1-1-2011

A customizable web service selection framework using MCDM approaches

Raed Karim
Ryerson University

A CUSTOMIZABLE WEB SERVICE SELECTION FRAMEWORK USING MCDM APPROACHES

by

Raed Karim

B.Sc in Computer Science, Ryerson University, Toronto, 2009.

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.

RAED KARIM

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.

RAED KARIM

BORROWER'S PAGE

Ryerson University requires the signatures of all persons using or photocopying this thesis.

Please sign below, and give address and date.

Name	Signature	Address	Date

A CUSTOMIZABLE WEB SERVICE SELECTION FRAMEWORK USING MCDM APPROACHES

Raed Karim
Master of Science, Computer Science, 2011
Ryerson University

ABSTRACT

With the tremendous increase of web services published online, the problem of selecting the best service offers becomes more challenging. Users need to make their decisions on multiple and conflicting non-functional requirements. It is a natural fit to apply the Multi-Criteria Decision Making (MCDM) theory to the service selection and ranking process. In our proposed QoS-based service selection system, we take the user-centric standpoint to design the system. We improve the original MCDM models so that the user requirements on the QoS criteria are included in the rank calculation process. Our proposed QoS weighting method considers the well-defined ANP method combined with the user- defined weights. We compared the improved selection methods and we found that the Constraint Programming method is the best in terms of its sensitivity to the changes made to the QoS weights. Consequently, the results produced from this comparison would be presented to the user.

ACKNOWLEDGEMENTS

I first would like to express my sincere gratitude to my supervisor Dr. Cherie Ding for her great support, encouragement and guidance throughout the course of my Masters degree. Her ideas and suggestions were invaluable to me in order to complete my degree. I also would like to pay a special thank to the committee members for their valuable suggestions and discussions. I would like to thank Dr. Isaac Woungang for allowing me to use DABNEL lab and for his support as a Graduate Program Director. Also, I would like to thank the Chair of our department Dr. Alireza Sadeghian, Dr. Ali Miri and Dr. Alex Ferworn (as a Graduate Program Director during my first year).

Finally, I would like to greatly thank my wife, Sally Jabbar, and express my unlimited love to her for the invaluable love, long patience and encouragement that she has shown and given to me. Without her support my task would be extremely difficult.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AHP: Analytical Hierarchy Process

ANN: Artificial Neural Network

ANP: Analytical Network Process

CI: Consistency Index

CP: Constraint Programming

CR: Consistency Ratio

CSP: Constraint Satisfaction Problems

DM: Decision Maker

DS: Discovery Service

ELECTRE: ELimination Et Choix Traduisant la REalité (ELimination and Choice Expressing REality)

ERP: Enterprise Resource Planning

ExI: Expert Interface

HTML: HyperText Markup Language

HTTP: Hypertext Transfer Protocol

MCDM: Multi-Criteria Decision Making

MIP: Mixed Integer Programming

PROMETHEE: Preference Ranking Organisation METHod for Enrichment Evaluations

QoS: Quality of Web Service

RMI: Remote Method Invocation

SCSP: Soft Constraint Satisfaction Problems

SOAP: Simple Object Access Protocol

SOA: Service Oriented Architecture

UI: User Interface

UDDI: Universal Description, Discovery and Integration

WSDL: Web Services Description Language

XML: Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1. Background Information

In its early years, the Internet was seen as a standardized medium where the information is transferred among different machines and platforms. The Internet users can view and exchange the information in its simplest way. As the information and e-business transactions have become more complex, the evolution for distributed computing systems becomes the main theme of the computer science and business world. However, the distributed computing (e.g. *Remote Method Invocation RMI*) is not able to cope with the increasing development of the applications as they are tightly coupled and platform dependent. This makes the use of the available applications in the e-business context difficult and costly. The Service Oriented Architecture (SOA) has emerged to enable the development of loosely coupled and platform independent applications and software in an open and heterogeneous computing environment.

One of the main implementations of the SOA is through Web Services. Web services are self contained, loosely coupled, discoverable, autonomous, and dynamic entities that are available in the network. The SOA uses web services to support the development of fast growing, reusable, low cost and interoperable software and applications [1][2]. They can be described, published and invoked among different parties over the network. Web services allow different stand-alone applications to form a coherent system that serves the user's needs.

The architecture of web services is established according to common standards that ensure the successful communication and information exchange among the service requestors, providers

and service agencies. The service applications over the network can communicate with each other by complying with these standards. There are four standards in the web service architecture: (1) SOAP is the low level protocol that acts as a vehicle to convey messages using the HTTP protocol as its road. It defines an envelope to carry messages and rules; (2) XML is the standard language for the web services. Its tags carry the meaning of the information in a certain degree. This is in comparison to the HTML in which the information can only be displayed and presented syntactically. The main advantage of XML technology is that it supports the data in different formats (documents or databases); (3) WSDL is an XML-based language that describes web services. Basically, for a web service to be discovered it must be described and it must have a defined interface. It is the role of the WSDL as an advertisement tool; (4) Directory Service (DS) is a repository that can be implemented as a database where information about web service providers, requesters and other software agents is stored and organized. The UDDI technology is the most popular technology to implement the Directory Service. [3].

The web service architecture consists of three essential operations: publish, find and bind. The three main actors of these operations and activities are: service providers, service registry (can be represented by a service broker), and service requester (the user). These components are not independent and they maintain consistent relationships. The relationships depict the web service activities over the network. The service provider owns the service. It provides a description and publishes it to the service registry (e.g. UDDI). The service requester finds the desired service in the service registry through the service descriptions. The final step is when the requester binds with the service provider to invoke the service according to the service description [4]. Figure 1.1 shows the general architecture model of web services.

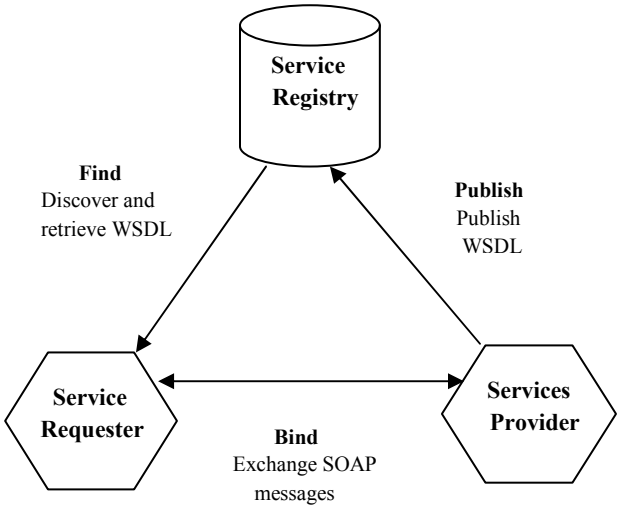

Figure 1.1-The basic architecture of web services [5].

1.2. Web Services Discovery and Selection

The web services are published into a service repository in the network. The next operation is to discover these web services; it is performed with the assistance of the WSDL documents associated with each web service. In the discovery stage, the functional requirements (i.e. functions or operations of the service, input/output parameters, etc) are the main players that determine if a web service is relevant according to the user's query. There is a very high chance that a long list of web services is returned to the user which satisfies the functional requirements of the user request.

After discovering the desired web services, the next step is to select a best service from this list. The "best" is usually measured by how a service satisfies the constraints set by the actual requester. In the web service selection phase, the obtained services from the discovery process are filtered out based on their satisfaction degrees to the user's non-functional requirements. The latter is represented by the QoS properties of web services.

