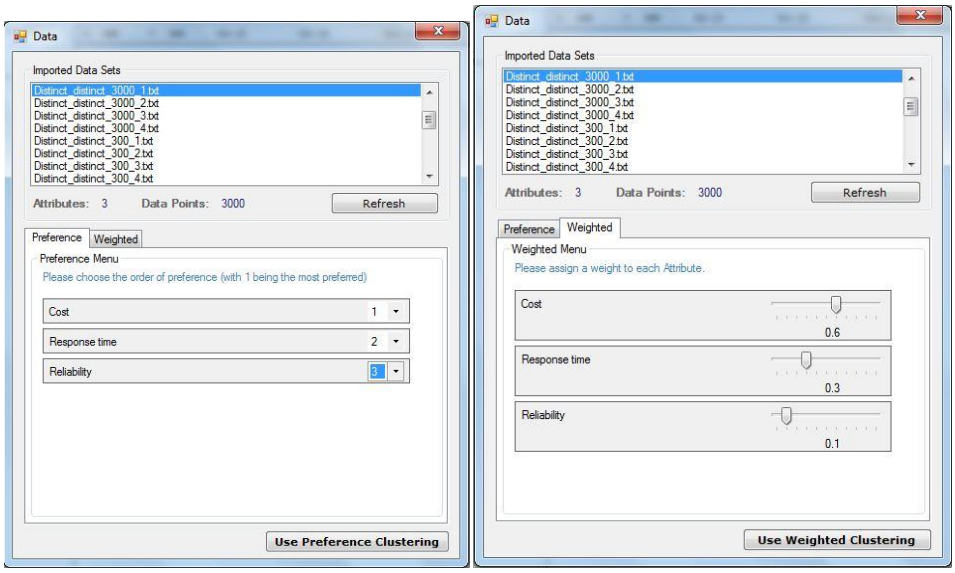

Figure 3 - User menu for clustering (preference-based and weighted)

- If the user has chosen to opt for vector-based clustering, the user menu for clustering will open up, prompting them to enter the number of clusters they require and to calculate the clusters. As soon as the results are clustered, the user now has the option to select any number of clusters and also visualize them on the plotting feature implemented. Every cluster is represented with its number, size and centroid interval values and clicking on it plots its distribution of the visualization feature.

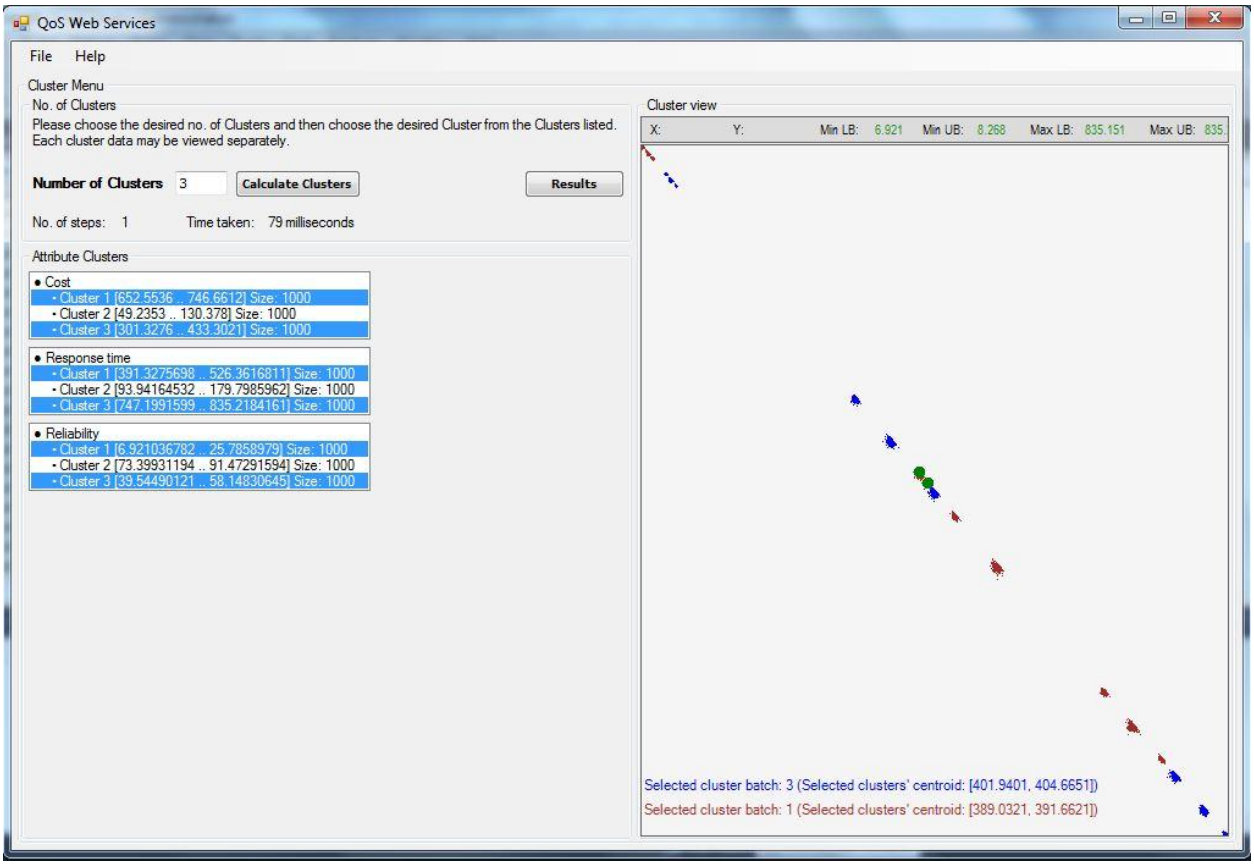

Figure 4 - Vector-based clustering with two selected clusters

- The same process is also followed for weighted clustering, except for the order in which the attributes are presented – which is in descending order of the weights assigned to each attribute.

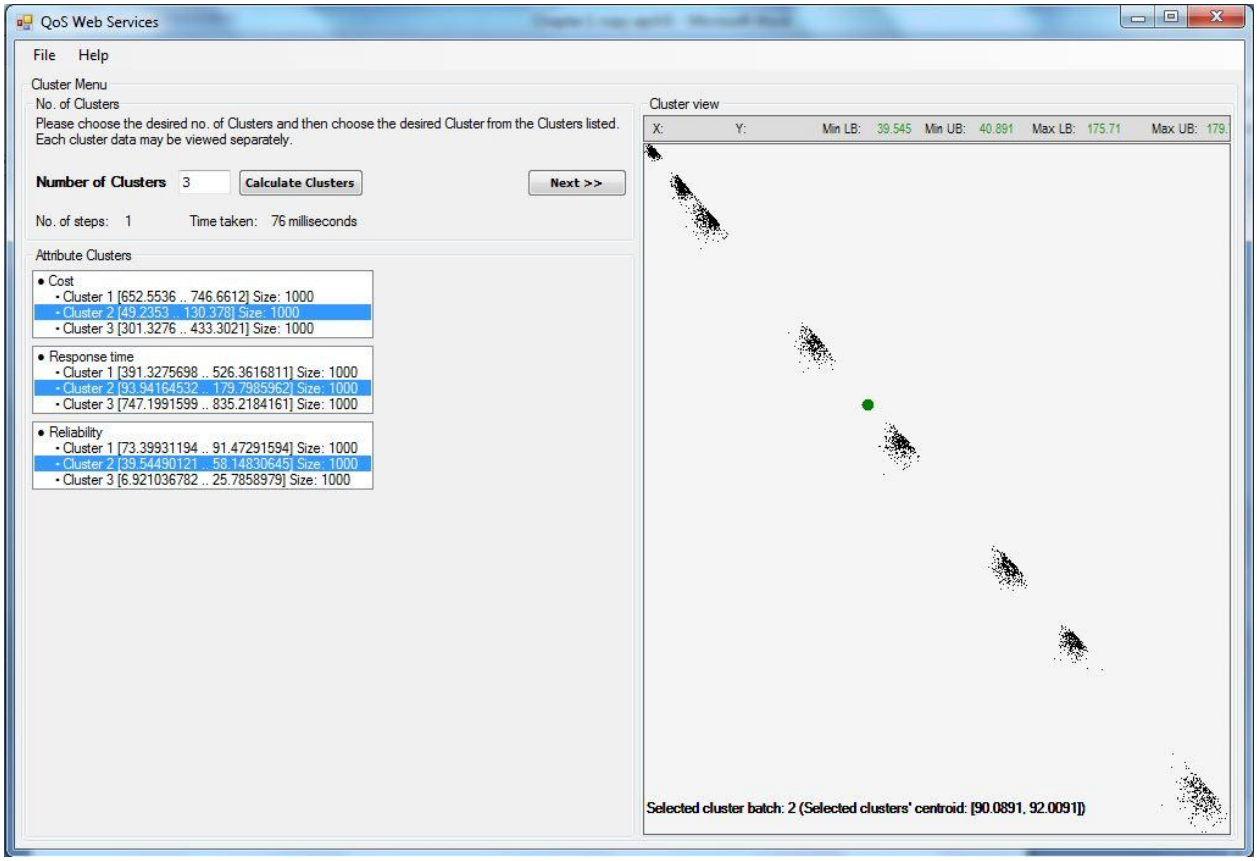

Figure 5 - Weighted clustering with one selected cluster

- For preference-based approach, the user will be presented with each attribute in the order of preference where the user will only be prompted to enter the number of clusters they want to see for current attribute. When the user has selected the number of clusters they wish to proceed with for the current attribute, they may go to the next attribute until all the attributes have been exhausted.

- After each cluster has been calculated, the user is also provided with the time taken to calculate the clusters and the total number of clicks or steps acquired. The user may choose a different number of clusters at any time and re-calculate the clusters.

Finally, when the user gets to the results page, a summary of the service selection is presented along with the time taken (in milliseconds) and number of steps taken to reach the results (this includes any backtracking the user may have done).

4.4 Performance evaluation of QoS based Web Service selection methods

As discussed in section 4.1, three groups of experiments were carried out to determine the performance of each of the browsing mechanisms developed namely – vector-based clustering approach, preference-based clustering approach and weight-based clustering approach.

The evaluations have been divided into three parts – the first was to evaluate the accuracy and effectiveness of the browsing methods with different dataset scenarios; the second experiment was carried out to evaluate the efficiency of different browsing methods; the third one was a user study carried out to draw a comparison among these three approaches regarding their performance from users' perspectives.

4.4.1 Experiments conducted based on accuracy

This experiment was conducted to draw a comparison between the clustering results produced by vector-based, preference-based and weighted clustering methods. In order to analyze the accuracy of these methods, different data distribution patterns of the datasets should be considered. The purpose of generating these different datasets was to investigate the effectiveness of the three clustering processes when being used on data following different distribution patterns, e.g. multi-dimensional data vectors naturally forming into groups, datasets with natural groupings found in one or a few but not all dimensions, datasets with data in each dimension formed into a small number of groups whereas as a vector scattered into a large number of sparse groups due to the combinatory effect, etc. As we all know, clustering

algorithms may not work well for all data distribution patterns, e.g. data following a uniformed distribution pattern. Even when there is a certain level of natural groupings found in the dataset, depending on the actual pattern, one clustering approach may work better than the other.

For a multi-dimensional dataset, the chance to identify natural groupings among the data vectors is very low, however, the chance for data on individual dimensions to have some natural groupings is much higher, and definitely, it is possible that we cannot find any natural grouping patterns on some dimensions. There would be many possible data distribution patterns to follow. In our experiment, to simplify the testing scenarios and emphasize more on scenarios when clustering works, we generate the datasets such that data in one dimension can be grouped into a small number of clusters, whereas vectors may or may not be easily grouped into a reasonable number of clusters. Below, we use two sample datasets to illustrate the comparison results.In order to analyze the results, two different types of sample datasets were considered:

Dataset 1: This dataset was generated in accordance with the distribution of all the other datasets generated for this study i.e. it followed a multivariate normal distribution.

Clust 1 (1000) (330,330,340)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 55$	$\sigma_{11} = 2.0$	$\mu_{11} = 100$	$\sigma_{11} = 2.0$	$\mu_{12} = 10$	$\sigma_{11} = 1.0$
	$\mu_{12} = 80$	$\sigma_{12} = 2.0$	$\mu_{12} = 140$	$\sigma_{12} = 1.8$	$\mu_{13} = 17$	$\sigma_{12} = 1.2$
	$\mu_{13} = 125$	$\sigma_{13} = 2.0$	$\mu_{13} = 170$	$\sigma_{13} = 3.0$	$\mu_{14} = 23$	$\sigma_{13} = 0.9$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 310$	$\sigma_{21} = 3.0$	$\mu_{21} = 400$	$\sigma_{21} = 3.0$	$\mu_{22} = 42$	$\sigma_{21} = 0.8$
	$\mu_{22} = 360$	$\sigma_{22} = 3.6$	$\mu_{22} = 450$	$\sigma_{22} = 2.4$	$\mu_{23} = 49$	$\sigma_{22} = 1.3$
	$\mu_{23} = 420$	$\sigma_{23} = 4.0$	$\mu_{23} = 510$	$\sigma_{23} = 4.0$	$\mu_{24} = 56$	$\sigma_{23} = 0.6$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 660$	$\sigma_{31} = 2.5$	$\mu_{31} = 760$	$\sigma_{31} = 3.6$	$\mu_{31} = 77$	$\sigma_{31} = 1.2$
	$\mu_{32} = 700$	$\sigma_{32} = 3.9$	$\mu_{32} = 800$	$\sigma_{32} = 3.0$	$\mu_{32} = 83$	$\sigma_{32} = 1.0$
	$\mu_{33} = 740$	$\sigma_{33} = 2.0$	$\mu_{33} = 830$	$\sigma_{33} = 2.0$	$\mu_{33} = 89$	$\sigma_{33} = 0.9$

Table 2 - Sample *dataset1* with distinct clusters

Table 2 shows all the mean values and the deviations used to generate the dataset1. This dataset was generated with three attributes, namely cost, response time and reliability; 3 distinct

clusters, each having 3 distinct sub-clusters within it. Each cluster had 1000 data objects and each of the three sub-clusters had 330, 330 and 340 data objects respectively. In this dataset, the clusters for all three attributes as a whole are consistent with the clusters for each individual attribute.

Dataset 2: In this dataset, the mean and the variance values for each attribute are the same as those in dataset1, however, the vectors on each dimension take values from different clusters randomly and inconsistently. With the purpose of clearly illustrating this dataset, the values of some of the QoS vectors within the first cluster are shown as follows:

Vector	Cost		Response Time		Reliability	
V_{11}	51.4106	52.0168	827.5879	828.9999	15.01601085	16.40055445
V_{12}	53.5154	56.6808	513.1627	515.6502	90.25130957	90.44658655
V_{13}	52.8768	53.2239	507.8527	513.4168	82.1930349	83.2157025
V_{14}	55.2002	59.7009	139.5173	142.0652	82.91533743	83.36551409
V_{15}	53.7688	53.9109	171.5768	174.6953	87.82043318	88.56912676
V_{21}	76.8459	80.6151	171.5768	174.6953	41.55582789	42.37364703
V_{22}	77.4858	81.0159	138.9387	140.8081	82.18391199	82.32017882
V_{23}	78.2691	80.564	168.6518	175.3955	48.76342773	49.48131977
V_{24}	79.6469	80.067	164.92	174.5698	82.36917166	83.43738986
V_{25}	77.3326	81.5828	97.1511	101.7728	89.31024055	90.43079527
V_{31}	125.1599	125.9433	511.4156	513.3135	22.96826917	24.58403481
V_{32}	122.5743	123.103	400.375	402.3721	9.453811319	10.27951506
V_{33}	125.1324	125.823	801.6929	804.2689	56.21122461	56.35002629
V_{34}	126.3047	126.354	794.5096	804.0235	48.82107482	49.93053083
V_{35}	125.6541	126.7155	398.6144	399.1517	89.15412145	89.38059961

Table 3 - Sample values of first cluster within *dataset2*

From Table 3, we could see that on the dimension of cost, the data distributions are the same as in dataset1, which clearly shows 3 groups of data, whereas on the other two dimensions, for vectors with cost values falling into a same cluster, their response time or reliability values may come from all different clusters.