The QoS can play a crucial role in determining the optimal service offer or a list of the optimal service offers. They technically represent the user constraints regarding the quality of a service the user wishes to have. Failing to (partially or completely) satisfy these constraints will negatively impact a service offer during the selection phase [6]. The user expectation with respect to his/her request must be considered. For example, the user might request the offer has the ability to provide a certain level of reliability (Reliability > 98%) and a response time (Response Time <1.5 sec), has a reasonable cost (Cost <= $15), and many other constraints.

1.3. Multi Criteria Decision Making (MCDM)

In our proposed solution to the problem of QoS-based web service selection, we consider different methods of the MCDM methodology. The MCDM is a theory that explains the mechanism of the decision making process. It provides systematic and proven methods to perform the decision process through comparison, evaluation and ranking the available objects in a problem's domain. In many cases, the decision maker (DM) has to deal with a problem that has multiple and conflicting criteria and different alternatives to choose from. There are three key routines in the MCDM process. The DM defines the problem's main components and its goal. He/she defines a set of service alternatives that are available and can fit into the solution. And he/she defines a set of criteria that determine the performance of the defined alternatives [7].

Generally, there are two distinctive decision making schools [8]: the French and American schools. The French school is based on the outranking relations that are established on the pair-wise comparisons for each pair of alternatives. These relations are governed by pre-determined thresholds that determine the preference boundaries. There are two main methods that belong to this school: ELECTRE (Elimination and Choice Expressing REality) and PROMETHEE

(Preference Ranking Organization METHod for Enrichment Evaluations). The American school, on the other hand, is based on finding the overall performance of each alternative through the use of an objective function. The pair-wise comparisons determine the priority of which the criteria are involved in the decision process. The two most popular methods are the Analytical Hierarchy Process (AHP), and the Analytical Network Process (ANP).

In our research study, we incorporate the QoS-based web service selection and ranking process with the theory and principles of MCDM methodology. We improve the process of the QoS weighting and the service selection and ranking to produce more satisfactory results from the user's perspectives. Our case study clearly illustrates the detailed steps to achieve our goal in solving our problem.

1.4. Research Motivation

Web service users often need to make their decisions on multiple and conflicting non-functional (QoS) criteria associated with the service offers when choosing among the functionally relevant services. Thus, the MCDM theory is a natural fit to the problem of QoS-based web service selection. It provides the essential principles to interpret the user' judgment and preference into quantitative figures through applying formal methods to reach the user' target goal (i.e. selecting a set of optimal services). However, the different MCDM methods that we use to solve our problem involve a certain degree of complexity and require a high workload from the user. This represents a great challenge that prevents the normal service users from a real involvement in the selection process.

In the QoS-based web service selection, the user position is a principle axis for which any solution is considered acceptable. In most of the existing research works, the selection task is to

find a service alternative that is the best in terms of optimizing the overall QoS criteria; however, it may not satisfy the user requirements on the non-functional criteria.

Moreover, most of the research works in this field assume the independency between the QoS criteria. This is not very accurate as the interdependency and feedback relationships exist between the criteria; specifically between the QoS properties. For example, a more abstract property may rely on a more primitive property, a subjective property may be decided by some objective property, two QoS properties may be dependent on each other, etc. These are real world cases that we cannot ignore, and they do have a great impact on the overall decision process.

Finally, various approaches have been introduced to solve the problem of QoS-based web service selection; however, no comparison and evaluation processes have been performed in order to find the most suitable selection method or algorithm.

1.5. Research Objective

The purpose of our research is to investigate and analyze the existing approaches for the problem of web service selection. We then propose our web service selection and ranking framework based on the QoS properties and user information. The proposed framework is a customizable tool that is able to address and solve some of the shortcomings in the current models and approaches.

Our research objective is to develop a customizable and flexible QoS-based web service selection and ranking system. It handles the user preferences in a more efficient and well-defined way. In order to overcome the complexity of using the MCDM based selection method we take into account the different levels of user experience and personal environment. In this context, the

service consumers could be an expert or regular consumer; he/she might be willing to directly or indirectly state his/her priorities. The user workload can be reduced by allowing a web service expert involvement in setting a default state of the selection system.

Our enhancements to the selection algorithms significantly improve the selection and ranking process to better suit the service selection problem. The proposed enhancements reflect the fact that the user requirements and constraints have the major role in the selection process of the relevant web services. The overall solution has been improved to meet the user's expectation. We demonstrate the importance of interdependency and feedback in prioritizing the QoS properties and modeling the general service selection process. The overall system efficiency has been preserved through profiling components that allow the users to save their preferences for future requests. The default priorities that encompass the underlying weighting scheme are also saved so that it can be used to finalize the criteria weights. A sensitivity analysis is performed in order to compare and assess the various selection methods.

1.6. Research Contributions

The core of our research contribution is the comprehensive improvement to the existing approaches, through our proposed web service selection framework. Below we explain the major contributions that feature our research work.

- We propose a flexible and customizable weighting method. Our service selection system includes two weighting schemes: (1) the system default weighting scheme; (2) the user defined weighting scheme. The ANP method is used to generate the former. The AHP method or a simple weighting method is used to generate the latter.. The reason for using multiple weighting methods is to accommodate the different levels of the user's expertise

and knowledge. Also, not all users are willing to spend too much time and efforts in this process.

- The most important characteristic of our proposed framework is the inclusion of users' non-functional requirements (constraints) in the selection and ranking process. We improve the original MCDM algorithms through a penalty function and a layer-based ranking algorithm. In the current selection systems, the user requirements are usually not considered during the ranking process. However in our system, the non-satisfying services will be penalized so that their ranking orders will be affected compared to the satisfying services.
- We consider the interdependency and feedback relationships among the various QoS properties. In our research, we try to identify the possible and indisputable dependency relationships between the QoS properties based on their definitions. This important feature is closer to the real world scenarios and it is often not considered in many approaches.
- Our proposed service selection system provides a functionality to compare the different selection and ranking methods. It measures the sensitivity degree of each method in terms of the changes we make in the QoS weights. Our system can recommend the ranked service list produced from the method that has a minimum sensitivity score.

1.7. Organization of Thesis

The rest of the thesis is organized as follows.

In Chapter 2, we review and analyze the existing research work in the field of web service selection. We identify different approaches in our research: (1) the MCDM based approaches, (2) the Constraint Programming (CP) approach; (3) the Utility Function based approach; (4) the

Trust based service selection; (5) and other approaches such as the Mixed Integer Programming (MIP) and Skyline based methods.

In Chapter 3, we introduce our proposed framework for the QoS-based web service selection using different MCDM approaches. We explain the methodology in producing the relative priorities (weights) of the QoS criteria and in selecting and ranking the service offers through our customizable and flexible web service selection framework. We also present our proposed framework architecture and the flow of our system operation using a sequence diagram. In this chapter, we thoroughly explain how we handle the user preferences and how the user non-functional requirements can play a crucial role in the process of selecting and ranking the services. This is done through some significant improvements to the original methods. We then present our sensitivity analysis procedure in order to compare and evaluate the different selection methods that we use in our proposed framework. Finally, the time complexity of the original methods and our enhancement procedures is presented.

In Chapter 4, we present our comprehensive case study to illustrate our proposed framework operations. We take the service selection system from the initial stage through the user and the service expert inputs towards producing multiple service ranked lists of the optimal service offers. Then we introduce to the user a single ranked list.

In our case study, we use nine service providers (alternatives) and nine QoS properties (criteria). We illustrate the detailed steps of weighting the QoS criteria and selecting and ranking the service. Then, we describe in detailed steps how our improvements make a significant contribution to the QoS-based web service selection system.

In Chapter 5, we present our conclusions by emphasizing the significant improvements in the original selection and ranking methods. A summary of our research contributions to the field of QoS-based web service selection is presented. Finally, a list of potential and future work is presented as well.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

In the last decade, numerous research works have been presented dealing with the problem of QoS-based web service selection. Different groups of researchers took different approaches to study and solve the problem. During our investigation, we have observed that there is no consensus on a specific research path with respect to the problem of web service selection. In this chapter, we survey and categorize the QoS-based web service selection related approaches into different classes: (1) the MCDM based approaches, (2) the CP approach; (3) the Utility Function-based approach; (4) the Trust based service selection; (5) and other approaches such as the MIP and Skyline-based methods.

2.2. MCDM Based Approaches

The MCDM is the process of choosing among multiple alternatives the one that: (1) has a highest chance to be effective and (2) is the best fit with the DM goal and preferences. Different MCDM methods have been applied to a wide range of real world problems. In this section, we review the MCDM applications on web service selection and other research fields. We also pay a special attention to some of the combination approaches that tackle our problem.

2.2.1. QoS-Based Web Service Selection

MCDM methodology has been used in recent years to solve the problem of selecting the optimal web services based on the functional and non-functional requirements. The rational for using the MCDM methodology is that the web service selection is actually a multi-criteria

decision problem. The user has to select the best service based on different and conflicting criteria.

In [9], a new QoS ontology for web services called WS-QoSOnto was proposed. The proposed ontology had multiple facets to describe QoS properties, metrics, tendencies and relationships. To define the user preferences, the AHP method was used. In this method, the problem's components and elements were decomposed into multiple levels based on how the DM views the problem's domain. In this paper the QoS properties have been categorized into mandatory and optional attributes in a hierarchical structure. Then their priorities (weights) have been computed through the pair-wise comparisons in each category.

The pair-wise comparison process was the main step in the AHP algorithm. It is based on determining how a QoS property is more important than the other in terms of achieving the parent or main goal. Eigenvector and Eigen value were calculated in each category of the QoS properties to obtain the relative weights. The service relative ranking for the QoS properties were aggregated by first forming the service relative ranking matrix and then multiplying the matrix values by the normalized weight vector of a category of QoS properties. Finally the categories were aggregated with their weights from bottom to top according to the hierarchical levels created by the DM. The result was a vector of the candidate web services that was sorted to obtain the final services ranking. In their proposed model, the selection framework did not include the user constraints over the QoS properties in order to determine how each service satisfies the QoS properties.

In [10], the ANP method was used for the problem of web service selection. The ANP method is a generalized form of the AHP. In this paper, the authors emphasized on the interdependency relationship between the QoS properties as an important feature to be consider.

As part of the ANP algorithm, the QoS properties have been categorized into three clusters: runtime, security and configuration management. The dependency network was then defined for these properties. The steps of the algorithm included building the unweighted matrix, cluster matrix, weighted matrix and the limiting matrix. Then the weights obtained from the limiting matrix were synthesized with services evaluations to obtain the service ranks. The final result was a list of optimal services. The ANP algorithm was applied directly to the problem without any modification to make it suitable to the problem domain and its specific requirements. Again, in their proposed model, the QoS requirements determined by the user were not included in the ranking process.

In [11] a generic solution was proposed using PROMETHEE method. It was compatible with any existing selection approaches. The PROMETHEE method was used to obtain the global priority rather than introducing the final ranking to the user. The weighting scheme used in this model was an extension of the Simos method. In PROMETHEE method, the outranking technique was the way of generating the preference relationship between each two candidates. Using PROMETHEE algorithm the positive and negative flow ranking and then the net flow ranking were computed. The net flow represented the overall performance of a service alternative; it was used to rank the list of the services. A subjective selection of a first 40% of the services was used to represent the global priority constraint of the best services.

In [12], the PROMETHEE method was also adopted for selecting the best web services. Similar to [11], the PROMETHEE algorithm steps were directly followed to obtain the three outranking flows (positive, negative and the net flows). Their selection model had subsequent phases (similar to the waterfall model of the traditional software engineering). The flow of selecting best services began with identifying the QoS, selecting, evaluating them and then

making the decision. In this paper, equal weights have been assigned to the QoS properties supported by the proposed model. Also, the author used different levels to represent the QoS constraints. The offered service could meet any value under one of these levels to be considered a satisfying service.

In [13], the AHP method was applied directly to select the best web service. The non-functional properties were not considered; rather the service implementation, business and functional aspects were included. The authors did not try to modify the original algorithm in order to consider the user request on the QoS constraints in the ranking process.

2.2.2. Combining Two MCDM Methods

The approach of combining different weighting and ranking methods and algorithms is not new in solving real world problems. In the field of web service selection, a very few research approaches take the direction of combining two or more algorithms in order to weight criteria and rank web services.

In [14], the authors successfully pointed out some important features of both AHP and PROMETHEE methods. The proposed combination approach came after detailed comparisons that recognized the positive and negative sides of both methods. In this approach, the AHP method was used to compute the priorities (weights) of the criteria. The PROMETHEE method was used to rank the best services.

In [15], the two popular MCDM methods; AHP and PROMETHEE have been combined. The AHP method was used to determine the weights of the criteria and PROMETHEE was used for ranking the health care web services. The original PROMETHEE algorithm was extended to include the fuzzy numbers. The paper did not present an experiment or a detailed case study to analyze and evaluate the potential results. The outcome of the combination approach could

generate more than one ranked list of the best web services. These were AHP-based, PROMETHEE- based and AHP/PROMETHEE-based service selection lists.

2.2.3. MCDM Approaches for Software and Products Selection

In this section, we investigate the MCDM methods used to solve some selected problems related to software, network and product selection. They are similar problems to the web service selection and web service in a way can also be deemed as software.

In [16], the problem of enterprise resource planning software selection was modeled based on the ANP method. The unique characteristic of the latter is its ability to model the interdependency relationships among the problem's criteria. The artificial neural network (ANN) model was trained on the ANP results in order to be used for the selection of ERP for the next new decision. According to the authors, the ANP method was the most suitable technique to use for the ERP selection problem compared with other MCDM methods such as the AHP method. The DM could consider the tangible and intangible factors. It also represented the qualitative information by using the numerical forms so that it is easy for the DM to understand and follow.

The ANP method was also used in [17] to assist in selecting the appropriate software that can be incorporated in the product development process. The authors attempted to assert the fact that using the MCDM-based approach has a better impact than using a general knowledge to decide about a certain problem. Choosing the ANP method, in this paper, was to measure the impact of the IT technology (software) on the task productivity, innovation, customer satisfaction and management control. This was done by considering the inter-component integration and compatibility with respect to the system functions. In the first stage of the proposed model, the standard AHP was used. It was further extended to analyze the inter-functional evaluation by

representing the feedbacks among the software functions with the combination of the alternatives as controlling components.

2.3. CP Based Web Service Selection

CP can be described as a powerful computational paradigm to solve problems that span in different scientific techniques such as computer science, operation research and artificial intelligence. The CP has been widely applied in real life problems such as planning, bioinformatics, networks, etc [18].

In [19], the definition of the Constraint Satisfaction Problems (CSP) was extended by replacing the crisp constraints with soft constraints (SCSP). For this purpose, penalties have been applied on the services that violate the agreed upon constraints. The proposed model was an attempt to solve the problem of the CP by including the partially and fully violating alternatives in the set of the solution. Usually this type of alternatives was excluded during the selection process. This approach seems close to our idea in terms of proposing a way to include this type of alternatives in the selection process. Although applying a penalty function is a common solution to different problems, altering the definition of the CS is not necessary to happen. The CSP technique has already taken into account the QoS constraints (as it is explained in Section 3.6.3).

Our strategy, as we explain it later, is based on defining some decision rules. The non satisfying services are grouped in layers according to their degrees of satisfying the user's requirements on the services' non-functional properties.