4.4.1.1 Vector-based clustering applied to *dataset1* and *dataset2*

With the purpose of observing how vector-based clustering fairs with the above dataset cases, it was applied to both the datasets. The number of clusters (k-value) were limited to 3 since the dataset was designed to produce best results with k = 3. The results of the experiment are as follows:

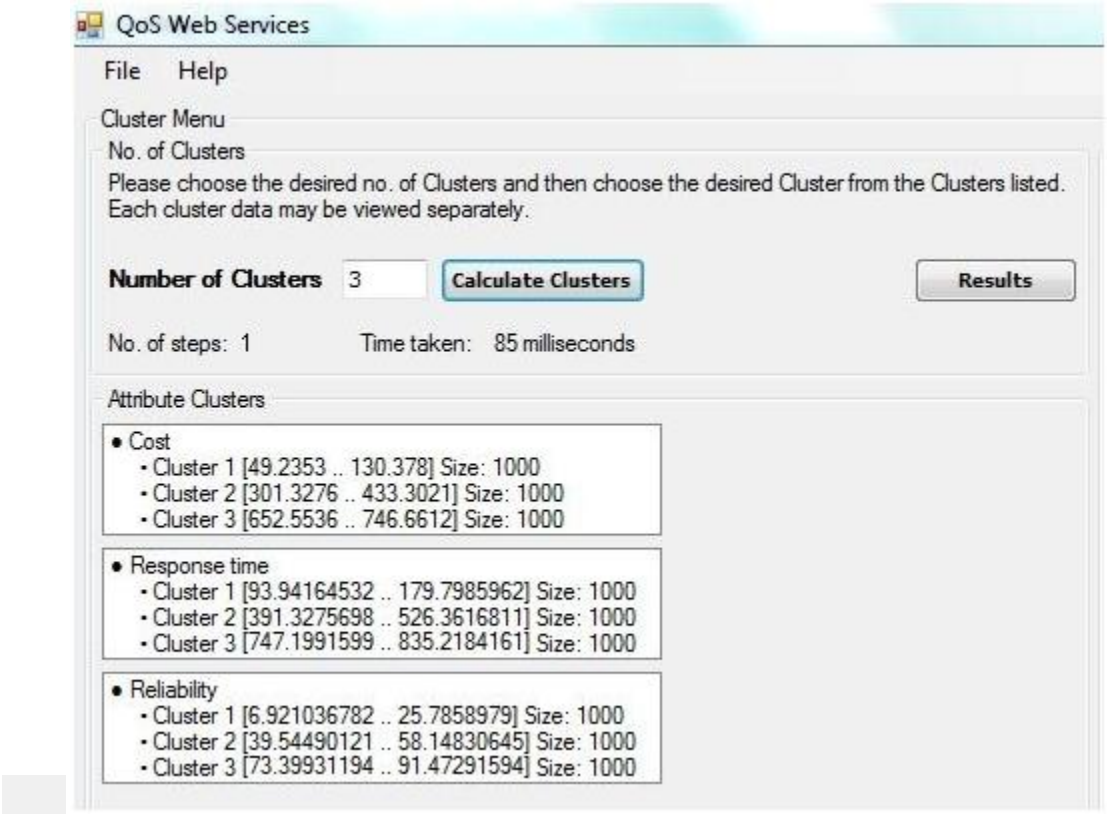

Figure 6 - Vector-based clustering applied to *dataset1*

In Figure 6, vector-based clustering is applied to dataset1 where the clusters predictably fall into cohesive groups since all the attributes had vectors with values which were aligned to be in the same clusters.

With the purpose of observing how vector-based clustering fairs with the above dataset cases, it was applied to both the datasets. With the purpose of observing how vector-based clustering fairs with the above dataset cases, it was applied to both the datasets. It is important to

analyze the k-value in this case since, in this case, k <=27 will give clear results of clustering but for a larger k value, the clusters will be much smaller and indistinguishable thus losing the combinatory effect of clustering. For example, had there been 5 groupings of value for each attribute, the apt k value would be 125 which would be too large and cumbersome for the user thus increasing its cognitive overload.

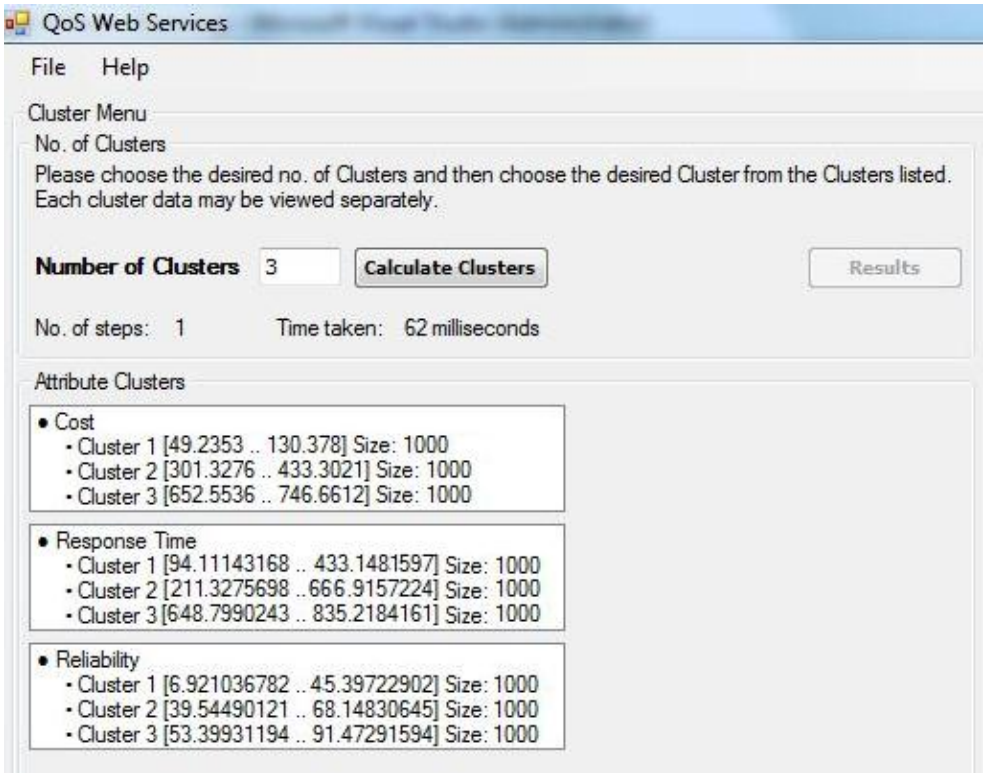

Figure 7 - Vector-based clustering applied to *dataset2*

In Figure 7 the inaccuracy of vector-based clustering is depicted with incorrect and overlapping cluster values for *response time* and *reliability*. It may be seen that clusters 1, 2 and 3 for both response time and reliability have overlapping values, thus providing indistinguishable clusters. This proves to be a significant drawback when using vector-based clustering. Since it treats all the attribute values of a Web Service as a whole within a vector, it may not always be possible to cluster the values on a fair basis since their values may fall into different clusters. In cases such as this, the requestors may get confused and discontented with the choices presented

to them and ultimately unable to make a decision or be able to proceed with the selection process.

4.4.1.2 Preference-based clustering applied to *dataset1* and *dataset2*

Preference-based clustering considers each attribute individually in the order of the requestor's preference. This allows similar values within each attribute to be grouped together, irrespective of their order within each vector. The sample dataset cases were also tested with preference-based clustering to yield the following results:

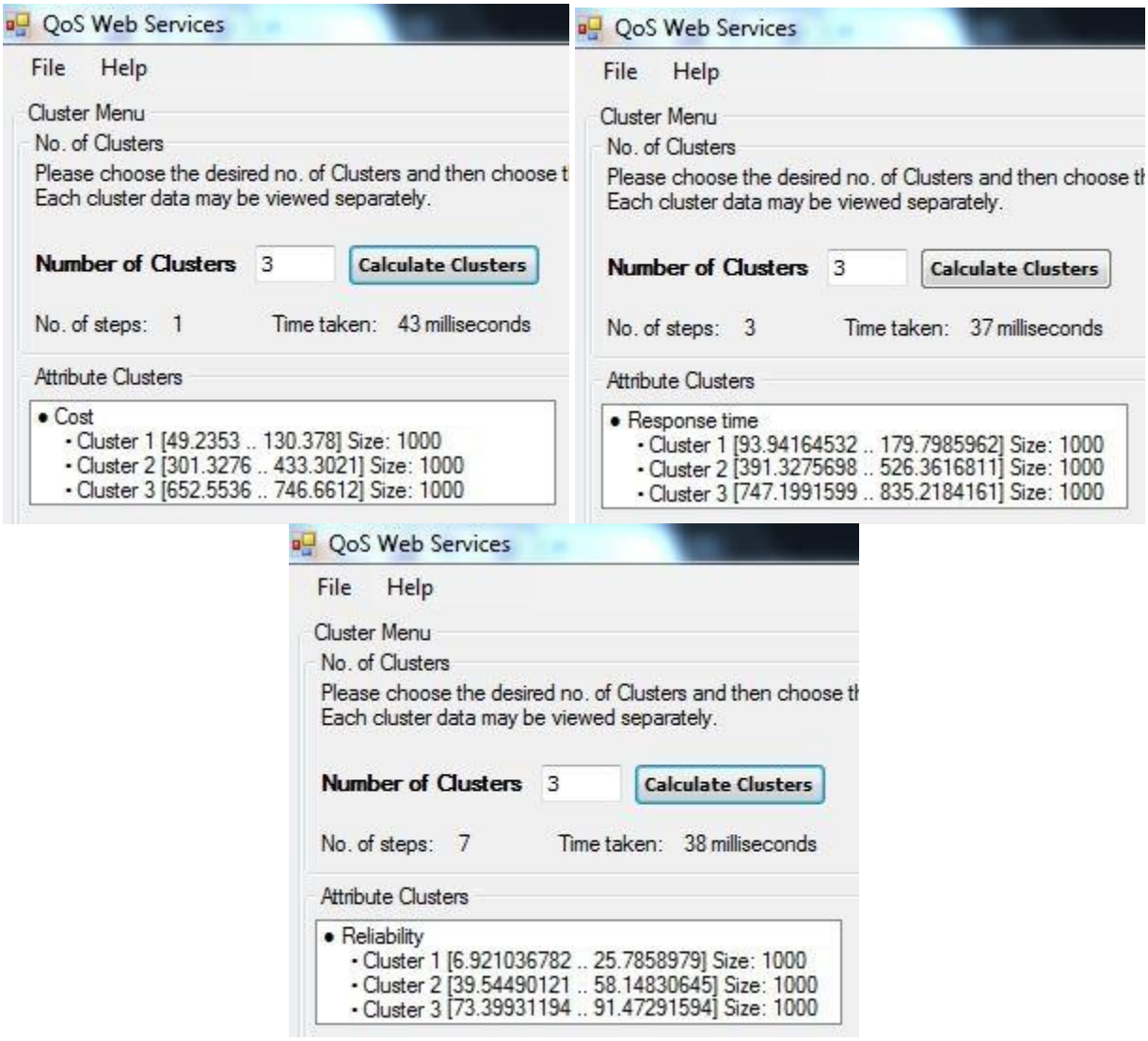

Figure 8 - Preference-based clustering applied to *dataset1*

In Figure 8Figure 8, preference based clustering was applied to *dataset1* with *cost* having the highest preference (1), *response time* with a lower preference (2) and *reliability* with the lowest preference (3) as seen in clockwise order from top left in Figure 8. In order to show the distribution of clustering, all three clusters were chosen to go to the next attribute.

Next, preference-based clustering was tested on *dataset2* and the results were observed as follows:

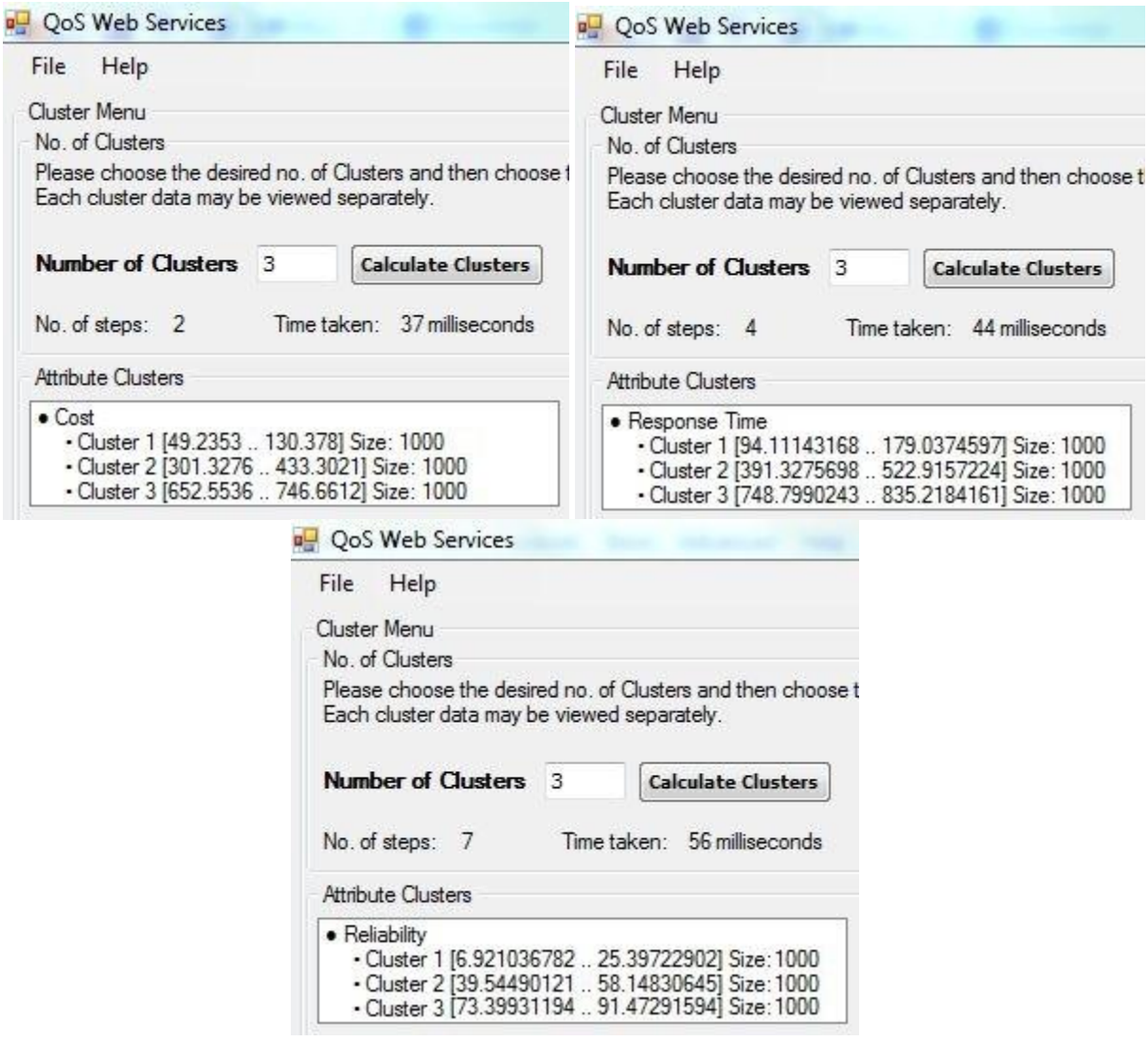

Figure 9 - Preference-based clustering applied to *dataset2*

In Figure 9, it is observed that the clusters are able to naturally fall into the groups based on their similarities. The clustering process is not hampered here by ties of the QoS vectors and this may be counted as one of the significant advantages of using preference-based clustering. As it may be seen, *response time* and *reliability* have clear and distinguishable clusters despite the randomized order within the vectors of *dataset2*.

4.4.1.3 Weighted clustering applied to *dataset1* and *dataset2*

Weighted clustering considers each QoS vector in the dataset rather than each attribute value for clustering; however, the clustering is heavily dependent on the weights assigned to each attribute. These weights are assigned to each attribute in the order of attribute preference where greater weights signify higher importance. Various weight combinations were tried on each attribute and tested on both *dataset1* and *dataset2*. The results from each case are as follows:

Case I:

Cost = 0.3 (average importance),

Response time = 0.6 (most important),

Reliability = 0.1 (low importance).

The results of the clustering were observed as follows:

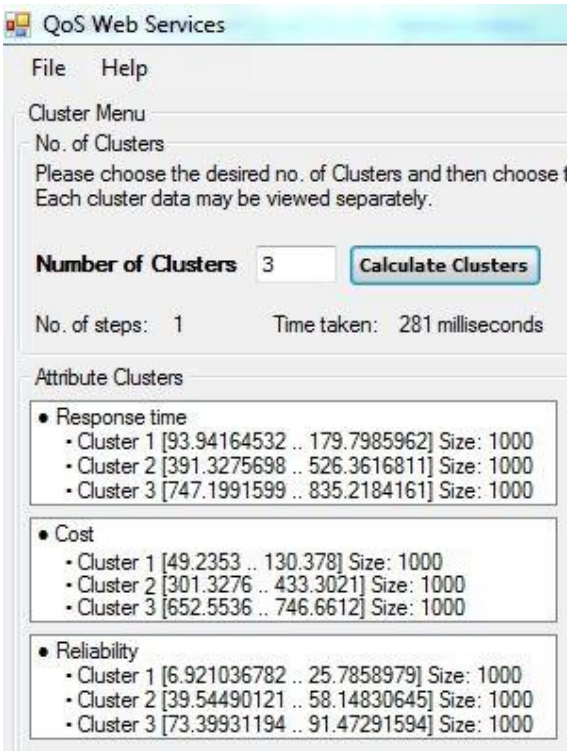

Figure 10 - Weighted clustering applied to *dataset1*

In Figure 10 above, the results of weighted clustering for *dataset1* have been shown. The clustering results are shown in the order of the weights where *response time* is is the dominant attribute. All the clusters are clearly and distinctly defined in this case owing to the nature of *dataset1*. The results of the same when performed on *dataset2* are as follows:

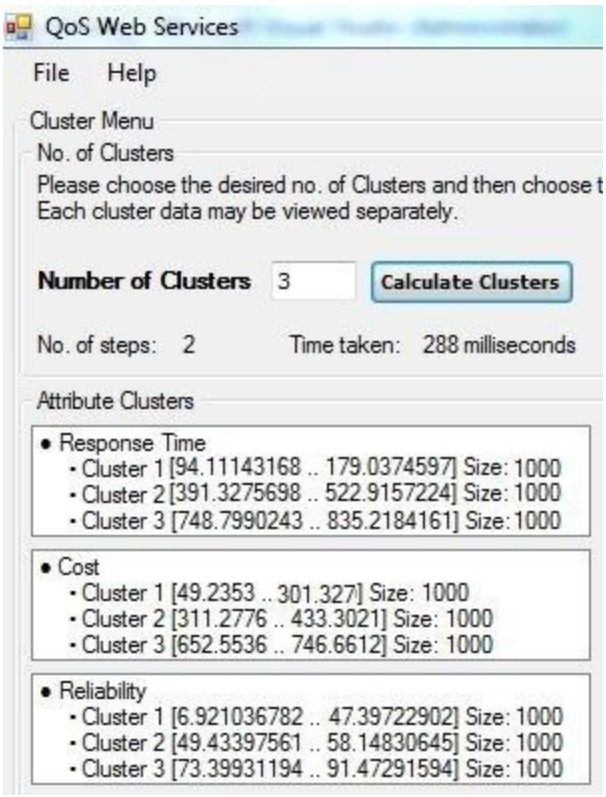

Figure 11 - Weighted clustering applied to *dataset2*

In Figure 11, the results of weighted clustering tested with dataset 2 have been shown. Since *response time* was the dominant attribute, its clusters are clear and distinct. *Reliability* was observed to have some anomalies in its result, which most likely occured due to the lower weights assigned to it and the nature of *dataset2*. The boundaries of the clusters have been compromised since reliability had the least importance and thus it was not given as much importance .