In [20], a procurement framework of web services was proposed. Procurement means finding the best web services among the offered ones according to the user demand. The

proposed framework was based on CP that allows checking the consistency and conformance between demands and offers. A constraint solver implements the checking process in order for the matchmaker to return the optimal solution. If no offer, that meets the user demand, is available, then a failure message is produced.

The main drawback of the CP approaches is the inaccuracy of building a set of optimal services that optimize the overall QoS criteria. The reason for this is the fact that the services that partially do not satisfy QoS properties are excluded from the selection procedure. This is not always realistic in the problem of selecting the best web services. The consumers are sometimes interested in making trade off during the service selection with the consideration of QoS of web services. We need to investigate about how the non satisfactory services can be included in the selection process.

2.4. Utility Function Based Approach

The use of utility functions was considered in [21]. The utility function is a normalized function whose value range is [0, 1]. Its domain is a QoS property. The user preference over the value of the QoS property is represented in this domain. In their proposed model, the utility functions were semantically defined, and the ontology used for this purpose extended the semantic framework from [22]. We noticed that the authors did not consider the minimum or the maximum computation of the utility assessment. However, a composition of different utility functions (one function for each QoS property) was exploited to compute a global utility value. Each utility function had a weight associated to it to determine the importance of each QoS property.

In [23], an adaptive selection framework was proposed in which the user defined a utility function that described a quality of service attribute. Then a proposed learning policy was used to

explore and exploit interesting services. This is to maximize the user's utility function according to his requirements. The proposed framework benefited from trust-based models to collect quality information regarding the desired services. After calculating the utility function of each quality, the user tries to contact the provider that is able to maximize his utility function. The learning aspect of the proposed model implied a trade off process between attempting to find an alternative that provides a higher optimization and making decision based on the current knowledge. The proposed model had an advantage by using a learning policy to collect the quality information using trust models. However, using a utility function to describe the user's preferences could be doubtful in terms of its accuracy to define his/her preferences. Also, not all quality types can be easily described through a utility function; especially those that are difficult to collect their information even through a learning mechanism.

2.5. Trustworthiness Based Approach

In [24], a semantic-based trustworthy framework was proposed for web service discovery and selection. The system mainly used the users' feedback and defined the service reputation in order to evaluate and select the best services. The authors claimed that the service selection based on other QoS attributes was not always the right decision for different reasons such as changing the service environment. In reality, considering only the service reputation and feedback could also be unreliable. The feedbacks given by other users are not all dependable; they are subjective actions that have no rules to comply with.

In [25], the same concept was used by adopting the service reputation in the service selection stage. In this stage, the user's feedback was considered to predict the service quality in the future. The prediction was also established according to the service provider's information. This work made two assumptions. First, a probabilistic behavior of the service and users was

considered. There were always differences between the actual QoS values and the ones the user reports. Second, a third party was used to provide credible QoS information and to evaluate the behavior of the users.

The main issue with both studies is the great reliance on service reputation and user feedback. In [25], the authors explained that despite of the accurate result that could be obtained from a third party, the information could be costly to both the service provider and requester. In our opinion, the service reputation could contribute towards solving the problem of web service selection but in lesser degree compared to the other important QoS properties (i.e. availability, response time, throughput, etc). The latter are more objective attributes than reputation.

2.6. Other Approaches

The MIP method and Skyline computation were other areas of our investigation. MIP was also used to solve the problem of web service selection. In [26], for the QoS specifications that contain only linear constraints, MIP could be implemented whereas CP could be used for non-linear constraints QoS specifications. Based on experiments conducted in this paper, the MIP approach outperformed the CP one when considering linear constraints. The process of matchmaking was achieved using MIP or a CP engine depending on the type of constraints (linear or nonlinear, respectively). In this case it was worth to build two implementations for the matchmaking algorithm for the sake of efficiency and performance.

The Skyline computation was another approach to solve the problem of QoS-based web service selection. In [27], the authors proposed a solution to deal with two issues in QoS-based web service selection; the QoS preferences set by the user and the dynamic environment of the QoS properties. The Skyline computation can be used so that providing preferences to the problem criteria is not mandatory for the users. In this paper a complementary tool was used to

include the desired services. It was called p-dominant service skyline. It was based on the domination relationship between service providers which is similar to the process of PROMETHEE method. The p-dominant service skyline was a threshold that determined whether a service provider will be included in the desired services list. One of the main issues with this proposed approach is that the user-defined preferences were not considered. In our opinion, the user preference is crucial for the web service selection because it controls the trade-off process of the optimization problem.

2.7. Summary

In this chapter, we reviewed and analyzed various approaches of the QoS-based web service selection. The research field is dynamic and rapidly growing. Many researchers are working in different directions in order to provide promising solutions. In our survey study we categorized the existing works into multiple groups; the MCDM base approaches, the CP approach, the Utility Function based approach, trust-based approaches and some other important approaches such as the MIP and the Skyline method.

Based on our review and analysis of the above research studies, we can identify several shortcomings. In most of these approaches, the selection procedure is formulated in such a way that the selection algorithm is applied directly. The interdependency relationships among the QoS properties have not been clearly defined and involved in the process of web service selection (except [10]). This aspect indeed reflects the real world problems' characteristics. Moreover, it has a major impact on the decision making process and the final solution.

The second major shortcoming is that many of the existing approaches did not consider the actual user's requirement on the QoS criteria. They considered service selection and ranking as an optimization problem; thus, the user requirements were not necessary to be included in the

service ranking process. However, in our opinion, it is very important to get the user request involved in the computation process of selecting and ranking the service offers. Moreover, the offers must be penalized for not satisfying the user requirements. In the selection and ranking process, the optimization process needs to be complemented by the satisfaction degree of a service to user's QoS request in order to meet the user expectation.

The other important issue with the reviewed works is the missing of the systematic comparisons to measure the stability or the accuracy of the selection and ranking methods. In MCDM process, we need to make sure that the obtained solution is stable enough in spite of the changes that happen in any of the parameters. We also observe that less attention was paid to the fact that the service users not necessarily have the sufficient expertise to perform the weighting procedure of the criteria. In our opinion, any service selection system should consider reducing the workload from the user. Also the system should provide the flexibility so that the users can provide their preferences in a way that is convenient to them.

In our proposed solution, we try to resolve and address these shortcomings. Our proposed framework considers important theories and tools that can help the DM obtain a robust and reliable solution. Firstly, we take into account the dependency and feedback relationships among the participating QoS properties. Secondly, we include the user requirements in the service selection and ranking computation process. Thirdly, the user preferences and the weighting methods have been systematically represented. Our weighting scheme is flexible in terms of representing the user preferences and accommodating the user's knowledge. Also, our framework provides a powerful tool to compare and evaluate the service selection methods. A sensitivity analysis was conducted to measure the sensitivity degree of the selection methods

used in our proposed model. The solution produced form the less sensitive method can be presented to the user.

CHAPTER 3

A CUSTOMIZABLE WEB SERVICE SELECTION SYSTEM

3.1. The Web Service Selection Based on Non-Functional Criteria

At the discovery stage, we assume that the functionally matching services have been obtained. This can be performed using a central registry or a P2P discovery mechanism [28]. The former is a centralized repository for web services where service providers publish and store their service descriptions. They can be later located and invoked by a service requestor. The latter is a non-centralized structure where web services are nodes in the networks. At the discovery stage, the service requester queries for an appropriate web service that matches the functional requirements specified by the requester. Then, either a match is found or the query is propagated through the network searching for an appropriate match.