Case II:

Cost = 0.2 (low importance),

Response time = 0.8 (most important),

Reliability = 0.2 (low importance); (*cost* and *reliability* have same importance).

When Case II was applied to *dataset1*, the results were identical to Figure 10 where reliability is the dominant attribute and owing to the distinctly defined nature of the clusters within *dataset1*, each attribute is sparingly clustered.

The results obtained from repeating this exercise on *dataset2* is similar to Figure 11; the only difference in this case was that both cost and reliability had some disparity in their clusters owing to the nature of the dataset and the low weights assigned to both *cost* and *reliability*.

Case III:

Cost = 0.4 (high importance),
Response time = 0.2 (low importance),
Reliability = 0.3 (moderate importance);

In Case III, *response time* is assigned the lowest importance with the least weight. When applied to *dataset1*, the results are organized in the order of the weights. The results for *dataset1* are again identical to the results observed for Case I and Case II, as seen in Figure 10. The results obtained from applying this case to *dataset2* are as follows:

The results acquired in this case favor *cost* owing to its high importance. Distinctly distributed clusters may be observed for both *cost* and *reliability*. There are certain anomalies in the clusters generated for response-time owing to its low importance. The values within *response time* are thus forced to tie to the vectors that have already fallen into clusters owing to the weights of the more important attributes.

4.4.1.4 Inference from experiments conducted based on dataset scenarios

Having observed and analysed the results obtained by each of the clustering methods when tested on two sample datasets representing two case scenarios, it was noted that preference-based clustering delivered the most accurate clusters. The following are the key points surmised from this experiment:

- Vector-based clustering shows a significant disadvantage when the attribute values within a vector tend to fall under different clusters. The same can also be said for redundant values since it is possible for two services to have the same *cost* but different *response time* and *reliability* values.
- Preference-based clustering succeeds in being the most accurate clustering method, providing clear and distinct clusters for both the case scenarios presented in the experiment. In this case, the attributes are clustered step by step and based on the user's current selection; the search space for clustering the next attribute is chosen. This is an intuitive and precision centric experience which countermands the constraints faced by vector-based approaches.
- Weighted clustering helps relax the constraints of vector-based clustering in that it allows for each attribute to have a certain weight attached to it which determines its importance while clustering. Where it succeeds in time efficiency, it also fails to be completely accurate. The dominant attribute (with the largest weight), which in effect is most important for the user does provide clear and distinct cluster groupings, however the same cannot be said for the less important attributes although in effect, that does not hamper the user experience as much since these attributes had low importance assigned to them to begin with.

4.4.2 Efficiency of the proposed Browsing Methods through Experiments

In order to test the efficiency and compare the three browsing methods with each other, each of the browsing methods were compared by the time taken to cluster all the services and yield the final result with respect to the different sizes of the datasets used, the different number of attributes in each of the datasets used, and different number of clusters (k-value) requested.

4.4.2.1 Comparison module for time consumed vs. dataset size

This experiment was conducted with a view of serving two purposes – establishing the time consumption of each of the clustering methods with respect to varying dataset sizes and establishin the effect different distribution of cluster groupings may have on the time consumption. The reason behind this exercise was to see if a dataset with clear and distinctly identified cluster groupings took less time than clustering a dataset with scattered and haphazardly distributed values. The time taken is the time taken by the system to calcute the clusters when the program was allowed to run in the debug-free mode to be as CPU cost-effective as possible. The results are as follows:

S.No	Clustering Type	Cluster distribution	Dataset Size	Number of Attributes	K-Value	Time (ms)
1	***Vector-based***	Distinct	***300***	3	3	41
2	"	"	***3000***	"	"	320
3	"	"	***30000***	"	"	16154
4	"	Indistinct	***300***	"	"	39
5	"	"	***3000***	"	"	401
6	"	"	***30000***	"	"	16136
1	***Preference-based***	Distinct	***300***	3	3/per attr	201
2	"	"	***3000***	"	"	426
3	"	"	***30000***	"	"	22689
4	"	Indistinct	***300***	"	"	203
5	"	"	***3000***	"	"	464
6	"	"	***30000***	"	"	22614
1	***Weighted***	Distinct	***300***	3	3/per attr	53
2	"	"	***3000***	"	"	302
3	"	"	***30000***	"	"	15535
4	"	Indistinct	***300***	"	"	49
5	"	"	***3000***	"	"	361
6	"	"	***30000***	"	"	16043

Table 4 - Time vs. dataset size and cluster grouping within dataset

The experiment has been designed such that the type of dataset cluster distribution, the total number of attributes in the dataset and the number of clusters have been kept constant, while the size of the dataset has been kept as a variable. In Table 4, it can be clearly seen by the time taken by each of the clustering methods, be it vector-based method, preference-based method or weighted method, that the type of cluster distribution within the dataset does not hamper or improve the time taken to perform the clustering. The difference between the time taken to cluster distinctly or indistinctly distributed datasets is none to negligible. Therefore, it is established that the distribution of the data objects within the dataset do not affect its performance and therefore, for the sake of simplicity and standardization, only datasets simulated with distinct clusters will be used for testing purposes.

The time taken by each of the clustering methods increases exponentially as the dataset sizes are increased from 300 to 3000 to 30,000. However, since preference-based clustering approach is a step by step approach, the cumulative time taken is greater than the time taken by vector-based or weighted approach which take comparateively the same amount of time for each of the datasets.

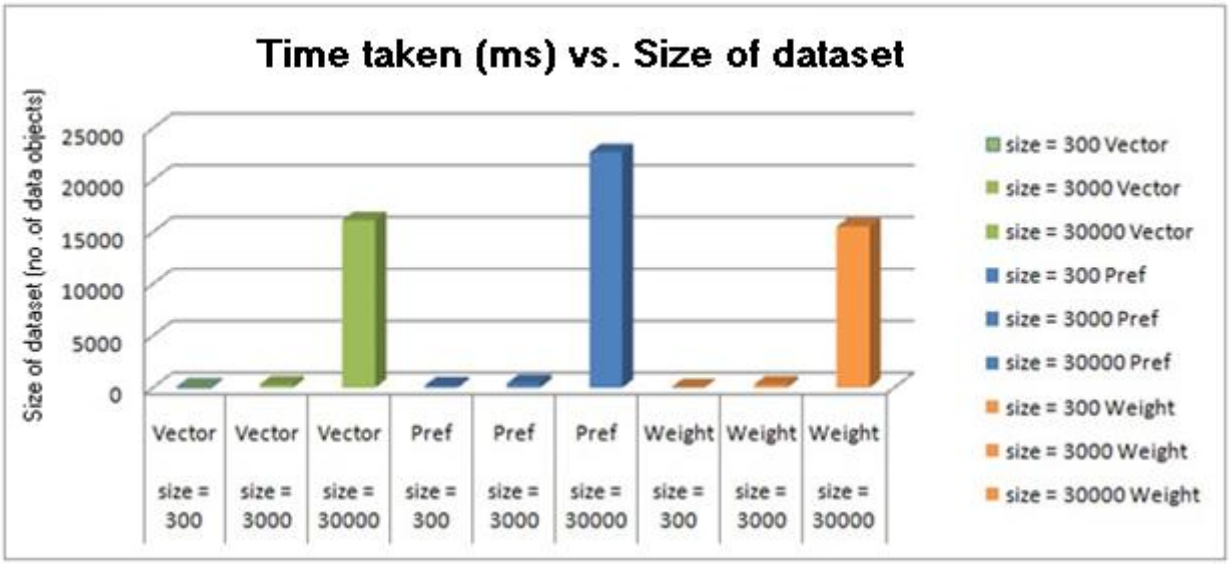

Figure 12 - Time taken for each clustering method vs. Size of dataset

In Figure 12, a bar graph has been drawn to compare the time-taken by each of the clustering methods for different dataset sizes. With this visual, it is clear that with larger dataset sizes, preference-based clustering tends to accumulate more time to yield the final result. Weighted clustering emerges as a winner in this case since it provides a preference-oriented result to some extent with a one-step, time-efficient approach.

4.4.2.2 Comparison module for time consumed vs. number of attributes

This experiment was conducted with a view of drawing a comparison between each of the three browsing methods keeping the number of attributes as a variable and, the dataset size

and the number of clusters constant while recording the time. The size of the dataset used was 3000 and the k-value was kept constant at 3 for each clustering method. When the numbers of attributes increase or decrease in a dataset it is essentially a size variation from a primary point of view. However, the calculation of clusters is dependent on the number of attributes and therefore, it is important to observe the time-consumption for each of the browsing methods when clustering datasets with varying number of attributes. The result from each of methods is listed as follows:

S.No	Clustering Type	Dataset Type	Dataset Size	Number of Attributes	K-Value	Time (ms)
1	***Vector-based***	Distinct	***3000***	**1**	3	332
2	"	"	"	**3**	"	442
3	"	"	"	**4**	"	538
4	"	"	"	**6**	"	898
5	"	"	"	**9**	"	1221
1	***Preference-based***	Distinct	***3000***	**1**	3/per attr	330
2	"	"	"	**3**	"	420
3	"	"	"	**4**	"	470
4	"	"	"	**6**	"	564
5	"	"	"	**9**	"	700
1	***Weighted***	Distinct	***3000***	**1**	3	338
2	"	"	"	**3**	"	440
3	"	"	"	**4**	"	535
4	"	"	"	**6**	"	848
5	"	"	"	**9**	"	1192

Table 5 - Number of attributes vs. time taken by each of the clustering methods

In Table 5, the time taken by each of the clustering methods to cluster datasets with different number of attributes has been illustrated. It is noted that the time taken to cluster each of the datasets increases linearly with increase in the number of attributes. It is also observed that preference-based clustering not only comes at par with the time-taken to cluster the datasets, but also decreases with increase in the number of attributes.

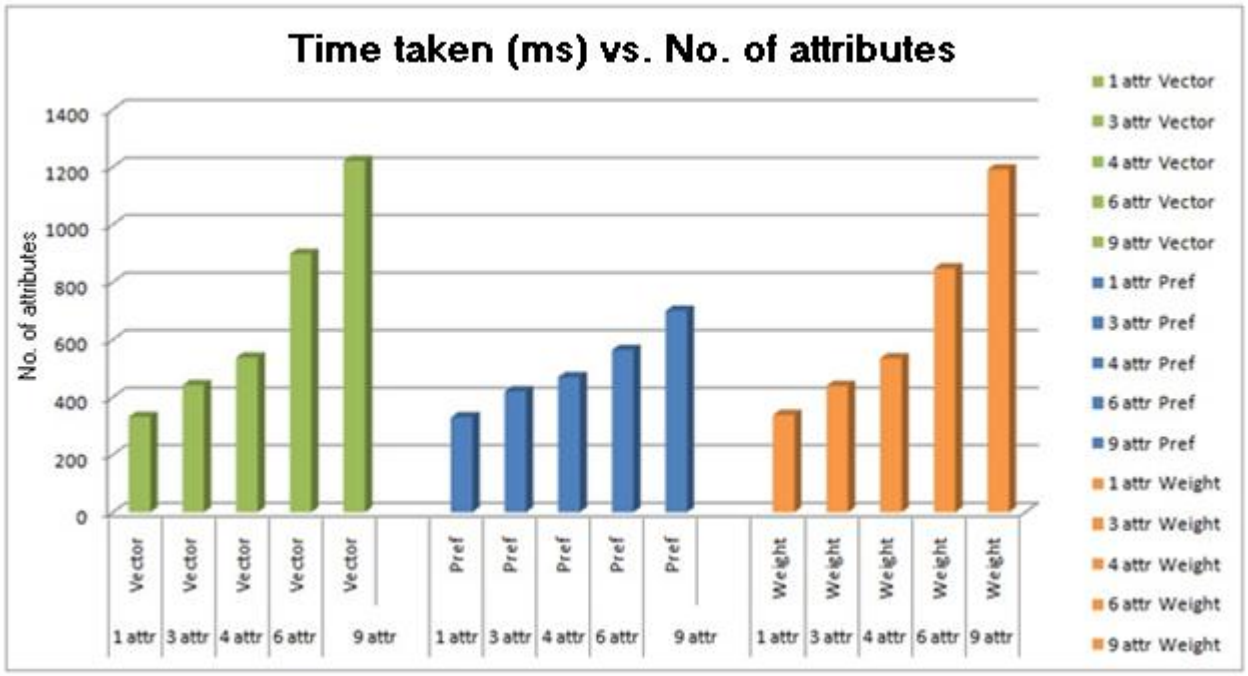

Figure 13 - Time taken for each clustering method vs. Number of attributes

In Figure 13, preference-based clustering is observed to be just as time efficient as vector-based and weighted clustering methods for lower number of attributes (1, 3 and 4); however for larger number of attributes, there is a smaller rise in time-taken for preference-based clustering when compared to the other two. Preference-based method clusters attributes one step at a time, where the clustering space for the next attribute(s) is defined by the selected clusters chosen in the previous step. Therefore, as the selection process progresses, the amount of data to be clustered reduces, thus taking less time to calculate the clusters.

Therefore in this case, preference-based clustering emerges as the most beneficial service selection method providing both accuracy and time efficacy with increase in number of attributes when compared to the other two clustering methods.

4.4.2.3 Comparison module for time consumed vs. number of clusters

This experiment was conducted to determine the effect on the total time taken to cluster the services for varying number of clusters. The dataset size and the number of attributes were kept constant at 3000 and 3 respectively. The results of the experiment are as follows:

S.No.	Clustering Type	Dataset Type	Dataset Size	Number of Attributes	K-Value	Time (ms)
1	***Vector-based***	Distinct	***3000***	3	**3**	421
2	"	"	"	"	**6**	523
3	"	"	"	"	**9**	679
4	"	"	"	"	**20**	1351
5	"	"	"	"	**30**	2363
6	"	"	"	"	**40**	3200
1	***Preference-based***	Distinct	***3000***	3	**3**	331
2	"	"	"	"	**6**	581
3	"	"	"	"	**9**	683
4	"	"	"	"	**20**	884
5	"	"	"	"	**30**	~1332
6	"	"	"	"	**40**	~1990
1	***Weighted***	Distinct	***3000***	3	**3**	412
2	"	"	"	"	**6**	551
3	"	"	"	"	**9**	651
4	"	"	"	"	**20**	1308
5	"	"	"	"	**30**	2492
6	"	"	"	"	**40**	3278

Table 6- K-value vs. time taken by each of the clustering methods

In Table 6, the time taken to yield the clusters for each of the k-values has been listed. As in section 4.3.2.2, it is noted once again, that the time taken to perform preference-based clustering has a progressively shallow gradient when compared to the time consumed by vector-based and weighted clustering. This may be attributed to the fact that as the k-values increase at each step, the selection is limited to one or a few clusters at each step which form the basis for clustering the next attribute(s).

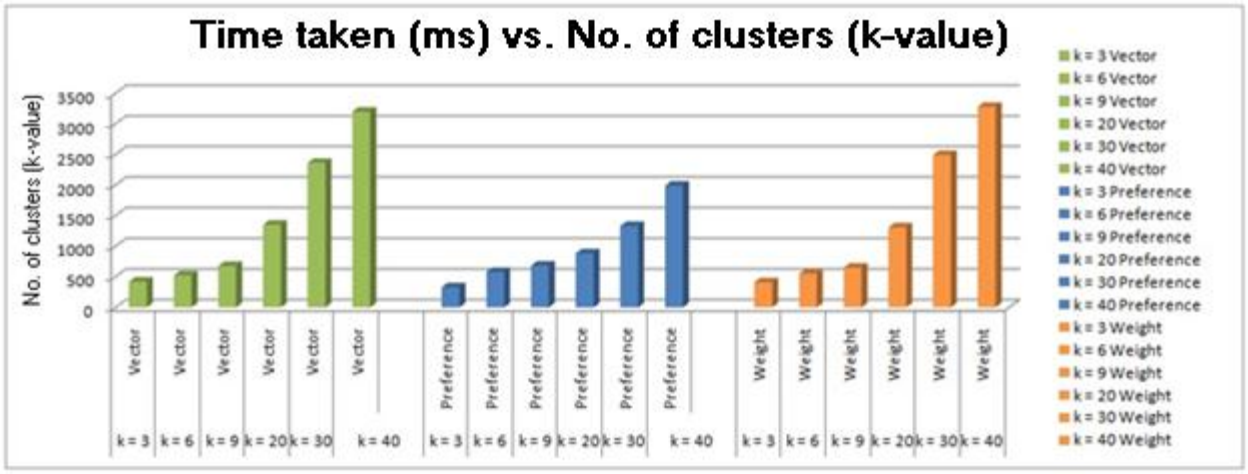

Figure 14 - Number of clusters (k-value) vs. Time taken for each clustering method

Figure 14 shows the bar-graph comparing the time-consumption of each clustering method with respect to the varying number of clusters. With increase in the number of clusters (per attribute in case of preference-based clustering), the increase in time is observed to be steeper in vector-based and weighted clustering when compared to preference-based clustering method. For example, if a requester decided to cluster the given dataset with three attributes with each attribute assigned to a preference level, then at the first step if the desired number of clusters k = 20, then the requester will be presented with 20 clusters for the first attribute. Each of these clusters will have a small number of services. In all realistic probability, the requestor will select a one or a few clusters and move to the next attribute. The values to be considered for clustering have now been reduced to only those attached to the vectors from the previous attribute. This reduces the time complexity exponentially. Therefore, where vector-based and weighted clustering methods were clustering the entire dataset into 20 clusters, preference-based method reduced the clustering space iteratively.