In Chapter 1, we discuss about the meaning of the QoS- based web service selection and ranking. The main task is to find a list of optimal services based on the user requirements on various QoS properties. The MCDM methods are used to solve this optimization problem. The first step in QoS-based service selection is weighting or prioritizing the QoS properties using a systematic and well defined weighting scheme. The second step, at the selection stage, is selecting and ranking the optimal services using the MCDM methods. The information provided by the user has become an essential part of the service weighting and selection stages. We take a user-centric view in terms of weighting the QoS criteria and ranking the selected services. Our proposed framework is customizable to accommodate different user experience levels and how much time and effort users wish to take in the service selection process.

It is important to mention that the methods, we use, are classified according to the purpose of using them in our research. The fist group includes the ANP [29], the AHP [30] and a simple weighting method. They are used to weight the QoS criteria and also to rank the services (except the simple method). The second group includes the PROMETHEE and CP methods. These two methods do not provide a weighting scheme; we use them in the step of selecting and ranking the service offers. In our research study, we also introduce a combination method of different weighting schemes. In the selection and ranking stage, we use a sensitivity analysis procedure to evaluate and compare the different selection methods from the first and second group. The goal is to produce a single ranked list of the optimal service offers based on the stability of the method.

3.2. The Architecture of Our Service Selection System

In our system architecture (shown in Figure 3.1) the **input interface** to our service selection system has two parts- Expert Interface (ExI) and User interface (UI). The **ExI component** allows for an expert on web services and QoS requirements to get involved in the process. The expert's task is to define the interdependency and feedback relationships among the QoS properties in the system. The output is a matrix that illustrates these dependencies so that it can be used to generate weights for the QoS properties. **The ANP weighting component** will use the ANP method for this purpose. The generated QoS weights are saved in the **default weighting component** because they are the default weights used in the system. This default weighting process is done only once, and then every time a user visits the system or a new query is submitted the saved default weights are used. When the user requirements or experience change or the domain knowledge is updated, the default weights can be changed by the expert user.

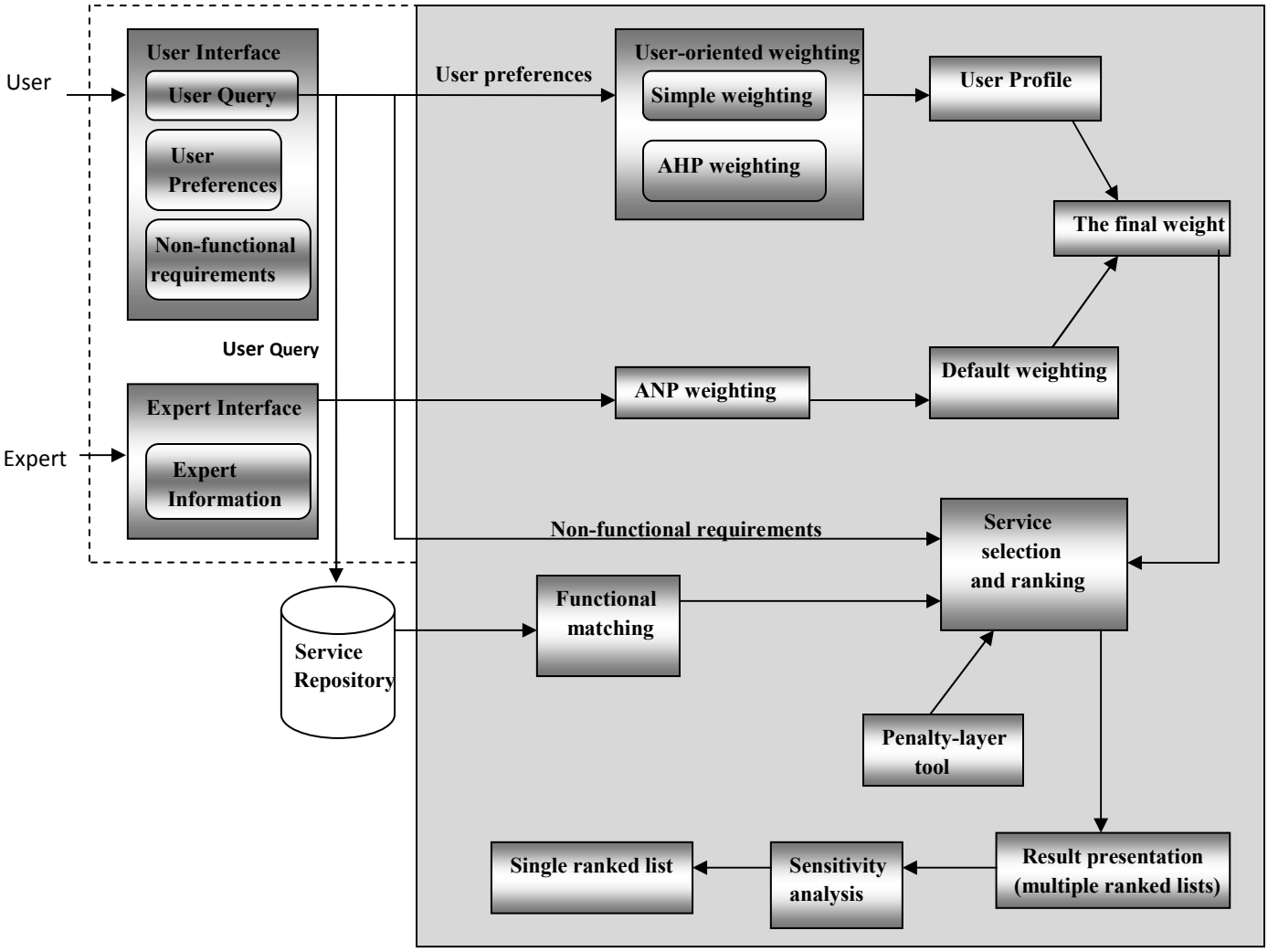

Figure 3.1- The service selection architecture model.

Through the UI component the user can perform three main tasks (later we show that it is not necessary to perform these three tasks every time he/she visits the system). First, the user needs to provide his query or a keyword for the subject he/she is interested in **(the user query component)**. Second, the user needs to state his/her preferences over a set of QoS properties of web services **(the user preferences component)**; the system will display a list of the extracted QoS properties. The user provides his/her preferences to as many QoS properties as he requires.

Here, there are two scenarios: (1) the user is able to compare each pair of QoS criteria according to the AHP algorithm. The system will provide multiple questionnaires to the user in order to assign an importance score for each comparison; (2) in the second scenario, the user is not able to perform the comparison procedure because of the lack of knowledge and experience or he is not willing to spend time and effort to do so. Then he can rate the QoS properties using a simple weighting scale that measures the direct importance of the properties to the main goal (i.e. selecting the best service). The system will provide a certain scaling scheme and it could be based on a range of 1 to 10**.** Then the system will feed the user input into the **user oriented weighting component** that is used by the system to produce the weights based on the user preferences.

In either option**,** if no preferences provided to one or some of the QoS properties, a weight of zero will be assigned. This indicates that the user has no preference over this particular property. The flexibility in setting the preferences and producing the QoS weights is one of the main advantages of the proposed framework; the user is not forced to provide unnecessary information. Moreover, the user preferences based on the AHP method or based on the simple weighting scheme are saved in the **user profile component**. Every time, the same user visits the system, his/her preferences will be displayed and he is asked whether he wants to use them or update his set of preferences. This is called a long-term interest in which the user may follow one set of preferences. On the other hand, the user can provide preferences every time he/she submits a new query to the system. Those are called a short-term interest. The new preferences are only associated to the new transaction and they are not necessarily overwriting the long-term preferences. The user has an option to replace the latter with the short-term preferences through the User Interface which means that he has decided to overwrite the old preferences.

Third, the user needs to determine his requirements or constraints over the QoS properties through **the user's non-functional requirements component**. For example, he can request that {availability >0.98, scalability >0.95, reputation >8, security= {very high}}. Later in the selection and ranking stage these requirements will play a crucial role in determining a list of the optimal offers and their ranks.