This is observed as another significant advantage for the preference-based browsing approach where both accuracy and time-efficiency are concurred.

4.4.2.4 Inference from experiments conducted to evaluate efficiency

From the experiment conducted to evaluate the efficiency of the three service browsing methods, the following are the conclusions drawn based on the results:

- Vector-based clustering is time efficient in case of tackling large datasets but since it compromises on accuracy as well as imposes constraints on several case scenarios, it is not considered a very competent browsing method.
- Preference-based clustering may be more time-intensive than the other two clustering algorithms when it comes to tackling large datasets, but it competes and supersedes the other two clustering methods (vector-based and weighted approach) when handling flexible number of attributes or varying number of clusters. Together with its precision oriented methodology, it supplants the other two browsing techniques.
- Weighted clustering is time efficient as well as flexible and adaptable to the user's needs which places it second in the rank when compared to vector-based and preference-based approaches in terms of efficiency. It was noted to be the most productive browsing method in terms of handling increasingly large sized data. Since weighted clustering is a one-step approach, it reduces the total time taken when cumulated with the total time taken by the user to make cluster selection decisions.

4.4.3 Experiments conducted for usability study

A usability study was conducted on a sample demographic of 12 volunteers to test the viability of the browsing techniques presented in this research and to draw a comparison between them from the Web Service requestors' standpoint.

4.4.3.1 Experiment design and results

Who: A total of 12 volunteers were recruited with the purpose of conducting this usability test. 10 of these volunteers were graduate students pursuing a Masters' degree in Computer Science and were therefore well versed in this particular area. The two remaining volunteers were well literate in Computer Sciences but had an academic background in Finance Administration. The academic background of the users did not however make a significant impact on the browsing studies once the experiment and the objectives of this research were explained. An Online Ethics Submission and Review System (Ryerson University) was followed as per University protocol; which deemed this study a negligible risk and impact study out of the scope of the ethics board.

Why: The target demographic was conscripted to perform certain tasks and provide feedback about the program, which then contributed to the performance evaluation of the browsing mechanism put in place. Having an audience support the claims made by this research study would not only strengthen the conclusions and inferences drawn from the experiments but also secure the contributions made by this study. Feedback from the users is also an important factor used to improve the employed service selection tool.

What: The experiment was designed such that each user was given a set of tasks to perform using the browsing tool and their experience was recorded in terms of the total time taken to perform the task, the Cost Per Click (CPC) or the total number of steps taken to arrive at the desired result (including backtracking) and the successes or failures encountered during these tasks.

A set of 3 sample datasets were considered for this study with the following specifications:

Dataset	Cluster distribution	Size	Number of attributes	Specifications
Dataset 1	Indistinct	3000	3 (Cost, Response time, Reliability)	All clusters were indistinctly distributed such that values within a vector may belong in different clusters.
Dataset 2	Distinct	3000	3 (Cost, Response time, Reliability)	All clusters were clearly and distinctly distributed.
Dataset 3	Partially distinct	3000	3 (Cost, Response time, Reliability)	All clusters were clearly distributed for one attribute (Cost) and indistinctly distributed for all other attributes.

Table 7 - Dataset specification for usability study

Table 7 shows the dataset specification of the three datasets considered for this usability test. The idea was to form a few service selection search tasks such that each user would be given a set of tasks to perform. Depending upon their success or failure to reach the target service (or the cluster within which it exists), the usability factor of the browsing method would also be evaluated. The search tasks given to users were simply certain QoS vectors from within the dataset and based on the cluster values, the user had to indicate which cluster the service may be in. Following are some sample search tasks provided to the users:

Search Task	Dataset	Cost ($)	Response time (msec)	Reliability (%)
Search Task 1	Dataset 1	*55.607 - 56.4962*	*99.2928 - 103.8888*	*10.15 - 10.40*
	Dataset 2	*120.45 - 122.32*	*23.08 - 24.99*	*12.399 - 17.463*
	Dataset 3	*155.607 - 156.4962*	*34.292 - 36.888*	*50.15 - 50.40*
Search Task 2	Dataset 1	*180.9338 - 183.311*	*317.787 - 319.553*	*82.399 - 87.463*
	Dataset 2	*40.9338 - 43.311*	*119.456 – 121.219*	*91.399 - 94.463*
	Dataset 3	*200.325 – 202.298*	*233.276 – 235.909*	*78.33 – 81.98*
Search Task 3	Dataset 1	*90.3872 - 93.231*	*217.667 - 219.532*	*53.439 - 56.328*
	Dataset 2	*110.476 - 113.229*	*90.757 – 93.177*	*95.395 - 97.613*
	Dataset 3	*329.781 – 322.843*	*127.703 – 130.267*	*72.366 – 74.198*

Table 8 - Sample search tasks for usability test

Table 8 contains some of the search tasks which were provided to the usability study volunteers. These users were then familiarized with the three datasets provided in Table 7 and asked to locate the clusters within which these vectors may exist. The selection of these QoS vectors have been made such that finding these clusters in some cases would be straightforward in some cases and dubious in some, thus testing the decision making process on the users' end. The results of the usability study are as follows:

Task	**User**	**Clustering method**	**Time**	**Number of Steps/CPC**	**Result**
Task 1 for Dataset 1	User 1	Vector-based	*65 sec*	*1*	*Success*
		Preference-based	*78 sec*	*3*	*Success*
		Weighted	*60 sec*	*1*	*Success*
	User 2	Vector-based	*75 sec*	*1*	*Success*
		Preference-based	*90 sec*	*3*	*Success*
		Weighted	*65 sec*	*1*	*Success*
	User 3	Vector-based	*34 sec*	*1*	*Failure*
		Preference-based	*45 sec*	*3*	*Success*
		Weighted	*35 sec*	*1*	*Success*
Task 2 for dataset 2	User 1	Vector-based	*80 sec*	*3*	*Failure*
		Preference-based	*78 sec*	*3*	*Success*
		Weighted	*75 sec*	*1*	*Failure*
	User 2	Vector-based	*60 sec*	*3*	*Failure*
		Preference-based	*69 sec*	*3*	*Success*
		Weighted	*55 sec*	*1*	*Success*
	User 3	Vector-based	*71 sec*	*1*	*Failure*
		Preference-based	*82 sec*	*5*	*Success*
		Weighted	*70 sec*	*1*	*Success*
Task 3 for dataset 3	User 1	Vector-based	*60 sec*	*1*	*Success*
		Preference-based	*75 sec*	*3*	*Success*
		Weighted	*55 sec*	*2*	*Success*
	User 2	Vector-based	*80 sec*	*4*	*Failure*
		Preference-based	*110 sec*	*6*	*Failure*
		Weighted	*75 sec*	*2*	*Failure*
	User 3	Vector-based	*69 sec*	*1*	*Success*
		Preference-based	*80 sec*	*3*	*Success*
		Weighted	*62 sec*	*1*	*Success*

Table 9 - Some results acquired from usability testing

Table 9 shows some of the results acquired during the usability testing. Only three search tasks tested with three users have been shown here in order to get oriented with the result criteria and the quality of results attained from this study. It was noted that most users tended to achieve success in finding the correct clusters with preference-based method and had moderate success with weighted method when compared to vector-based method. Preference-based clustering did assume more time due to its multilevel approach, but it was trade-off for achieving the desired result with accuracy and a higher probability. In order to analyze the results obtained from all the 12 participants of this study, a few gradients and chart-analysis were carried out. Some of the useful trends were studying the success rate of the users for each of the search tasks.

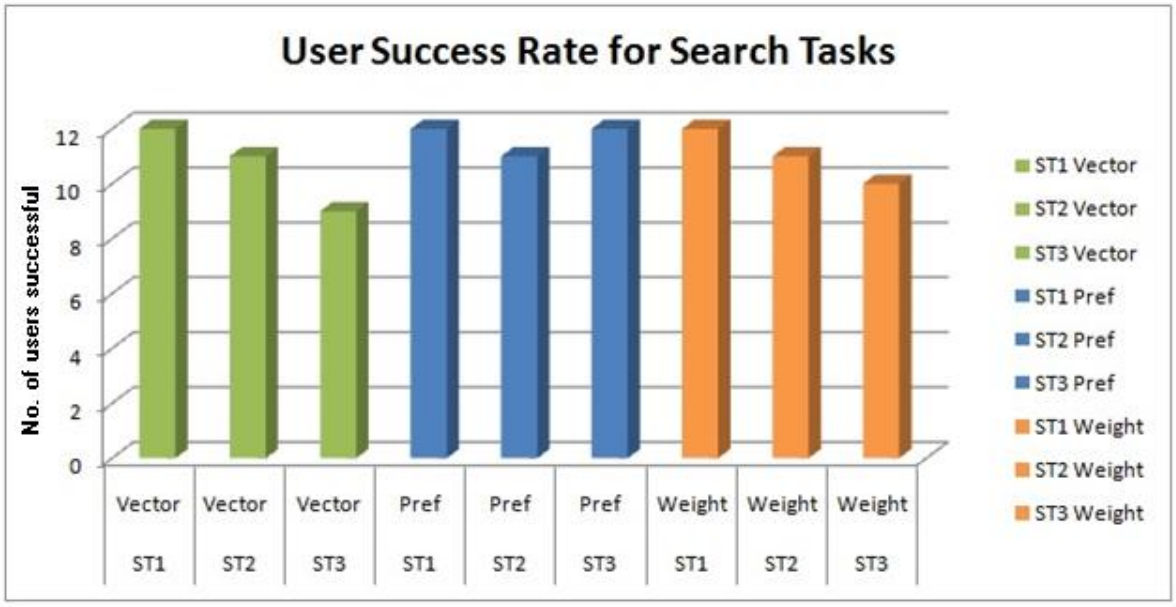

Figure 15 - User success rate for Search Tasks

Figure 15 is a comparison between each of the browsing methods and the success rate users had with respect to finding the given query. It was observed that users had a much higher chance of finding the right cluster in any given dataset scenario or use case and a greater failure rate when using vector-based clustering in terms of handling different kinds of search tasks.

Weighted clustering fared in between preference-based and vector-based methods with respect to success rate but still had a greater advantage when compared to vector-based browsing method.

Having discussed the success rates achieved by each of the clustering algorithms, it was also important to study the time expended by each user to perform these search tasks. In order to draw a comparison module, the average time taken by each of the 12 participants was calculated and charted against the respective search tasks.

Figure 16 - Time efficiency for search tasks

Figure 16 depicts the average time taken to complete each of the search tasks for each browsing method. It was observed that preference-based clustering was more time-consuming when compared to the other two methods. Even though it was proved that the run-time of the preference-based method is at par and even better in several case compared to vector-based and weighted browsing methods, it is noted here that due to the multi-level step by step approach the user tended to take more time in studying the results and choosing the most suitable (in this case

the given Search Task) cluster. This may not be deemed entirely as a wasteful exercise since the trade-off for a few extra seconds leads to a highly satisfied service requestor. However, weighted clustering was observed to be more time-efficient as well as goal-oriented, which allowed users to choose their preferred attribute(s) as well as save time on the clustering process in most cases. The trade-off here was a significant amount of time saved for a small amount of inaccuracy or inability to handle all scenarios and situations.

It was also considered important to measure the number of clicks incurred for each clustering method as a performance metric for the expense of the tool. This included any backtracking the participant may have done during the browsing process since it provides an insight into the decision making process of the user. The comparison module for each browsing method was drawn against the average measure of the number of clicks (also known as Cost per Click or CPC).

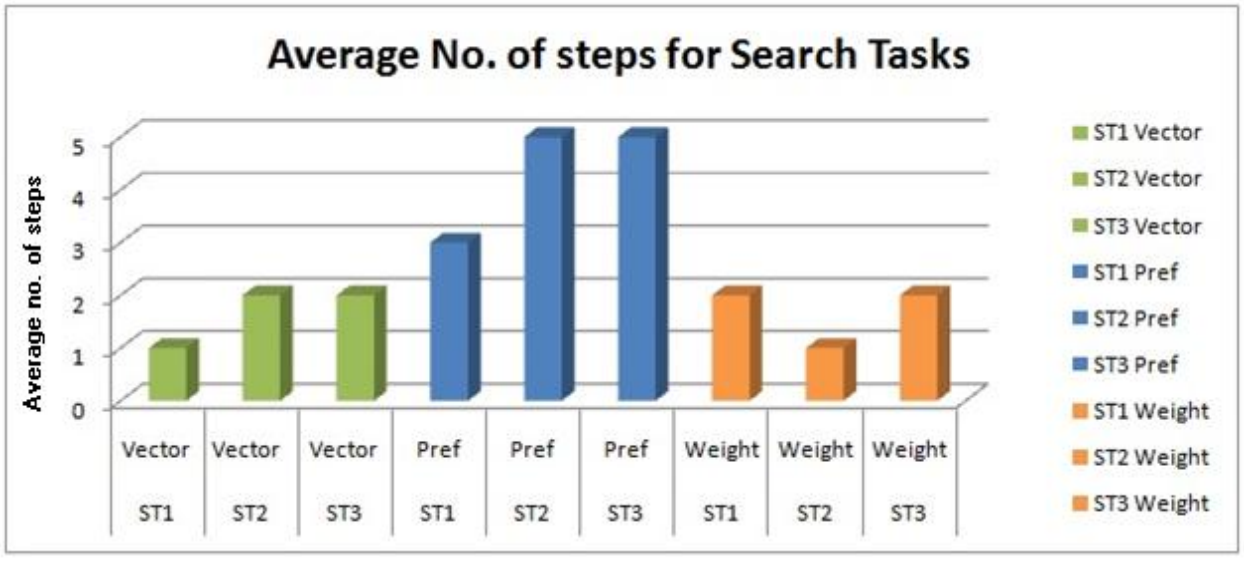

Figure 17 - Average number of clicks for Search Tasks

Figure 17 illustrates the average number of steps encountered for each search task when testing each of the browsing methods. It was clearly indicated that preference-based approach

claimed more clicks from the user as compared to vector-based or weighted clustering method. Since weighted method has a higher accuracy rate, it may be considered as the most suitable browsing method with regards to the cost of clicks.

The participants of the usability study were also asked for their opinion about some the quality aspects of the program such as ease of use, navigation through the tool, accessibility to each component, exception handling and the visualization work done in the program. The participants were asked to rate the tool with respect to each of these factors on a scale of 1-10, 10 being the highly successful in fulfilling the program usability criterion.

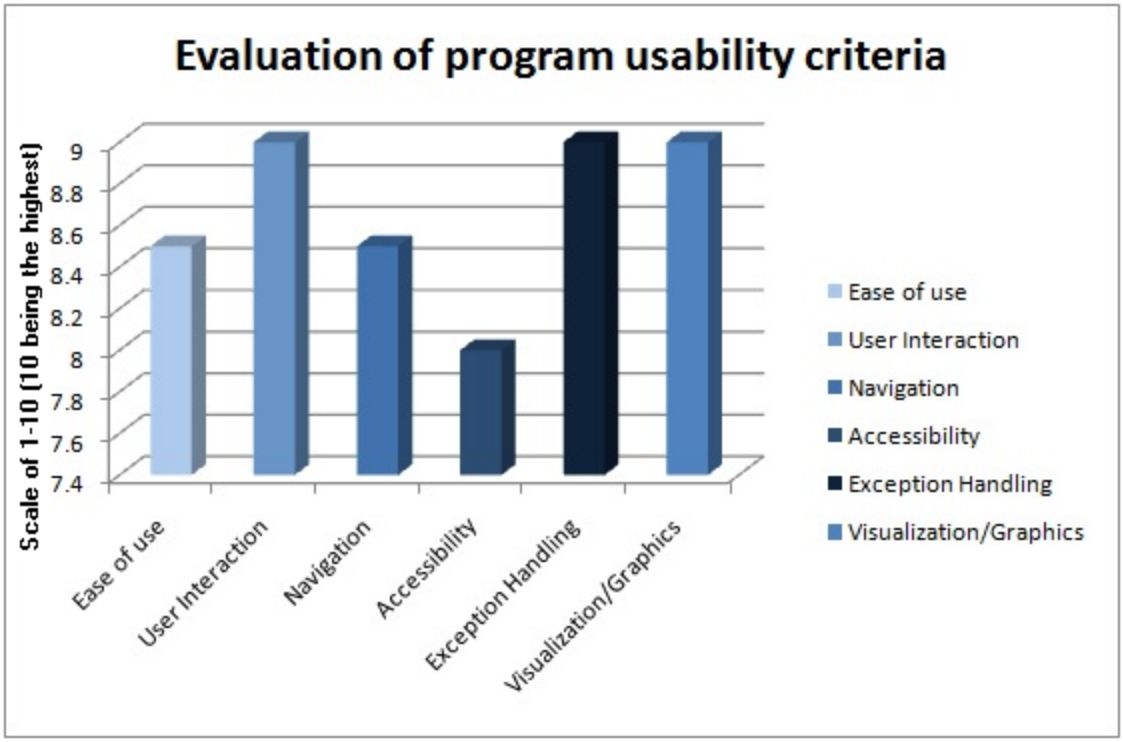

Figure 18 – Evaluation of program usability criteria

Figure 18 illustrates the average rating assigned to each program usability criteria. Exception handling and visualization components were given graded higher than accessibility, navigation and ease of use. User Interaction also scored highly over other criterions. This

feedback was important to improve as well as gain an insight into what the participants' estimated to be the quality aspects of the program.