The existence of the **default weighting component** and the **user profile component** gives our system the major advantage of being efficient in handling complicated scenarios. Using **the final weight component**, the weights generated from the default weights and the user preferences are combined to obtain the final weights of the QoS properties. Later, these weights will be used in the service selection and ranking stage.

The second stage in our service selection system is selecting and ranking the web services based on the user preferences (QoS weights) obtained from the first stage and based on the user constraints and requirements on the QoS properties. The **service selection and ranking component** handles this major task. We also consider the view of the user as a central aspect in this stage. The QoS (non-functional) requirements, requested by the user, will be involved in the process as a discriminating factor that determines the best services that satisfy the service requester. Using the QoS weights, the MCDM methods, and the CP algorithm the optimization problem can be solved by generating the possible ranked service lists.

Here, our enhancements play the most important role in the entire process to better select the desired services since the original algorithms do not consider the user requirements in producing the final solution. For the MCDM based methods, we use our proposed penalty function. For the CP algorithm, we use our layer-based strategy. This process is represented by the **penalty-layer component** in our framework.

In order to evaluate and compare the results to obtain a stable and robust solution **the sensitivity analysis component** is used. At this stage, multiple ranked lists generated from the various algorithms are compared and evaluated. A single ranked list can be produced and presented to the user based on the sensitivity analysis. The sensitivity analysis can be an optional component. An experienced user can choose which ranking algorithm he/she wants to use. Or based on previous sensitivity analysis procedures, a certain algorithm might perform better than others. Then, the selection system can choose this algorithm as a default algorithm to use. If the user does not want to use the sensitivity analysis this default algorithm will always be used. This process will make the selection system more efficient because the sensitivity analysis procedure takes extra time to implement.

Also the system gives a detailed explanation and shows examples for the supported weighting schemes and ranking algorithms. The user then has a better understanding on how he can submit preferences and how the different algorithms work.

In the following part of this chapter, we will describe and analyze each component and show how they connect with each other toward achieving the ultimate goal. The sequence diagram shows the entire process flow (Figure 3.2).

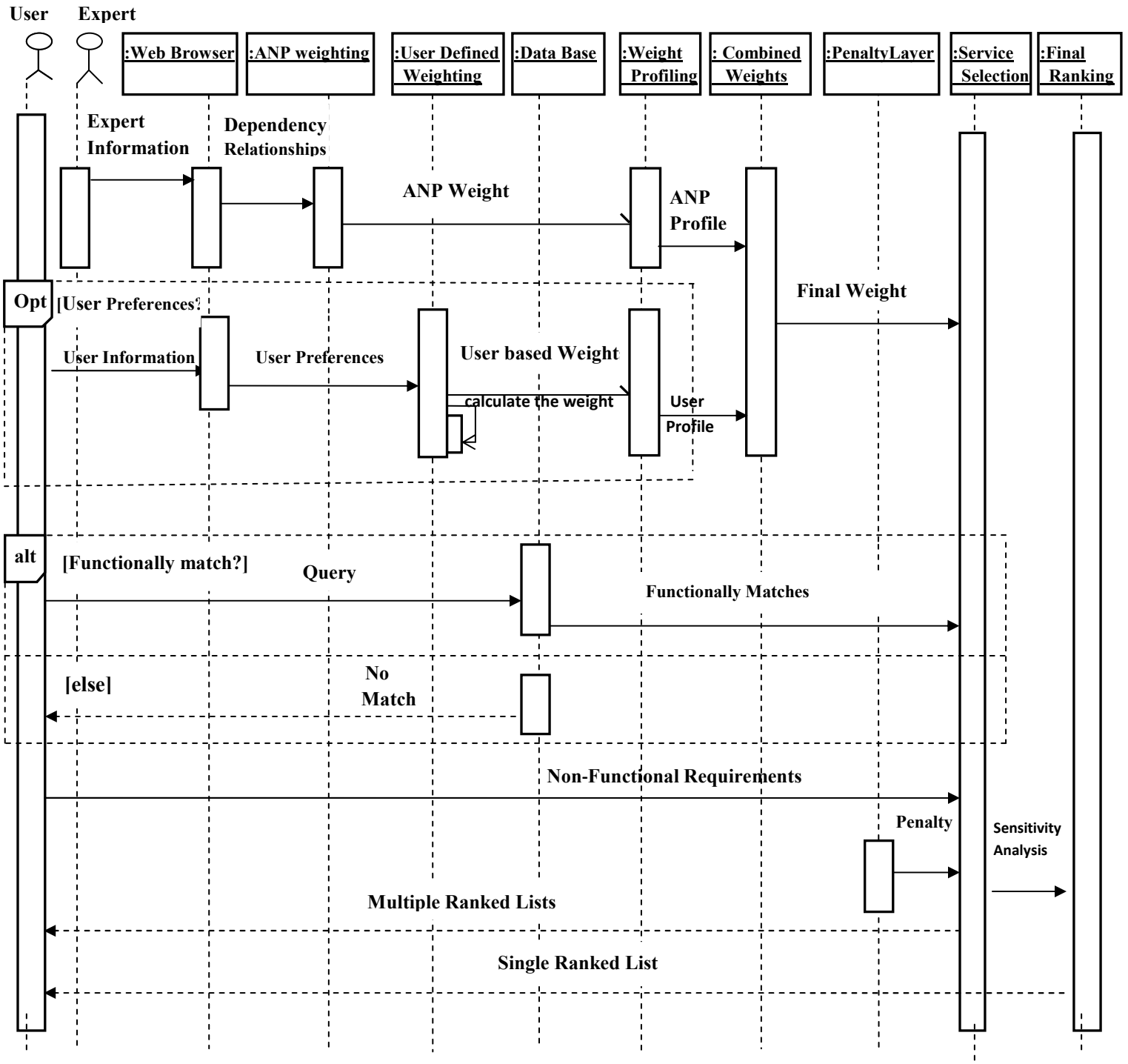

Figure 3.2- A sequence diagram illustrating the flow of the weighting and selection process.

3.3. The QoS (Non-Functional) Properties of Web Services

Since we deal with the QoS-based web service selection problem, we should identify the QoS criteria [31] [32] which are important for service selection. We should also group the QoS criteria in a specific structure according to the relationships among them. This could be a hierarchical or a network structure. We use the AHP method to build the former and we use the ANP method to build the latter [30]. QoS properties of web services can be categorized in many different ways [10] [33]. As we state in Section3.2, the ANP method is used to compute the QoS default weights. Therefore we choose to build a network model of the QoS properties that depicts the interdependency and feedback relationships among them. We propose to divide them into three clusters – High Level QoS, Low Level QoS, and User &Management QoS.

Based on our preliminary investigation, we are able to identify some important dependency relationships between the QoS properties of web services supported by our service selection system. In this specific research we carefully examine the QoS definitions, requirements and common relationships between properties [31] [32] [34] [35]. For example, the scalability influences both reliability and reputation. However, the nature of the relationship between scalability and reliability is different from the one between scalability and reputation. The former is clearly defined by their technical properties with no ambiguity. Their dependency relationships are based on their atomic definitions and commonly agreed on dependencies. Thus, we call it an objective dependency. The latter is arguable among various users. Their dependency relationship is ambiguous and disputable among the service users. Thus we call it a subjective dependency. Since all the dependencies between reputation and other properties are subjective, reputation is considered as a subjective criterion. In our list of properties, service cost and reputation properties are considered as subjective and all of the rest are objective.

Taking into account the above QoS dimensions and consideration, the three clusters in our network model will contain the following QoS properties:

1. The High-Level QoS cluster contains QoS properties that are defined in a higher abstraction level and are determined or measured by lower-level properties. Some of these properties are defined and explained below:

- Availability (**AV**): measures the probability a service is operating normally and can be accessed by users successfully.
- Reliability (**RL**): measures the ability of a service to operate and function as expected under a specific state of conditions over time.
- Composability (**CM**): measures the capability of a service to be well composed with other services to form a more complex application.
- Scalability (**SC**): is the ability of a service to scale up to handle a growing amount of workload.

2. The Low-Level QoS cluster contains the QoS properties that have specific meanings or purposes and have specific metrics for their measurements. Compared to the high level properties, they are usually more primitive. A few representative properties are listed below:

- Response Time (**RT**): is the time from the end of the request to the beginning of the response.
- Throughput (**TP**): measures the volume of data which could be handled at a given period of time.

3. Most of the QoS properties which fall into the first two clusters are performance related. The non-performance-related properties are categorized into the third cluster – the User and Management QoS cluster. Many properties within this category represent users' major

concerns when selecting services or they can be measured by user supplied data, or are related to the management and administrative issues. Some of these properties are listed below:

- Cost (**CO**): refers to how much a user needs to pay in order to use a service.
- Reputation (**RP**): refers to the trustworthiness of a service, which could usually be measured by consumers' ratings and feedbacks based on their experiences of using the service.
- Security (**SU**): measures in what degree security mechanisms (e.g. authentication, authorization, data encryption, non-repudiation, etc.) are supported by a service and how strong these mechanisms are.

Our selection framework is flexible if we need to include more properties or if we have another way of formulating clusters. However, for simplicity, we only consider the QoS properties listed above.

3.4. The Weighting Methods

3.4.1. Overview of the ANP Method

ANP is a multicriteria theory where the relative priorities can be derived from a series of individual judgments. The judgments represent the influence factors among the problem's elements - criteria and alternatives. ANP is a generalized form of its preceding AHP. The former considers all the possible unidirectional and bidirectional relationships among the QoS criteria. The latter structures the problem elements in a hierarchical form; the QoS relationships take only a top-bottom network form.

Due to the fact that the problem's components interact and influence each other, not all real world problems can be solved using a hierarchical model. That is one of the main reasons for the

proposal of the ANP model. The main consideration of the ANP model is the dependency and the interconnection among the different components of a real world problem. To convert the influence relationships to quantitative forms, a process of pair-wise comparisons is conducted. The next section explains this process.

3.4.2. Pair-Wise Comparison and Reciprocal Judgment

Humans use feelings and knowledge to do judgments during the process of the problem solution. The pair-wise comparison [29] [30] is the tool to implement that judgment between a pair of criteria. In both AHP and ANP methods the pair-wise comparison is the backbone of the whole process. In each method, two different and specific questions are asked in order to conduct the comparison. In the case of AHP algorithm, the question that the DM needs to think about is "Which one of the two elements is more important to the immediate goal or its parent goal". This question is associated with each pair of criteria within a group of elements in a specific level in the hierarchy and between two groups in a specific level until the main goal is reached.

In the case of the ANP algorithm, when conducting the judgment the question that DM needs to think about is "What is the influence degree of one element on the other with respect to an element (control criterion/cluster)? Or which one the two criteria has more influence with respect to a control criterion". This ANP-related question is asked whenever we need to compare the influence of two elements in each cluster in the model, and between two clusters in terms of the control cluster until we complete all the possible comparisons. The control criterion and the control cluster represent, to the DM, a way to concentrate his/her thinking. This mechanism helps the DM decomposing the complex problem into a series of influences. The latter is converted to numbers that correspond to the intensity of the influence relationship determined by the DM.

The result is the pair-wise comparison matrix. Its main aspect is the reciprocal value. It represents the inverse of the pair-wise comparison value. For example if it is determined that *a* is five times more important than *b* in terms of other property then 5 will be entered in the corresponding cell of the matrix. And a reciprocal value of 1/5 is entered in the corresponding cell that refers to the element with a less degree of important level. Saaty [29] proposed a scale of relative measurement that is used when performing the pair-wise comparisons (i.e. 1-9 scale: 1 is equal importance, 9 is highest level of importance). It is used in both ANP and AHP methods. Table 3.1 shows the scale. From the pair-wise comparison matrix, the priority values of the included criteria can be calculated using the Eigen vector method [30].

Table 3.1- The fundamental scale of numbers indicating the importance level [30].

The Degree of Importance	1	2	3	4	5	6	7	8	9
Meaning	Equal importance	Weak importance	Moderate importance	Moderate plus	Strong importance	Strong plus	Very strong	Very, very strong importance	Extreme importance

We need to be careful when interpreting the judgments made through the pair-wise comparisons in the ANP method. The resulted numbers in the pair-wise comparison matrix show how a criterion is more important than the other with respect to the control criterion in the cluster they belong to. The numbers do not mean the importance of a criterion compared to the other towards achieving the immediate goal (this is the case of AHP).

3.4.3. Outline of the ANP Algorithm

Below, we explain in detailed steps the ANP algorithm [36]. By following these steps we are able to: (1) generate weights to the QoS properties (criteria) then, (2) select and rank the web

services. The overall evaluation of each service offer determines its rank within the final set of the service offers.

Step 1. The DM (a web service expert, in our case) determines the ultimate goal of the problem – the QoS-based web service selection. Next, he/she builds the network model of the problem's components as shown in Figure 3.3.

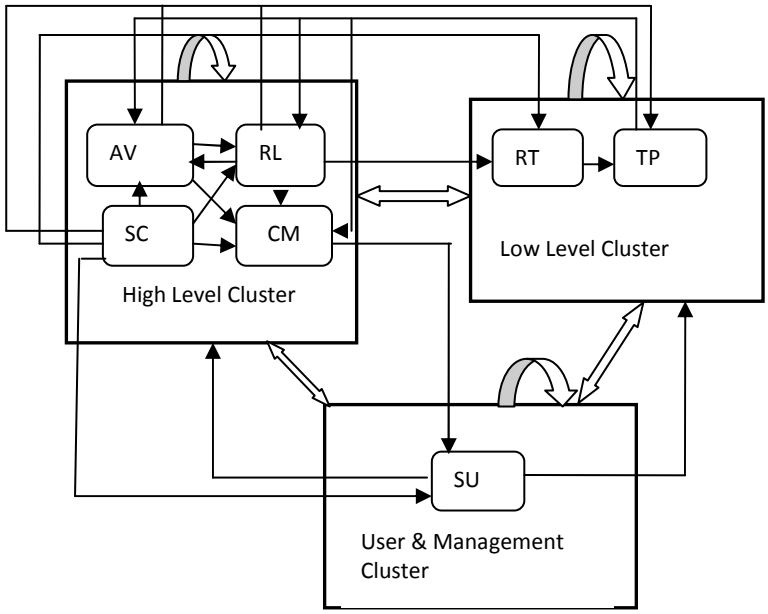

Figure 3.3. The ANP network model for QoS-based service selection.

In this figure, an arrow represents a dependency relationship, which could be between clusters or within clusters, and it usually goes from the affecting element (source) to the affected one (sink). The dependency could be unidirectional or bidirectional (i.e. both the source and the sink depend on each other). An arrow going from an element to itself represents a feedback relationship. After building the network model, the rest of the ANP algorithm steps are followed to generate the ANP-based weights of the QoS properties.

Step 2. A dependency table is constructed in which the criteria (QoS properties listed in Section 3.3) are listed on the topmost row and the leftmost column. The table is partitioned into different parts (blocks) that show how the QoS properties in one cluster depend on properties in the same or another cluster. The input of the table represents the DM judgments. If a property from the top row is affecting a property from the left column then **1** is entered in the corresponding cell otherwise the cell is left **blank**.