4.5 Result analysis and summary

This chapter focused upon thoroughly investigating vector-based, preference-based and weighted service selection browsing methods by accounting for the various experiments conducted to evaluate their performance. The evaluation was carried out in three segments – evaluation based on clustering accuracy and adaptability to different dataset scenarios, evaluation based on efficiency and evaluation based on a usability study. Evaluation based on accuracy was conducted based on two common dataset scenarios encountered in real datasets. Evaluation based on efficiency was conducted based on the time efficiency for several performance metrics such as varying dataset size, number of attributes and number of clusters. Evaluation based on the usability study was conducted to compare the three browsing methods from the users' perspective and to gain feedback on the performance of the three clustering methods as well as the program usability criteria for the implemented tool.

The results obtained from the experiment conducted to evaluate the accuracy of the clusters proved that preference-based clustering was the most adaptable as well as accurate in terms of finding the desired and the most suitable service in a methodological way. Weighted clustering was also deemed accurate; however it wasn't entirely adaptable with all dataset scenarios and case studies. Vector-based clustering presented several constraints with regards to locating the desired clusters in datasets which do not already have distinctly defined natural groupings within them.

The result obtained from the experiment conducted to evaluate the efficiency of the clusters established that preference-based clustering was the most efficient browsing method and fared well even when the number of clusters per attribute or the number of attributes were made variant. It did not show promising results when the size of the datasets were increased, in which case, weighted clustering rose to the top and delivered not only quality results but also managed a better measure of time. Vector-based clustering also had a good measure of time owing to its single-step approach, however, the quality of results and restrictive accessibility did not compete with the performance of preference-based or weighted browsing methods.

The feedback obtained from the usability experiment confirmed that the users preferred accuracy over transactions of small amounts of time-loss and were satisfied with the performance of preference-based clustering, however, the time taken by the participants and the cost of clicks incurred were both evaluated to be higher in the case of preference-based clustering when compared to the other two browsing methods. Another segment of the usability study also required the participants to rate the implemented tool in terms of quality criterions where user interaction, exception handling and visualization component scored marginally higher over accessibility, navigation and ease of use of the program.

CHAPTER 5

CONCLUSION

5.1 Summary and Results

This thesis was written with the proclivity to make a contribution in the field of semantic and QoS based Web Service discovery and selection.

The current e-commerce industry calls for dynamic and quality-driven service discovery, browsing and selection mechanisms. This research work has been focussed upon user-centric requirements and preferences without causing information overload or a highly involved multi level process. Clustering was used as a means to group and streamline the various functionally relevant Web Services in order to browse through them efficiently. The three main clustering algorithms investigated in this thesis work were Vector-based approach, Preference-based approach and weight-based approach. Vector-based clustering method has already been investigated earlier however, in order to draw a fair comparison; the base model from that implementation was considered and re-implemented along with Preference-based and weighted clustering methods in this thesis.

It was determined, with the help of the performance analysis conducted on the experiments and a usability study carried out to assess and compare the three methods implemented, that preference-based browsing method was more efficient that the other methods in comparison. It was widely accepted and fared better in most experiments in terms of quality of cluster, accuracy and ease of discovery as well as average time taken. Weight-based method was the next best method in the performance charts owing to its ease of cluster identification and time efficiency.

5.2 Future Work

There have been several studies and research works that have been done in the field of Web Service discovery and selection. This study was conducted with a view of easing that process with the aid of a browsing mechanism which uses clustering as its basis for grouping qualitatively similar Web Services together. Several issues and drawbacks which were observed in previously existing and comparable methods were addressed with this browsing mechanism. The idea behind this study was also to provide an extensive comparison model of three main clustering based methods used for service discovery.

As observed earlier, one of the more comparable recent works done in this particular field (Sambamoorthy, Interactive QoS bowsing for web service selection, 2009) was that of an interactive QoS based browsing method where a pre-packaged tool was used to implement dynamic clustering for QoS based Web Service data sets. Some of the key shortcomings observed in that study have been addressed and beyond in this research work, and are listed as follows:

a) The lack of a visual interface prohibited the users from making an informed decision since users could only guess the span of the clusters and their locality. This problem has been addressed in this dissertation since a complete GUI has been implemented with interactive features built to improve the user experience.
b) The lack of standard datasets was a clear limitation in Sambamoorthy's (Sambamoorthy, QoS Browsing for Web Service Selection, 2009) exposition. The study was incomplete without standardized data sets which would help provide a comparative performance evaluation of the cluster analysis. In this study, a step has been taken towards simulating

data sets which have been inspired from various real sources as well previously established norms of data set formats and distributions.

c) The clustering performed using the SODUS package was one dimensional and did not cater or multi-dimensional clustering where users may contribute to the process of clustering with a preference or weighted clustering browsing method. This has been a key element in the current study where comparisons have also been drawn between each clustering type in terms of performance and efficiency.

d) In Sambamoorthy's work, only three QoS attributes were considered in each experiment, thereby limiting the search space and reducing the field within which the users could browse. In this study, upto nine attributes have been provided to the users in some experiments should they choose to have them. Each QoS domain was studied before incorporating the most prominent attributes into the datasets.

Although forethought has been put towards achieving a comparative and cohesive browsing mechanism for Web Service requestors, there are certain limitations of this work which may be addressed as part of a future study or continuing research. Some of these future works are listed as follows:

a) The performance evaluation performed as part of this comparative study could not incorporate a standard index co-efficient (such as C-H index, C-index and Γ-index (C Ding, 2009)) due to the multi-level and interactive nature of the clustering methods. It would be a considerable contribution if a standardized performance index could be developed for this purpose.

b) The data sets used for this study were simulated and inspired from several real data sources and use case scenarios because the purpose of this study was to compare the three

different clustering methods, however, if such a browsing method were to be used in real time, real data sets must be collected and implemented in such a system.

c) Although all datasets generated were in text (.txt) or tab separated (.csv) format, they should be converted to XML since it is closer to Web Standards and is easier to use with machine languages.

d) It is also important to keep in mind that K-means, although suitable for this study, does have several drawbacks especially its tendency to optimize locally and choice of random prototypes resulting in different clusters every time. Future studies may also include performing a comparative study or research in other dynamic clustering algorithms.

APPENDIX A: DATA GENERATION

The following data specifications and corroborating details specify the datasets generated for user study, use case scenarios and testing purposes.

- All datasets are generated using Matlab functions.
- Each dataset follows random, multi-variate, normal distribution.
- The filenames have been given using the following format.
  (Type of main clusters (distinct/ indistinct)_(Type of sub clusters)_(Number of data in each column)_(version (in case more than one file with same distribution is generated again))
  Example: Distinct_distinct_900_2 denotes distinct clusters containing distinct sub clusters with 900 data points and the second of this kind of distribution.

- The first column denotes the cluster, size of cluster and the respective sizes of each sub cluster. It is easy to determine the start and end of each sub cluster in the dataset, in order to calculate accuracy.
  Example: Clust 1 (1000) (330, 330, 340) means that row 1 - row 330 are 1st sub cluster, the next 330 data points (i.e. row 331 – 660) denote the 2nd sub cluster and the next 340 (row 661-1000) are the 3rd sub cluster.

Dataset 1:

Filename: distinct_distinct_3000_1

In this dataset we have generated random data following a multivariate normal distribution for three interval variables Cost ($), Response time (ms) and Reliability (scale of 100) respectively. Even though this data is simulated, the idea was to base it on a hypothesis so real data following these distributions may actually exist in almost all cases. The following were the specifications followed. The idea was to generate 3 main clusters, each with its own set of 3 sub clusters. We have generated a total of 3000 data points with an equal distribution spread in each cluster.

Clust 1 (1000) (330,330,340)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 55$	$\sigma_{11} = 2.0$	$\mu_{11} = 100$	$\sigma_{11} = 2.0$	$\mu_{12} = 10$	$\sigma_{11} = 1.0$
	$\mu_{12} = 80$	$\sigma_{12} = 2.0$	$\mu_{12} = 140$	$\sigma_{12} = 1.8$	$\mu_{13} = 17$	$\sigma_{12} = 1.2$
	$\mu_{13} = 125$	$\sigma_{13} = 2.0$	$\mu_{13} = 170$	$\sigma_{13} = 3.0$	$\mu_{14} = 23$	$\sigma_{13} = 0.9$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 310$	$\sigma_{21} = 3.0$	$\mu_{21} = 400$	$\sigma_{21} = 3.0$	$\mu_{22} = 42$	$\sigma_{21} = 0.8$
	$\mu_{22} = 360$	$\sigma_{22} = 3.6$	$\mu_{22} = 450$	$\sigma_{22} = 2.4$	$\mu_{23} = 49$	$\sigma_{22} = 1.3$
	$\mu_{23} = 420$	$\sigma_{23} = 4.0$	$\mu_{23} = 510$	$\sigma_{23} = 4.0$	$\mu_{24} = 56$	$\sigma_{23} = 0.6$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 660$	$\sigma_{31} = 2.5$	$\mu_{31} = 760$	$\sigma_{31} = 3.6$	$\mu_{31} = 77$	$\sigma_{31} = 1.2$
	$\mu_{32} = 700$	$\sigma_{32} = 3.9$	$\mu_{32} = 800$	$\sigma_{32} = 3.0$	$\mu_{32} = 83$	$\sigma_{32} = 1.0$
	$\mu_{33} = 740$	$\sigma_{33} = 2.0$	$\mu_{33} = 830$	$\sigma_{33} = 2.0$	$\mu_{33} = 89$	$\sigma_{33} = 0.9$

Dataset 2:

Filename: distinct_distinct_300_1

In this dataset, the same distribution and order is followed as in Dataset 1, only this time, we have generated only 300 data points. The idea behind this dataset generation was to test the efficiency of our clustering algorithm and its computation capability regarding different sized datasets.

Clust 1 (100) (33,33,34)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 55$	$\sigma_{11} = 2.0$	$\mu_{11} = 100$	$\sigma_{11} = 2.0$	$\mu_{11} = 10$	$\sigma_{11} = 1.0$
	$\mu_{12} = 80$	$\sigma_{12} = 2.0$	$\mu_{12} = 140$	$\sigma_{12} = 1.8$	$\mu_{12} = 17$	$\sigma_{12} = 1.2$
	$\mu_{13} = 125$	$\sigma_{13} = 2.0$	$\mu_{13} = 170$	$\sigma_{13} = 3.0$	$\mu_{13} = 23$	$\sigma_{13} = 0.9$
Clust 2 (100) (33,33,34)						
	$\mu_{21} = 310$	$\sigma_{21} = 3.0$	$\mu_{21} = 400$	$\sigma_{21} = 3.0$	$\mu_{21} = 42$	$\sigma_{21} = 0.8$
	$\mu_{22} = 360$	$\sigma_{22} = 3.6$	$\mu_{22} = 450$	$\sigma_{22} = 2.4$	$\mu_{22} = 49$	$\sigma_{22} = 1.3$
	$\mu_{23} = 420$	$\sigma_{23} = 4.0$	$\mu_{23} = 510$	$\sigma_{23} = 4.0$	$\mu_{23} = 56$	$\sigma_{23} = 0.6$
Clust 3 (100) (33,33,34)						
	$\mu_{31} = 660$	$\sigma_{31} = 2.5$	$\mu_{31} = 760$	$\sigma_{31} = 3.6$	$\mu_{31} = 77$	$\sigma_{31} = 1.2$
	$\mu_{32} = 700$	$\sigma_{32} = 3.9$	$\mu_{32} = 800$	$\sigma_{32} = 3.0$	$\mu_{32} = 83$	$\sigma_{32} = 1.0$
	$\mu_{33} = 740$	$\sigma_{33} = 2.0$	$\mu_{33} = 830$	$\sigma_{33} = 2.0$	$\mu_{33} = 89$	$\sigma_{33} = 0.9$

Dataset 3:

Filename: Distinct_distinct_30000_1

The following dataset follows the same distribution pattern as the above data sets, only, here the dataset contains 30,000 data points. A larger dataset is generated to compare the time taken, effectiveness and efficiency of our algorithm.

Clust 1 (10,000) (3300,3300,3400)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 55$	$\sigma_{11} = 2.0$	$\mu_{11} = 100$	$\sigma_{11} = 2.0$	$\mu_{11} = 10$	$\sigma_{11} = 1.0$
	$\mu_{12} = 80$	$\sigma_{12} = 4$	$\mu_{12} = 140$	$\sigma_{12} = 1.8$	$\mu_{12} = 17$	$\sigma_{12} = 1.2$
	$\mu_{13} = 125$	$\sigma_{13} = 4$	$\mu_{13} = 170$	$\sigma_{13} = 3.0$	$\mu_{13} = 23$	$\sigma_{13} = 0.9$
Clust 2 (10,000) (3300,3300,3400)						
	$\mu_{21} = 310$	$\sigma_{21} = 3.0$	$\mu_{21} = 400$	$\sigma_{21} = 3.0$	$\mu_{21} = 42$	$\sigma_{21} = 0.8$
	$\mu_{22} = 360$	$\sigma_{22} = 3.6$	$\mu_{22} = 450$	$\sigma_{22} = 2.4$	$\mu_{22} = 49$	$\sigma_{22} = 1.3$
	$\mu_{23} = 420$	$\sigma_{23} = 4.0$	$\mu_{23} = 510$	$\sigma_{23} = 4.0$	$\mu_{23} = 56$	$\sigma_{23} = 0.6$
Clust 3 (10,000) (3300,3300,3400)						
	$\mu_{31} = 660$	$\sigma_{31} = 2.5$	$\mu_{31} = 760$	$\sigma_{31} = 3.6$	$\mu_{31} = 77$	$\sigma_{31} = 1.2$
	$\mu_{32} = 700$	$\sigma_{32} = 3.8$	$\mu_{32} = 800$	$\sigma_{32} = 3.0$	$\mu_{32} = 83$	$\sigma_{32} = 1.0$
	$\mu_{33} = 740$	$\sigma_{33} = 2.0$	$\mu_{33} = 830$	$\sigma_{33} = 2.0$	$\mu_{33} = 89$	$\sigma_{33} = 0.9$

Dataset 4:

Filename: Distinct_undistinct_3000_1

Here, we have generated a 3000 data point dataset following a multivariate normal distribution such that there are 3 distinct clusters however each cluster has 3 indistinct sub clusters even though each sub cluster has its own mean and std. deviation. The following were the specifications used to generate this dataset.

Clust 1 (1000) (330,330,340)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 150$	$\sigma_{11} = 10$	$\mu_{11} = 100$	$\sigma_{11} = 4$	$\mu_{11} = 50$	$\sigma_{11} = 2$
	$\mu_{12} = 170$	$\sigma_{12} = 9$	$\mu_{12} = 110$	$\sigma_{12} = 5$	$\mu_{12} = 55$	$\sigma_{12} = 3$
	$\mu_{13} = 200$	$\sigma_{13} = 12$	$\mu_{13} = 120$	$\sigma_{13} = 6$	$\mu_{13} = 57$	$\sigma_{13} = 3$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 410$	$\sigma_{21} = 10$	$\mu_{21} = 250$	$\sigma_{21} = 4$	$\mu_{21} = 66$	$\sigma_{21} = 2$
	$\mu_{22} = 430$	$\sigma_{22} = 20$	$\mu_{22} = 260$	$\sigma_{22} = 6$	$\mu_{22} =69$	$\sigma_{22} = 2.5$
	$\mu_{23} = 460$	$\sigma_{23} = 15$	$\mu_{23} = 275$	$\sigma_{23} = 7$	$\mu_{23} = 73$	$\sigma_{23} = 3$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 700$	$\sigma_{31} = 12$	$\mu_{31} = 370$	$\sigma_{31} = 8$	$\mu_{31} = 80$	$\sigma_{31} = 3$
	$\mu_{32} = 730$	$\sigma_{32} = 11$	$\mu_{32} = 385$	$\sigma_{32} = 10$	$\mu_{32} = 85$	$\sigma_{32} = 2$
	$\mu_{33} = 750$	$\sigma_{33} = 8$	$\mu_{33} = 400$	$\sigma_{33} = 10$	$\mu_{33} = 88$	$\sigma_{33} = 3$

Dataset 5:

Filename: Distinct_undistinct_300_1

Clust 1 (100) (33,33,34)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 150$	$\sigma_{11} = 10$	$\mu_{11} = 100$	$\sigma_{11} = 4$	$\mu_{11} = 50$	$\sigma_{11} = 2$
	$\mu_{12} = 170$	$\sigma_{12} = 9$	$\mu_{12} = 110$	$\sigma_{12} = 5$	$\mu_{12} = 55$	$\sigma_{12} = 3$
	$\mu_{13} = 200$	$\sigma_{13} = 12$	$\mu_{13} = 120$	$\sigma_{13} = 6$	$\mu_{13} = 57$	$\sigma_{13} = 3$
Clust 2 (100) (33,33,34)						
	$\mu_{21} = 410$	$\sigma_{21} = 10$	$\mu_{21} = 250$	$\sigma_{21} = 4$	$\mu_{21} = 66$	$\sigma_{21} = 2$
	$\mu_{22} = 430$	$\sigma_{22} = 20$	$\mu_{22} = 260$	$\sigma_{22} = 6$	$\mu_{22} =69$	$\sigma_{22} = 2.5$
	$\mu_{23} = 460$	$\sigma_{23} = 15$	$\mu_{23} = 275$	$\sigma_{23} = 7$	$\mu_{23} = 73$	$\sigma_{23} = 3$
Clust 3 (100) (33,33,34)						
	$\mu_{31} = 700$	$\sigma_{31} = 12$	$\mu_{31} = 370$	$\sigma_{31} = 8$	$\mu_{31} = 80$	$\sigma_{31} = 3$
	$\mu_{32} = 730$	$\sigma_{32} = 11$	$\mu_{32} = 385$	$\sigma_{32} = 10$	$\mu_{32} = 85$	$\sigma_{32} = 2$
	$\mu_{33} = 750$	$\sigma_{33} = 8$	$\mu_{33} = 400$	$\sigma_{33} = 10$	$\mu_{33} = 88$	$\sigma_{33} = 3$

Dataset 6:

Filename: Distinct_undistinct_30000_1

Clust 1 (10,000) (3300,3300,3400)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 150$	$\sigma_{11} = 10$	$\mu_{11} = 100$	$\sigma_{11} = 4$	$\mu_{11} = 50$	$\sigma_{11} = 2$
	$\mu_{12} = 170$	$\sigma_{12} = 9$	$\mu_{12} = 110$	$\sigma_{12} = 5$	$\mu_{12} = 55$	$\sigma_{12} = 3$
	$\mu_{13} = 200$	$\sigma_{13} = 12$	$\mu_{13} = 120$	$\sigma_{13} = 6$	$\mu_{13} = 57$	$\sigma_{13} = 3$
Clust 2 (10,000) (3300,3300,3400)						
	$\mu_{21} = 410$	$\sigma_{21} = 10$	$\mu_{21} = 250$	$\sigma_{21} = 4$	$\mu_{21} = 66$	$\sigma_{21} = 2$
	$\mu_{22} = 430$	$\sigma_{22} = 15$	$\mu_{22} = 260$	$\sigma_{22} = 4$	$\mu_{22} =69$	$\sigma_{22} = 2.5$
	$\mu_{23} = 460$	$\sigma_{23} = 20$	$\mu_{23} = 275$	$\sigma_{23} = 7$	$\mu_{23} = 73$	$\sigma_{23} = 3$
Clust 3 (10,000) (3300,3300,3400)						
	$\mu_{31} = 700$	$\sigma_{31} = 12$	$\mu_{31} = 370$	$\sigma_{31} = 8$	$\mu_{31} = 80$	$\sigma_{31} = 3$
	$\mu_{32} = 730$	$\sigma_{32} = 11$	$\mu_{32} = 385$	$\sigma_{32} = 10$	$\mu_{32} = 85$	$\sigma_{32} = 2$
	$\mu_{33} = 750$	$\sigma_{33} = 8$	$\mu_{33} = 400$	$\sigma_{33} = 10$	$\mu_{33} = 88$	$\sigma_{33} = 3$

Dataset 7:

Filename: Case_1_distinct_900_1

For this data generation we have followed certain use case scenarios to show the difference between our clustering algorithm in comparison with methods that cluster the entire vector. With this dataset, our browsing system lets the requestor choose each cluster for each parameter as opposed to being able to choose only one desired parameter in a cluster. This dataset also follows a multivariate normal distribution with 900 data points.

Clust 1 (300) (33,33,34)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 50$	$\sigma_{11} = 2.5$	$\mu_{11} = 90$	$\sigma_{11} = 1.2$	$\mu_{11} = 60$	$\sigma_{11} = 1.8$
	$\mu_{12} = 75$	$\sigma_{12} = 2.8$	$\mu_{12} = 100$	$\sigma_{12} = 2$	$\mu_{12} = 67$	$\sigma_{12} = 1.6$
	$\mu_{13} = 98$	$\sigma_{13} = 3$	$\mu_{13} = 115$	$\sigma_{13} = 1.8$	$\mu_{13} = 74$	$\sigma_{13} = 1.5$
Clust 2 (300) (33,33,34)						
	$\mu_{21} = 310$	$\sigma_{21} = 4$	$\mu_{21} = 185$	$\sigma_{21} = 2.2$	$\mu_{21} = 80$	$\sigma_{21} = 1.4$
	$\mu_{22} = 340$	$\sigma_{22} = 2.8$	$\mu_{22} = 200$	$\sigma_{22} = 2$	$\mu_{22} = 86$	$\sigma_{22} = 1.1$
	$\mu_{23} = 365$	$\sigma_{23} = 2.6$	$\mu_{23} = 220$	$\sigma_{23} = 2.1$	$\mu_{23} = 93$	$\sigma_{23} = 1.2$
Clust 3 (300) (33,33,34)						
	$\mu_{31} = 210$	$\sigma_{31} = 2.5$	$\mu_{31} = 285$	$\sigma_{31} = 3$	$\mu_{31} = 49$	$\sigma_{31} = 1$
	$\mu_{32} = 235$	$\sigma_{32} = 2.9$	$\mu_{32} = 300$	$\sigma_{32} = 2.7$	$\mu_{32} = 55$	$\sigma_{32} = 1.2$
	$\mu_{33} = 260$	$\sigma_{33} = 4$	$\mu_{33} = 315$	$\sigma_{33} = 2.5$	$\mu_{33} = 58$	$\sigma_{33} = 0.8$

Dataset 8:

Filename: Complete_overlap_1500_1

In the following dataset, the idea was to generate three sub clusters as in the previous data sets. However, the last sub cluster is generated such that it contains data points with large intervals, thus engulfing the other sub clusters into it. This dataset has 1000 data points in it.

	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
500 data points (170,170,160)	$\mu_{11} = 100$	$\sigma_{11} = 2$	$\mu_{11} = 125$	$\sigma_{11} = 2.4$	$\mu_{11} = 50$	$\sigma_{11} = 2.5$
	$\mu_{12} = 125$	$\sigma_{12} = 2.8$	$\mu_{12} = 140$	$\sigma_{12} = 2$	$\mu_{12} = 63$	$\sigma_{12} = 3$
	$\mu_{13} = 155$	$\sigma_{13} = 3.2$	$\mu_{13} = 160$	$\sigma_{13} = 2$	$\mu_{13} = 74$	$\sigma_{13} = 2$
500 data points (170,170,160)						
	$\mu_{21} = 210$	$\sigma_{21} = 4$	$\mu_{21} = 185$	$\sigma_{21} = 2.2$	$\mu_{21} = 80$	$\sigma_{21} = 1.4$
	$\mu_{22} = 245$	$\sigma_{22} = 3$	$\mu_{22} = 200$	$\sigma_{22} = 2.2$	$\mu_{22} = 87$	$\sigma_{22} = 1.5$
	$\mu_{23} = 265$	$\sigma_{23} = 2.5$	$\mu_{23} = 220$	$\sigma_{23} = 2$	$\mu_{23} = 95$	$\sigma_{23} = 1.5$
500 data points (500)						
	$\mu_{31} = 160$	$\sigma_{31} = 40$	$\mu_{31} = 150$	$\sigma_{31} = 25$	$\mu_{31} = 70$	$\sigma_{31} = 15$

In the above case, our algorithm still works better than when compared to vector based clustering because vector based clustering would have massive overlapping (in fact it would not be able to distinguish any sub clusters).

Dataset 9:

Filename: Partial_overlap_1500_1

Another example of this kind of data can be seen below. Here, only one parameter is generated with data points having large intervals, all other parameters are normal. In this case, vector clustering would still yield the same results whereas, in preference based clustering, distinct sub clusters may be seen for other parameters thus minimizing the anomaly.

	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
500 data points (170,170,160)	$\mu_{11} = 100$	$\sigma_{11} = 2$	$\mu_{11} = 125$	$\sigma_{11} = 2.4$	$\mu_{11} = 60$	$\sigma_{11} = 1.8$
	$\mu_{12} = 125$	$\sigma_{12} = 2.8$	$\mu_{12} = 140$	$\sigma_{12} = 2$	$\mu_{12} = 67$	$\sigma_{12} = 1.6$
	$\mu_{13} = 155$	$\sigma_{13} = 3.2$	$\mu_{13} = 160$	$\sigma_{13} = 2$	$\mu_{13} = 74$	$\sigma_{13} = 1.5$
500 data points (170,170,160)						
	$\mu_{21} = 210$	$\sigma_{21} = 4$	$\mu_{21} = 185$	$\sigma_{21} = 2.2$	$\mu_{21} = 80$	$\sigma_{21} = 1.4$
	$\mu_{22} = 245$	$\sigma_{22} = 3$	$\mu_{22} = 200$	$\sigma_{22} = 2.2$	$\mu_{22} = 86$	$\sigma_{22} = 1.1$
	$\mu_{23} = 265$	$\sigma_{23} = 2.5$	$\mu_{23} = 220$	$\sigma_{23} = 2$	$\mu_{23} = 93$	$\sigma_{23} = 1.2$
500 data points (170,170,160)						
	$\mu_{31} = 180$	$\sigma_{31} = 40$	$\mu_{31} = 285$	$\sigma_{31} = 3$	$\mu_{31} = 49$	$\sigma_{31} = 1$
			$\mu_{32} = 300$	$\sigma_{32} = 2.7$	$\mu_{32} = 55$	$\sigma_{32} = 1.2$
			$\mu_{33} = 315$	$\sigma_{33} = 2.5$	$\mu_{33} = 58$	$\sigma_{33} = 0.8$

Dataset 10:

Filename: Complete_Meshed_900_1

Let us consider a dataset where there is absolutely no possibility of having distinct clusters. Such dataset reveals no pattern and provides enmeshed, overlapped clusters each time.

The following is an example where all values are chaotic.

300 data points (100,100,100)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 55$	$\sigma_{11} = 5$	$\mu_{11} = 100$	$\sigma_{11} = 10$	$\mu_{11} = 30$	$\sigma_{11} = 2.5$
	$\mu_{12} = 70$	$\sigma_{12} = 5$	$\mu_{12} = 140$	$\sigma_{12} = 10$	$\mu_{12} = 43$	$\sigma_{12} = 2$
	$\mu_{13} = 85$	$\sigma_{13} = 5$	$\mu_{13} = 170$	$\sigma_{13} = 10$	$\mu_{13} = 35$	$\sigma_{13} = 2.5$
300 data points (100,100,100)						
	$\mu_{21} = 50$	$\sigma_{21} = 5$	$\mu_{21} = 110$	$\sigma_{21} = 10$	$\mu_{21} = 42$	$\sigma_{21} = 2.8$
	$\mu_{22} = 65$	$\sigma_{22} = 5$	$\mu_{22} = 155$	$\sigma_{22} = 10$	$\mu_{22} =45$	$\sigma_{22} = 2.5$
	$\mu_{23} = 80$	$\sigma_{23} = 5$	$\mu_{23} = 180$	$\sigma_{23} = 10$	$\mu_{23} = 37$	$\sigma_{23} = 2$
300 data points (100,100,100)						
	$\mu_{31} = 52$	$\sigma_{31} = 5$	$\mu_{31} = 125$	$\sigma_{31} = 10$	$\mu_{31} = 48$	$\sigma_{31} = 2.5$
	$\mu_{32} = 63$	$\sigma_{32} = 5$	$\mu_{32} = 165$	$\sigma_{32} = 10$	$\mu_{32} = 36$	$\sigma_{32} = 2$
	$\mu_{33} = 78$	$\sigma_{33} = 5$	$\mu_{33} = 195$	$\sigma_{33} = 10$	$\mu_{33} = 33$	$\sigma_{33} = 2$

Dataset 11:

Filename: Partial_meshed_900_1

The following table is a diminutive from the above example where only one parameter is distributed chaotically and the others are distributed in a sound normal way. The difference between the two datasets is that in this case our algorithm would contain the overlapping distribution only to one parameter. This however does not mean that the browsing process would go smoothly. If the chaotically distributed parameter is higher up in the preference order, the user will be provided with overlapping list of clusters. The chosen cluster will trigger the set of clustering for other parameter(s) which will lead to increasingly inaccurate results. Therefore these kind of datasets do not work perfectly with our algorithm, though they do have an upper hand over vector based clustering.

	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
300 data points (100,100,100)	$\mu_{11} = 55$	$\sigma_{11} = 5$	$\mu_{11} = 100$	$\sigma_{11} = 2.0$	$\mu_{11} = 10$	$\sigma_{11} = 1.0$
	$\mu_{12} = 70$	$\sigma_{12} = 5$	$\mu_{12} = 140$	$\sigma_{12} = 1.8$	$\mu_{12} = 14$	$\sigma_{12} = 1.1$
	$\mu_{13} = 85$	$\sigma_{13} = 5$	$\mu_{13} = 170$	$\sigma_{13} = 3.0$	$\mu_{13} = 17$	$\sigma_{13} = 0.9$
300 data points (100,100,100)						
	$\mu_{21} = 50$	$\sigma_{21} = 5$	$\mu_{21} = 400$	$\sigma_{21} = 3.0$	$\mu_{21} = 42$	$\sigma_{21} = 0.8$
	$\mu_{22} = 65$	$\sigma_{22} = 5$	$\mu_{22} = 450$	$\sigma_{22} = 2.4$	$\mu_{22} =45$	$\sigma_{22} = 1.3$
	$\mu_{23} = 80$	$\sigma_{23} = 5$	$\mu_{23} = 510$	$\sigma_{23} = 4.0$	$\mu_{23} = 48$	$\sigma_{23} = 0.6$
300 data points (100,100,100)						
	$\mu_{31} = 52$	$\sigma_{31} = 5$	$\mu_{31} = 760$	$\sigma_{31} = 3.6$	$\mu_{31} = 82$	$\sigma_{31} = 1.2$
	$\mu_{32} = 63$	$\sigma_{32} = 5$	$\mu_{32} = 800$	$\sigma_{32} = 3.0$	$\mu_{32} = 85$	$\sigma_{32} = 1.0$
	$\mu_{33} = 78$	$\sigma_{33} = 5$	$\mu_{33} = 830$	$\sigma_{33} = 2.0$	$\mu_{33} = 88$	$\sigma_{33} = 0.9$

Dataset 12:

Filename: Redundant_3000_1

In the following datasets we have depicted redundancy issues which can be effectively taken care of in parameter based clustering and which lead to storage wastage in vector based clustering.

	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
Clust 1 (1000) (330,330,340)	$\mu_{11} = 30$	$\sigma_{11} = 3$	$\mu_{11} = 400$	$\sigma_{11} = 3.0$	$\mu_{11} = 55$	$\sigma_{11} = 1.0$
	$\mu_{12} = 30$	$\sigma_{12} = 3$	$\mu_{12} = 430$	$\sigma_{12} = 3.8$	$\mu_{12} = 57$	$\sigma_{12} = 1.2$
	$\mu_{13} = 55$	$\sigma_{13} = 2.5$	$\mu_{13} = 455$	$\sigma_{13} = 2.0$	$\mu_{13} = 60$	$\sigma_{13} = 0.9$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 310$	$\sigma_{21} = 3.0$	$\mu_{21} = 400$	$\sigma_{21} = 3.0$	$\mu_{21} = 42$	$\sigma_{21} = 0.8$
	$\mu_{22} = 360$	$\sigma_{22} = 3.6$	$\mu_{22} = 450$	$\sigma_{22} = 2.4$	$\mu_{22} =45$	$\sigma_{22} = 1.3$
	$\mu_{23} = 420$	$\sigma_{23} = 4.0$	$\mu_{23} = 510$	$\sigma_{23} = 4.0$	$\mu_{23} = 47$	$\sigma_{23} = 0.6$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 660$	$\sigma_{31} = 2.5$	$\mu_{31} = 760$	$\sigma_{31} = 3.6$	$\mu_{31} = 55$	$\sigma_{31} = 1.2$
	$\mu_{32} = 700$	$\sigma_{32} = 3.9$	$\mu_{32} = 800$	$\sigma_{32} = 3.0$	$\mu_{32} = 47$	$\sigma_{32} = 1.0$
	$\mu_{33} = 740$	$\sigma_{33} = 2.0$	$\mu_{33} = 830$	$\sigma_{33} = 2.0$	$\mu_{33} = 42$	$\sigma_{33} = 0.9$

Dataset 13:

Filename: Partial_redundant_3000_1

Following the above example, some more datasets were generated with different patterns to observe how the algorithms perform in each case. Below is a case where only one parameter has redundancy and others are all discrete.

Clust 1 (1000) (330,330,340)	Cost ($)		Response Time (ms)		Reliability (scale of 100)	
	$\mu_{11} = 150$	$\sigma_{11} = 2$	$\mu_{11} = 100$	$\sigma_{11} = 4$	$\mu_{11} = 45$	$\sigma_{11} = 1.5$
	$\mu_{12} = 170$	$\sigma_{12} = 2.5$	$\mu_{12} = 100$	$\sigma_{12} = 4$	$\mu_{12} = 52$	$\sigma_{12} = 1.5$
	$\mu_{13} = 200$	$\sigma_{13} = 3$	$\mu_{13} = 120$	$\sigma_{13} = 6$	$\mu_{13} = 59$	$\sigma_{13} = 1$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 310$	$\sigma_{21} = 2$	$\mu_{21} = 250$	$\sigma_{21} = 2$	$\mu_{21} = 68$	$\sigma_{21} = 1$
	$\mu_{22} = 325$	$\sigma_{22} = 2$	$\mu_{22} = 275$	$\sigma_{22} = 3$	$\mu_{22} =73$	$\sigma_{22} = 1.5$
	$\mu_{23} = 350$	$\sigma_{23} = 3$	$\mu_{23} = 275$	$\sigma_{23} = 2$	$\mu_{23} = 79$	$\sigma_{23} = 1$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 420$	$\sigma_{31} = 2$	$\mu_{31} = 100$	$\sigma_{31} = 4$	$\mu_{31} = 84$	$\sigma_{31} = 1.4$
	$\mu_{32} = 435$	$\sigma_{32} = 3$	$\mu_{32} = 160$	$\sigma_{32} = 5$	$\mu_{32} = 89$	$\sigma_{32} = 1$
	$\mu_{33} = 460$	$\sigma_{33} = 1.8$	$\mu_{33} = 275$	$\sigma_{33} = 6$	$\mu_{33} = 96$	$\sigma_{33} = 1.2$

Dataset 14:

Filename: fourAttribute_Distinct_distinct_3000_1

The following are specifications for a dataset generated for 4 attributes namely, Cost, Response Time, Reliability and Availability (%).

Clust 1 (1000) (200 x 5 (cost),330,330,340 (res, rel)	Cost ($)		Response Time (ms)		Reliability (scale of 100)		Availability (%)	
	$\mu_{11} = 50$	$\sigma_{11} = 1.2$	$\mu_{11} = 120$	$\sigma_{11} = 1.8$	$\mu_{11} = 65$	$\sigma_{11} = 1.2$	$\mu_{11} = 56$	$\sigma_{11} = 1.2$
	$\mu_{12} = 65$	$\sigma_{12} = 1.5$	$\mu_{12} = 130$	$\sigma_{12} = 1.5$	$\mu_{12} = 75$	$\sigma_{12} = 1.2$	$\mu_{12} = 65$	$\sigma_{12} = 1.2$
	$\mu_{13} = 80$	$\sigma_{13} = 2$	$\mu_{13} = 150$	$\sigma_{13} = 2$	$\mu_{13} = 81$	$\sigma_{13} = 1.1$	$\mu_{13} = 62$	$\sigma_{13} = 1$
	$\mu_{14} = 95$	$\sigma_{14} = 2$						
	$\mu_{15} = 120$	$\sigma_{15} = 2.2$						
Clust 2 (1000) (500 x 2)/ (330,330,340)								
	$\mu_{21} = 170$	$\sigma_{21} = 2.1$	$\mu_{21} = 200$	$\sigma_{21} = 2$	$\mu_{21} = 46$	$\sigma_{21} = 1.5$	$\mu_{21} = 75$	$\sigma_{21} = 1.5$
	$\mu_{22} = 195$	$\sigma_{22} = 1.5$	$\mu_{22} = 220$	$\sigma_{22} = 2.5$	$\mu_{22} = 55$	$\sigma_{22} = 1.8$	$\mu_{22} = 78$	$\sigma_{22} = 0.5$
							$\mu_{23} = 80$	$\sigma_{23} = 1$
Clust 3 (1000) (330,330,340)	$\mu_{31} = 230$	$\sigma_{31} = 2$	$\mu_{31} = 56$	$\sigma_{31} = 1.6$	$\mu_{31} = 87$	$\sigma_{31} = 2$	$\mu_{31} = 92$	$\sigma_{31} = 1.2$
	$\mu_{32} = 240$	$\sigma_{32} = 2.4$	$\mu_{32} = 60$	$\sigma_{32} = 1.8$	$\mu_{32} = 96$	$\sigma_{32} = 1$	$\mu_{32} = 96$	$\sigma_{31} = 1.3$
	$\mu_{33} = 255$	$\sigma_{33} = 2$	$\mu_{33} = 75$	$\sigma_{33} = 2$			$\mu_{33} = 99$	$\sigma_{31} = 0.8$
			$\mu_{34} = 88$	$\sigma_{34} = 2.2$				

(Note: Here are the row distribution of the sub clusters in case they aren't clear above).

Clust 1

Cost: 1-200, 201-400,401-600,601-1000
Resp time: 1-330, 331-660,661-1000
Reliability: 1-330,331-660, 661-1000
Availability: 1-330,331-660, 661-1000

Clust 2

Cost: 1-500, 501-1000
Resp time: 1-500, 501-1000
Reliability: 1-500,501-1000
Availability: 1-330,331-660, 661-1000

Clust 3

Cost: 1-330,331-660, 661-1000
Resp: 1-250, 251-500, 501-750, 751-1000
Reliability: 1-500, 501-1000
Availability: 1-330,331-660, 661-1000

Dataset 15:

fourAttribute_undistinct_3000_1

The following dataset was generated with the same specifications as above, only it has completely overlapping values for each cluster (and hence, no sub-clusters exist, or even if they did, they would still have completely overlapping values).

Clust (3000) (1000,1000,1000)	Cost ($)		Response Time (ms)		Reliability (scale of 100)		Availability (%)	
	$\mu_{11} = 50$	$\sigma_{11} = 20$	$\mu_{11} = 120$	$\sigma_{11} = 30$	$\mu_{11} = 78$	$\sigma_{11} = 10$	$\mu_{11} = 80$	$\sigma_{11} = 5$
	$\mu_{12} = 100$	$\sigma_{12} = 35$	$\mu_{12} = 150$	$\sigma_{12} = 20$	$\mu_{12} = 85$	$\sigma_{12} = 7$	$\mu_{12} = 85$	$\sigma_{12} = 5$
	$\mu_{13} = 75$	$\sigma_{13} = 28$	$\mu_{13} = 180$	$\sigma_{13} = 20$	$\mu_{13} = 90$	$\sigma_{13} = 3$	$\mu_{13} = 88$	$\Sigma_{13} = 3$

Datasets 16:

singleAttribute_distinct_3600_1

The following dataset was generated following the trend for having datasets with different number of attributes. It's been generated for a single attribute (cost) and has distinct clusters and sub clusters. There are four main clusters here, each with 900 data points. Within each cluster three sub clusters have been generated each containing 300 data points. Here, some main clusters have distinct sub clusters and some have indistinct sub clusters. (c1 and c4 have distinct, c2 and c3 have indistinct.)

900 data points (300,300,300)	Cost ($)	
	$\mu_{11} = 55$	$\sigma_{11} = 3$
	$\mu_{12} = 70$	$\sigma_{12} = 2$
	$\mu_{13} = 85$	$\sigma_{13} = 3$
900 data points (300,300,300)		
	$\mu_{21} = 240$	$\sigma_{21} = 4$
	$\mu_{22} = 260$	$\sigma_{22} = 2$
	$\mu_{23} = 275$	$\sigma_{23} = 3$
900 data points (300,300,300)		
	$\mu_{31} = 125$	$\sigma_{31} = 4$
	$\mu_{32} = 140$	$\sigma_{32} = 2$
	$\mu_{33} = 155$	$\sigma_{33} = 3$
900 data points (300,300,300)		
	$\mu_{31} = 325$	$\sigma_{31} = 3$
	$\mu_{32} = 340$	$\sigma_{32} = 3$
	$\mu_{33} = 360$	$\sigma_{33} = 2$

Dataset 17:

Filename: sixAttribute_partial_redundant_3000_1

The following dataset was generated for some additional QoS attributes such as *Accessibility*, which we will define as the probability of successful reach and installation by service requestors at a given point of time. It will be measured in %. Another QoS parameter may be *Security*, which may be defined as the level of privacy a web service provides to its subscribers. These may include confidentiality measures, third party access to user information, message encryption and providing access control. This attribute may be measured on a scale of 100, 100 being the most secure, private and rigid in terms of access control and 1 being the most lenient. The following distribution follows a distinct cluster with redundant values in some attributes, multivariate normal distribution and has been generated for all 6 attributes consisting of 3000 data points.

Clust 1 (1000) (330,330,340)	**Cost ($)**		**Response Time (ms)**		**Reliability (scale of 100)**	
	$\mu_{11} = 150$	$\sigma_{11} = 2$	$\mu_{11} = 100$	$\sigma_{11} = 3$	$\mu_{11} = 45$	$\sigma_{11} = 2$
	$\mu_{12} = 170$	$\sigma_{12} = 2$	$\mu_{12} = 100$	$\sigma_{12} = 3$	$\mu_{12} = 50$	$\sigma_{12} = 1.5$
	$\mu_{13} = 200$	$\sigma_{13} = 2.5$	$\mu_{13} = 120$	$\sigma_{13} = 2$	$\mu_{13} = 55$	$\sigma_{13} = 1$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 310$	$\sigma_{21} = 2$	$\mu_{21} = 250$	$\sigma_{21} = 3$	$\mu_{21} = 64$	$\sigma_{21} = 1.2$
	$\mu_{22} = 325$	$\sigma_{22} = 3$	$\mu_{22} = 275$	$\sigma_{22} = 2$	$\mu_{22} =69$	$\sigma_{22} = 1$
	$\mu_{23} = 350$	$\sigma_{23} = 3.5$	$\mu_{23} = 275$	$\sigma_{23} = 2.5$	$\mu_{23} = 76$	$\sigma_{23} = 1.2$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 420$	$\sigma_{31} = 2$	$\mu_{31} = 100$	$\sigma_{31} = 4$	$\mu_{31} = 83$	$\sigma_{31} = 1$
	$\mu_{32} = 435$	$\sigma_{32} = 3$	$\mu_{32} = 160$	$\sigma_{32} = 3$	$\mu_{32} = 89$	$\sigma_{32} = 0.5$
	$\mu_{33} = 460$	$\sigma_{33} = 3$	$\mu_{33} = 275$	$\sigma_{33} = 3$	$\mu_{33} = 95$	$\sigma_{33} = 1$
Clust 1 (1000) (330,330,340)	**Availability (%)**		**Accessibility (%)**		**Security (scale of 100)**	
	$\mu_{11} = 82$	$\sigma_{11} = 1.5$	$\mu_{11} = 100$	$\sigma_{11} = 0$	$\mu_{11} = 48$	$\sigma_{11} = 2$
	$\mu_{12} = 89$	$\sigma_{12} = 1.8$	$\mu_{12} = 92$	$\sigma_{12} = 3$	$\mu_{12} = 54$	$\sigma_{12} = 3$
	$\mu_{13} = 95$	$\sigma_{13} = 1$	$\mu_{13} = 95$	$\sigma_{13} = 1$	$\mu_{13} = 60$	$\sigma_{13} = 3$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 62$	$\sigma_{21} = 1.8$	$\mu_{21} = 58$	$\sigma_{21} = 2$	$\mu_{21} = 72$	$\sigma_{21} = 2$
	$\mu_{22} = 77$	$\sigma_{22} = 1$	$\mu_{22} = 65$	$\sigma_{22} = 2$	$\mu_{22} =77$	$\sigma_{22} = 1.8$
	$\mu_{23} = 70$	$\sigma_{23} = 1.5$	$\mu_{23} = 50$	$\sigma_{23} = 1.8$	$\mu_{23} = 83$	$\sigma_{23} = 2$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 52$	$\sigma_{31} = 1.2$	$\mu_{31} = 75$	$\sigma_{31} = 1.2$	$\mu_{31} = 88$	$\sigma_{31} = 1.1$
	$\mu_{32} = 45$	$\sigma_{32} = 1.2$	$\mu_{32} = 80$	$\sigma_{32} = 0.8$	$\mu_{32} = 96$	$\sigma_{32} = 1$
	$\mu_{33} = 40$	$\sigma_{33} = 1.5$	$\mu_{33} = 85$	$\sigma_{33} = 1.2$	$\mu_{33} =$ **54**	$\sigma_{33} = 3$

Dataset 18:

Filename: nineAttribute_distinct_distinct_3000_1

The following dataset has been generated for nine attributes. It has been generated for distinct clusters and sub clusters. The three additional attributes added are Compliance, Latency and Documentation. Compliance, measured in percentage, relates to the successful accordance of the service i.e. when the % of times the users' requests have been understood and complied with successfully. Latency, measured in ms, is the delay time of the service itself (it is different from response time, as response time is the actual time taken by the service to respond to the users' request). Documentation is measured on a scale of 100.

Clust 1 (1000) (330,330,340)	**Cost ($)**		**Response Time (ms)**		**Reliability (scale of 100)**	
	$\mu_{11} = 70$	$\sigma_{11} = 1.8$	$\mu_{11} = 15$	$\sigma_{11} = 1.8$	$\mu_{11} = 45$	$\sigma_{11} = 2$
	$\mu_{12} = 95$	$\sigma_{12} = 1.5$	$\mu_{12} = 30$	$\sigma_{12} = 1.5$	$\mu_{12} = 50$	$\sigma_{12} = 1.5$
	$\mu_{13} = 115$	$\sigma_{13} = 1.5$	$\mu_{13} = 50$	$\sigma_{13} = 2$	$\mu_{13} = 55$	$\sigma_{13} = 1$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 250$	$\sigma_{21} = 2$	$\mu_{21} = 250$	$\sigma_{21} = 2$	$\mu_{21} = 64$	$\sigma_{21} = 1.2$
	$\mu_{22} = 270$	$\sigma_{22} = 1.5$	$\mu_{22} = 275$	$\sigma_{22} = 1.8$	$\mu_{22} =69$	$\sigma_{22} = 1$
	$\mu_{23} = 285$	$\sigma_{23} = 1.8$	$\mu_{23} = 288$	$\sigma_{23} = 2$	$\mu_{23} = 76$	$\sigma_{23} = 1.2$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 340$	$\sigma_{31} = 1.8$	$\mu_{31} = 100$	$\sigma_{31} = 2$	$\mu_{31} = 83$	$\sigma_{31} = 1$
	$\mu_{32} = 365$	$\sigma_{32} = 2$	$\mu_{32} = 120$	$\sigma_{32} = 1.5$	$\mu_{32} = 89$	$\sigma_{32} = 0.5$
	$\mu_{33} = 380$	$\sigma_{33} = 2$	$\mu_{33} = 140$	$\sigma_{33} = 2$	$\mu_{33} = 95$	$\sigma_{33} = 1$
Clust 1 (1000) (330,330,340)	**Availability (%)**		**Accessibility (%)**		**Security (scale of 100)**	
	$\mu_{11} = 45$	$\sigma_{11} = 1.2$	$\mu_{11} = 46$	$\sigma_{11} = 1.8$	$\mu_{11} = 45$	$\sigma_{11} = 2$
	$\mu_{12} = 52$	$\sigma_{12} = 1$	$\mu_{12} = 51$	$\sigma_{12} = 1.5$	$\mu_{12} = 50$	$\sigma_{12} = 1.5$
	$\mu_{13} = 57$	$\sigma_{13} = 1$	$\mu_{13} = 57$	$\sigma_{13} = 1$	$\mu_{13} = 55$	$\sigma_{13} = 1$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 66$	$\sigma_{21} = 1.3$	$\mu_{21} = 67$	$\sigma_{21} = 1.5$	$\mu_{21} = 64$	$\sigma_{21} = 1.2$
	$\mu_{22} = 71$	$\sigma_{22} = 1$	$\mu_{22} = 74$	$\sigma_{22} = 1.2$	$\mu_{22} =69$	$\sigma_{22} = 1$
	$\mu_{23} = 77$	$\sigma_{23} = 1.2$	$\mu_{23} = 79$	$\sigma_{23} = 1$	$\mu_{23} = 76$	$\sigma_{23} = 1.2$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 84$	$\sigma_{31} = 1.5$	$\mu_{31} = 87$	$\sigma_{31} = 1.2$	$\mu_{31} = 83$	$\sigma_{31} = 1$
	$\mu_{32} = 90$	$\sigma_{32} = 1$	$\mu_{32} = 92$	$\sigma_{32} = 1$	$\mu_{32} = 89$	$\sigma_{32} = 0.5$
	$\mu_{33} = 96$	$\sigma_{33} = 0.8$	$\mu_{33} = 97$	$\sigma_{33} = 1$	$\mu_{33} = 95$	$\sigma_{33} = 1$
Clust 1 (1000) (330,330,340)	**Compliance (scale of 100)**		**Latency (ms)**		**Documentation (scale of 100)**	
	$\mu_{11} = 82$	$\sigma_{11} = 1.2$	$\mu_{11} = 4$	$\sigma_{11} = 0.5$	$\mu_{11} = 40$	$\sigma_{11} = 1.5$
	$\mu_{12} = 89$	$\sigma_{12} = 1.2$	$\mu_{12} = 10$	$\sigma_{12} = 1.5$	$\mu_{12} = 46$	$\sigma_{12} = 1.5$
	$\mu_{13} = 95$	$\sigma_{13} = 1$	$\mu_{13} = 18$	$\sigma_{13} = 2$	$\mu_{13} = 52$	$\sigma_{13} = 1.5$
Clust 2 (1000) (330,330,340)						
	$\mu_{21} = 60$	$\sigma_{21} = 1.4$	$\mu_{21} = 40$	$\sigma_{21} = 1.5$	$\mu_{21} = 62$	$\sigma_{21} = 1$
	$\mu_{22} = 66$	$\sigma_{22} = 1$	$\mu_{22} = 46$	$\sigma_{22} = 1$	$\mu_{22} =69$	$\sigma_{22} = 1$
	$\mu_{23} = 71$	$\sigma_{23} = 1.5$	$\mu_{23} = 52$	$\sigma_{23} = 1.2$	$\mu_{23} = 75$	$\sigma_{23} = 1.2$
Clust 3 (1000) (330,330,340)						
	$\mu_{31} = 52$	$\sigma_{31} = 1.2$	$\mu_{31} = 75$	$\sigma_{31} = 1.2$	$\mu_{31} = 84$	$\sigma_{31} = 1.1$
	$\mu_{32} = 45$	$\sigma_{32} = 1$	$\mu_{32} = 80$	$\sigma_{32} = 0.8$	$\mu_{32} = 90$	$\sigma_{32} = 1.3$
	$\mu_{33} = 40$	$\sigma_{33} = 1.3$	$\mu_{33} = 85$	$\sigma_{33} = 1.2$	$\mu_{33} = 97$	$\sigma_{33} = 1$

APPENDIX B: SOLUTION CODE

The code used to implement K-means clustering in various forms as Vector-based, Preference-based and weighted clustering has been included in this section. The language used was C#.

The code used to implement K-means clustering is follows:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Windows.Forms;

namespace QoSKMeans
{
   public enum ClusteringType
   {
      Preference,
      Weighted
   }

   public class KMeans
   {
      /// <summary>
      /// Global random number generator (used to select centroids in the first iteration)
      /// </summary>
      public static Random random = new Random();

      /// <summary>
      /// used to check the precision of Cluster Set Sums in two subsequent iterations.
      /// </summary>
      public const double Epsilon = 0.0001;

      /// <summary>
      /// Global instance of current ClusterSet
      /// </summary>
      ClusterSet currentClusterSet;
```

```
        /// <summary>
        /// Global instance of previous ClusterSet
        /// </summary>
        ClusterSet previousClusterSet;

        /// <summary>
        /// Global instance of ProcessedAttribute Class (used in GenerateCentroids, CalculateDistances and PopulateClusters
        /// </summary>
        ProcessedAttribute processedAttribute;

        /// <summary>
        /// The original attribute that was passed to KMeans
        /// </summary>
        Attribute originalAttribute;

        /// <summary>
        /// Calculates given no. of clusters on a given attribute.
        /// </summary>
        /// <param name="attribute"> attribute selected based on the preference order selected in the interface</param>
        /// <param name="numOfClusters"> number of clusters specified by the user in the interface</param>
        /// <returns>The resulting ClusterSet</returns>
        public ClusterSet CalculateKMeans(Attribute attribute, int numOfClusters)
        {
            originalAttribute = attribute;

            //keeps a track of the number of iterations
            int iteration = 0;

            //stores the sum of distances in a Cluster Set
            double previousSumOfClusterSet = 0.0;

            while (true)
            {
                //First iteration
                if (iteration == 0)
                {
                    currentClusterSet = new ClusterSet(attribute.Label, numOfClusters); // Create new clusterset with the attribute label and the 'k' clusters chosen by user

                    processedAttribute = new ProcessedAttribute(attribute); // Checks for duplicates and condenses the original attribute into a processed attribute

                    GenerateCentroids(); // Generates initial centroids randomly
```

```
            List<List<double>> distanceSets = CalculateDistances(); // Calculates the distance between every interval and every centroid

            if (!PopulateClusters(distanceSets)) // Places intervals into clusters with centroids with which they have MINIMUM distance
            {
                MessageBox.Show("Encountered an empty cluster, reseting K-Means");
#if TRACE
                Trace.WriteLine("Encountered an empty cluster, reseting K-Means");
#endif
                continue;
            }

            previousSumOfClusterSet = CalculateDistanceSum(distanceSets); // Calculates the total distance sum of a ClusterSet

            previousClusterSet = currentClusterSet; //Stores the current ClusterSet into previous ClusterSet (for future comparison)

            iteration++; //Moving on to the next iteration
        }
        else
        {
            currentClusterSet = new ClusterSet(attribute.Label, numOfClusters); // Create new clusterset with the attribute label and the 'k' clusters chosen by user

            RecalculateCentroids(); //Calculates new centroids by taking the mean of the previous ClusterSet's centroids

            List<List<double>> distanceSets = CalculateDistances(); // Calculates the distance between every interval and every centroid

            if (!PopulateClusters(distanceSets)) // Places intervals into clusters with centroids with which they have MINIMUM distance
            {
#if TRACE
                Trace.WriteLine("Encountered an empty cluster, reseting K-Means");
#endif
                iteration = 0;
                continue;
            }

            double currentSumOfClusterSet = CalculateDistanceSum(distanceSets); // Calculates the total distance sum of a ClusterSet
```

```
            if (Math.Abs(previousSumOfClusterSet - currentSumOfClusterSet) < double.Epsilon) // Checks if the Previous ClusterSetSum is equal to the current ClusterSetSum (within a threshold)
            {
#if DEBUG
                Trace.WriteLine("Ran for " + (iteration + 1) + " iteration(s)");
#endif
                break;
            }
            else
            {
                previousSumOfClusterSet = currentSumOfClusterSet;
                previousClusterSet = currentClusterSet; //Stores the current ClusterSet into previous ClusterSet (for future comparison)
                iteration++; //next iteration
            }
        }
    }

    return currentClusterSet; //if the Sums of ClusterSets in subsequent iterations are same then the ClusterSet is returned
}

/// <summary>
/// Generates initial centroids randomly
/// It scans through the intervals in a processed attribute and chooses given no. of centroids randomly.
/// Once centroids have been chosen, their copies in the processed Attribute are removed.
/// </summary>
private void GenerateCentroids()
{
    for (int i = 0; i < currentClusterSet.Count; i++)
    {
        int intervalIndex = random.Next(processedAttribute.Count); //chooses a random interval from the processed attribute
        currentClusterSet[i].Centroid = processedAttribute[intervalIndex]; // places it as the centroid of a cluster in a cluster Set

        //if (processedAttribute.IntervalCount[intervalIndex] >= 1)
        //{
        //    processedAttribute.IntervalCount[intervalIndex] -= 1;
        //}
        //else
        //{
        //    processedAttribute.RemoveAt(intervalIndex); //removes chosen intervals (now centroids) from processed Attribute
```

```
            //    processedAttribute.IntervalCount.RemoveAt(intervalIndex);
            //}
        }

        processedAttribute.TrimExcess(); //trims processed Attributes to its actual capacity
        processedAttribute.IntervalCount.TrimExcess();
    }

    /// <summary>
    /// Calculates the distance between every interval and every centroid and stores them in a
list
    /// </summary>
    /// <returns></returns>
    private List<List<double>> CalculateDistances()
    {
        List<List<double>> distances = new List<List<double>>();

        foreach (Interval interval in processedAttribute)
        {
            List<double> intervalDistances = new List<double>();

            foreach (Cluster cluster in currentClusterSet)
            {
                intervalDistances.Add(Interval.Distance(interval, cluster.Centroid));
            }

            distances.Add(intervalDistances);
        }

        return distances;
    }

    /// <summary>
    /// Places intervals into clusters with centroids with which they have MINIMUM distance
    /// Uses the list of distances calculated in CalculateDistances() to find the minimum distance
between an interval
    /// and a centroid and places it in a cluster with that centroid
    /// </summary>
    /// <param name="distances"></param>
    private bool PopulateClusters(List<List<double>> distances)
    {
        for (int i = 0; i < distances.Count; i++)
        {
            double minimumDistance = distances[i][0]; //minimum distance between an interval
and a centroid, set to the first distance as default
            int minimumDistanceIndex = 0; //stores the minimum distance index
```

```
            for (int j = 0; j < distances[i].Count; j++)
            {
                if (Math.Min(minimumDistance, distances[i][j]) < minimumDistance)
                {
                    minimumDistance = Math.Min(minimumDistance, distances[i][j]); // storing the minimum distance between an interval and a centroid
                    minimumDistanceIndex = j; //updating the minimum distance index
                }
            }

            currentClusterSet[minimumDistanceIndex].Add(processedAttribute[i]); // placing the interval in the cluster consisting of the centroid with which it has minimum distance
        }

        foreach (Cluster cluster in currentClusterSet)
        {
            if (cluster.Count == 0)
            {
                processedAttribute = new ProcessedAttribute(originalAttribute);

                return false;
            }
        }

        return true;
    }

    /// <summary>
    /// Calculates new centroids by taking the mean of the previous ClusterSet's centroids
    /// </summary>
    private void RecalculateCentroids()
    {
        List<Interval> intervalSums = new List<Interval>();
        List<int> intervalCounts = new List<int>();

        //calculating the sum of intervals in the previous ClusterSet's clusters
        foreach (Cluster cluster in previousClusterSet)
        {
            Interval sums = new Interval();

            sums.LowerBound = 0.0;
            sums.UpperBound = 0.0;

            foreach (Interval interval in cluster)
            {
```

```
                sums.LowerBound += interval.LowerBound;
                sums.UpperBound += interval.UpperBound;
            }

            intervalCounts.Add(cluster.Count);
            intervalSums.Add(sums);
        }

        //assigning new centroids as the mean of previous ClusterSet's intervals
        for (int i = 0; i < currentClusterSet.Count; i++)
        {
            Interval newCentroid = new Interval();

            newCentroid.LowerBound = intervalSums[i].LowerBound / intervalCounts[i];
            newCentroid.UpperBound = intervalSums[i].UpperBound / intervalCounts[i];

            currentClusterSet[i].Centroid = newCentroid;
        }
    }

    /// <summary>
    /// Calculates the total distance sum of a ClusterSet
    /// </summary>
    /// <param name="distanceSets"> list of distances between each interval and its centroid for
every cluster</param>
    /// <returns></returns>
    private static double CalculateDistanceSum(List<List<double>> distanceSets)
    {
        double sumOfClusterSet = 0;

        foreach (List<double> distances in distanceSets)
        {
            double sumOfCluster = 0;
            foreach (double distance in distances)
            {
                sumOfCluster += distance;
            }
            sumOfClusterSet += sumOfCluster;
        }

        return sumOfClusterSet;
    }
  }
}
```

REFERENCES

1. Abramowicz, W., Haniewicz, K., Kaczmarek, M., & Zyskowsk, D. (2007). Architecture for web services filtering and clustering. *Second International Conference on Internet and Web Applications and Services, ICIW'07, Art. Number: 4222920.* Mauritius.

2. Alrifai, M., Skoutas, D., & Risse, T. (April 2010). Selecting Skyline Services for QoS-based Web Service Composition. *WWW'10 - 19th international conference on World wide web.* Raleigh, North Carolina, USA: ACM.

3. *Amazon Web Services*. (2006). Retrieved May 2011, from Amazon : http://aws.amazon.com/

4. Bajaj, S. e. (2006). *Web Services Policy Framework (WSPolicy).* Retrieved April 30 , 2006, from http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/wspolicy- 2006-03-01.pdf

5. Borzsonyi, S., Kossmann, D., & Stocker, K. (2001). The Skyline Operator. International Conference on Data Enginerring (*ICDE)*, (pp. 421--430).

6. Brugger, N. (2010). Web History - Historical perspective on the World Wide Web. McGraw Hill.

7. C Ding, P. Sambamoorthy. (2009). QoS Browsing for Web Service Selection. *Service Oriented Computing Lecture Notes in Computer Science,Volume 5900/2009* , pp. 285-300.

8. Cao, H., Feng, X., Sun, Y., Zhang, Z., & Wu, Q. (2007). A Service Selection Model with Multiple QoS Constraints on the MMKP. *IFIP International Conference on Network and Parallel Computing* (p. 584). IEEE.

9. Cao, J., Huang, J., Wang, G., & Gu, J. (2009). QoS and Preference based Web Service. *Eighth International Conference on Grid and Cooperative Computing* (pp. 420 - 426). IEEE.

10. Carvalho, F., Brito, P., & Bock, H. (2006). Dynamic clustering for interval data based on L2 distance. *Computational Statistics, Vol. 21, Issue 2* , 231-250.

11. Carvalho, F., Souza, R., Chavent, M., & Lechevallier, M. (2006). Adaptive Hausdorff Distances and Dynamic Clustering of Symbolic Interval Data. *Pattern Recognition Letters, 27(3)* , 167-179.

12. Chavent, M., Carvalho, F. A., Lechevallier, Y., & Verde, R. (2006). New Clustering Methods for Interval Data. *Computational Statistics. Vol. 21, Issue 2* , pp. 211-229.

13. CLARK, D. D. (1992). Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. *SIGCOMM*, (pp. 14-26).

14. Clark, M. (2001). *UDDI Weather Report*. Retrieved from http://www.webservicesarchitect.com/content/articles/clark04.asp

15. Cobb, C. W., & Douglas, P. H. (1928). A Theory of Production. American Economic Review 18 (Supplement).

16. Cruz, R. L. (1995). Quality of service guarantees in virtual circuit switched networks. *Select Areas Commun. 13 (6)* (pp. 1048-1056). IEEE J.

17. Devis, B., Antonellis, V. D., & Melochiori, M. (2004). QoS in Ontology-based Service Classification and discovery. *15th International Workshop on Database and Expert Systems Applications*, (pp. 145-150).

18. Diday, E., & Noirhomme-Fraiture, M. (2008). *Symbolic Data Analysis and the SODUS Software.* New York, USA: Wiley-Interscience.

19. Dong, X., Halevy, A., Madhavan, J., Nemes, E., & Zhang, J. (2004). Similarity Search for Web Services. *30th VLDB Conference* (pp. 372--383). Toronto: Canada.

20. Dong, X., Madhavan, J., & Halevy, A. (2004, December). Mining structures for semantics. *ACM Special Interest Group on Knowledge Discovery and Data Discovery (SIGKDD) Explorations Newsletter* , p. Volume 6 Issue 2.

21. Fan, Z., Zhang, L., Shen, J., & Wang, S. (2010). A User's Preference based Method for Web Service Selection. *Second International Conference on Computer Research and Development* (pp. 39-45). IEEE.

22. Farhana, Z., & Patrick, M. (2011). An Adaptive and Intelligent SLA Negotiation System for Web Services. *IEEE Transactions on Services Computing, Vol 4, Number 1* .

23. Gunther, N. J. (1998). *The Practical Performance Analyst.* McGraw-Hill.

24. Han, J., & Kamber, M. (2001). *Data Mining: Concepts and Techniques.* San Diego: Academic Press.

25. Han, J., Taehwan, K., & J., C. Web Document Clustering By Using Automatic Keyphrase Extraction. *International Conferences on Web Intelligence and Intelligent Agent Technology*.

26. Hardy, A., & Baune, J. (2007). Clustering and Validation of Interval Data. *Selected Contributions in Data Analysis and Classification, Part I* , 69-82.

27. Herssens, C., Jureta, I. J., & Faulkner, S. (2009). Capturing and Using QoS Relationships to Improve Service Selection.

28. IBM Corporation. (2003). *Web Service Level Agreement (WSLA) Language Specification Version 1.0.* Retrieved from http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

29. Irpino, A., & Tontodonato, V. (2006). Clustering reduced interval data using Hausdorff distance. *Computational Statistics, Vol. 21, Issue 2* , 271-288.

30. Keller, A., & Ludwig, H. (March 2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management* , vol. 11, number1.

31. L.Vu, Hauswirth, M., Porto, F., & Aberer, K. (2006). A search engine for QoS-enabled discovery of semantic web services. *International Journal of Business Process Integration and Management* , 244-255.

32. Lamparter, S., Ankolekar, A., Studer, R., & Grimm, S. (2007). Preference based Selection of Highly Configurable Web Services", In Proceedings of the. *16th International Conference on World Wide Web*, (pp. 1013-1022).

33. Li-Li, Q., & Yan, C. (2009). QoS Ontology Based Efficient Web Services Selection. *16th International Conference on Management Science & Engineering* (p. 14). Moscow, Russia: IEEE.

34. Liu, Y., Ngu, A. T., & Zeng, L. (2004). QoS Computation and Policing in Dynamic Web Service. *13th International Conference on World Wide Web* (pp. 66-73). New York: IEEE.

35. Ludwig, H., A., K., Dan, A., & King, R. (March, 2003). A Service Level Agreement Language for Dynamic Electronic Services. *Electronic Commerce Research* , 43-59.

36. MacKay, D. (2003). Chapter 20 - An Example Inference Task: Clustering. In *Information Theory, Inference and Learning Algorithms* (pp. 284–292). Cambridge University Press.

37. Mali, K., & Mitra, S. (2003). Clustering and its validation in a symbolic framework. *Pattern Recognition Letters, Vol. 24* , 2367-2376.

38. Marie, C., & Lechevallier, Y. (2002). Dynamical Clustering Algorithm of Interval Data: Optimization of an Adequacy Criterion Based on Hausdorff Distance. In Sokolowsky, Bock, & H. (Eds.), *Classification, Clustering and Data Analysis* (pp. 53-59). Heidelberg: Springer-Verlag.

39. MathWorks . (n.d.). *MATLAB Products*. Retrieved May 2011, from MathWorks: http://www.mathworks.com

40. Moor, A. d., & Van den Heuvel, W. (2004). Web service selection in virtual communities. *37th Hawaii International Conference on System Sciences.*

41. Peng, W., & Li, T. (2006). Interval Data Clustering with Applications. *18th IEEE International Conference on Tools with Artificial Intelligence*, (pp. 355-362). Arlington, VA.

42. Peng, W., & Li, T. (2006). Interval Data Clustering with Applications. *18th IEEE International Conference on Tools with Artificial Intelligence.*

43. Ran, S. (2003). A Model for Web Services Discovery with QoS. *CSIRO Mathematical and Information Sciences, ACM* .

44. Sambamoorthy, P., Ding C. (2009). Thesis Dissertation:" Interactive QoS bowsing for web service selection." Ryerson University.

45. Sambamoorthy, P., Ding C. (2009). QoS Browsing for Web Service Selection. *International Conference on Service Oriented Computing - ICSOC*, (pp. 285-300).

46. SAP NEWS DESK. (December 18, 2005). Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort.

47. Seekda corporate author. (2007). *Home: Seekda*. Retrieved March 2010, from Seeka: http://www.webservices.seekda.com

48. Souza, R. d., & Carvalho, F. d. (2004). Clustering of Interval Data Based on City-Block Distances. *Pattern Recognition Letters, Vol. 25, Issue 3* , 353-365.

49. Vu, L.-H., Hauswirth, M., Porto, F., & Aberer, K. (2006). A Search Engine for QoS-enabled Discovery of Semantic Web Services. *International Journal of Business Process Integration and Management 2006 - Vol. 1, Number4* , 244 - 255.

50. Wang, Y., & Stroulia, E. (2007). Semantic structure matching for assessing web-service similarity. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics Vol. 2910* , pp. 194-207.

51. Wang, Y., & Vassileva, J. (2007). Toward Trust and Reputation Based Web Service Selection: A Survey. *International Transactions on Systems Science and Applications (ITSSA) Journal, Vol. 3* , 118-132.

52. XQ, F., X.W., F., & C-J, J. (2011, 02 26). Research on Web service selection based on cooperative evolution. *Expert Systems with Applications 38* , pp. 9736-9743.

53. Xu, Z., Martin, P., Powley, W., & Zulkernine, F. (2007). "Reputation-Enhanced QoS – based Web Services Discovery. *Proceedings of the IEEE International Conference on Web Services* (pp. 249-256). IEEE.

54. Xu, Z., Martin, P., W.Powley, & Zulkernine, F. (2007). Reputation-Enhanced QoS-based Web Services Discovery. *IEEE International Conference on Web Services* (pp. 249-256). Salt Lake City, Utah: IEEE .