Step 3. The unweighted matrix is constructed (it is also called super matrix) in which the criteria are listed on the topmost row and the leftmost column. In each partition (block) of the table, there could be multiple **1s** in a column (under a control criterion on the top). This means that this criterion affects all the criteria (with 1s). For these criteria, the DM has to perform a pair-wise comparison to determine which one is more important in terms of the influences with respect to the control criterion; here, Saaty's scale can be used. This process is repeated for all the blocks. The outcome of the pair-wise comparisons is an Eigenvector that represents the priorities or the weights of these criteria in that partition of the table with respect to the control criterion.

Step 4. The cluster weight matrix is constructed in which the clusters are listed on the topmost row and the leftmost column of the table. The pair-wise comparisons are performed on the clusters themselves, and Saaty's scale is used. We need to determine the importance levels of the other clusters in the network with respect to a control cluster (e.g. the one that has more influence). The process of determining the weights of the clusters depends on the feedback in the cluster itself and on the dependency relationships between this cluster and the others. The outcome of the pair-wise comparisons is an Eigenvector that represents the priorities or the weights of the clusters with respect to the control cluster.

Step 5. The weighted matrix is constructed; it is obtained by multiplying each number in the cluster matrix with the corresponding numbers in the unweighted matrix block. The matrix is stochastic because the unweighted numbers are normalized based on each number of the cluster matrix to have their relative priorities. As a result the sum of the blocks under the control criteria will be equal to one.

Step 6. The limiting matrix is obtained by raising the weighted matrix to a highest power until all columns converge so that all columns are identical. This is because the weights balance out as influences (dependencies) cycle among all criteria. Here the influences are acquired via overall transitivity of the criteria in multidirectional distances. Finally, a column taken from the limiting matrix represents the priorities of the criteria (weights). These weights can then be used in the service ranking process.

According to our observation, a criterion must have sufficient degree of transitivity with other criteria in multiple and different directions and distance to obtain reasonable weights. The less transitivity and/or unidirectional distances from the others the lower weight a criterion will obtain. An interesting finding is that some frequently used criteria are considered important in term of its role towards achieving the main objective of a certain problem (e.g. Scalability in Web services). However, when considering the influences related to this criterion and its unidirectional distance with regard to other criteria, its weight is very low because its influence does not pass through a multidirectional path to others.

3.4.4. The AHP Method

The AHP method is a specific form of the ANP method. Both have common characteristics and underlying features. The obvious difference is that the concept of the interdependency relationships among the different criteria is adopted in ANP whereas in AHP the problem is

structured hierarchically. The upper level represents the problem's ultimate goal then the problem is decomposed into different levels that contain the main criteria and subcriteria. In each level (depending on the problem's complexity) there could be different groups. In each group there could be multiple criteria.

In AHP method, the dependency relationships among the criteria are bottom to top and do not spread in any other direction. The pair-wise comparison process is the subjective judgment made by the DMs among the problem's elements. The purpose is to turn the intangibles into actual numbers that represent the user preferences (the weights of the criteria). The AHP method has been widely used in the industry and different scientific fields. It is easy to follow and requires less establishing time compared to the ANP method. Besides, it is a systematic way to convert the user personal judgments into numbers that reflect his/her preferences over the problem's criteria. Below we explain its algorithm to produce the priorities of the QoS criteria [37].

Step 1. The DM specifies the problem's goal (which in our case ranking the best web services), the criteria (QoS properties) and alternatives (web services).

Step 2. The problem is decomposed to its constituent elements. There could be multiple levels and groups in the hierarchy that include various criteria.

Step 3. Starting from the bottom level, for each group, the pair-wise comparisons will be performed to compute the Eigenvector for the weights for QoS properties. These generated weights imply the user preferences.

Step 4. The process will be repeated for the upper levels to find the Eigenvector. The goal at the top of the hierarchy is simply weighted **1**.

Step 5. Within each group in the bottom level, the QoS weights are multiplied by the weight of its parent group to obtain their relative weights. Thus, the weights of all QoS properties are now normalized. Actually we are interested in these QoS properties when proceeding in the selection and ranking phase. In Figure 3.4, we show a hierarchy model of our QoS-based web service selection problem.

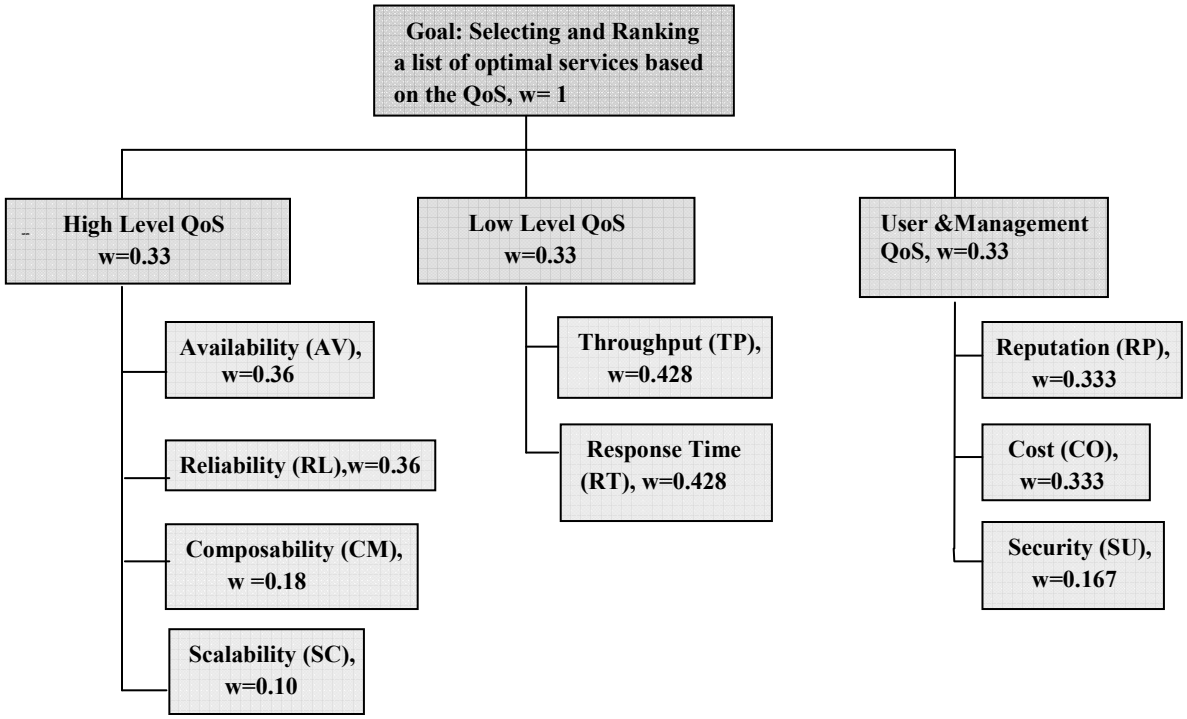

Figure 3.4- The AHP hierarchical model illustrating the problem's goal and criteria.

The hierarchy is composed of three levels; level 3 represents the ultimate goal, level 2 represents the three main criteria groups (High Level QoS, Low Level QoS, User &Management QoS) and level 1 represents the sub criteria (Availability, Reliability, Composability, Scalability, Throughput, Response Time, Reputation, Cost, Security) that belong to each group in level 2. After performing the pair-wise comparison and computing the Eigenvector of the subcriteria, the weights will be obtained as shown in Figure 3.4.

3.4.5. Measuring the Consistency Ratio

One of the important features that are available in both ANP and AHP methods is measuring the consistency ratio [29]. As mentioned before, the user judgment in terms of the comparison between pairs of criteria naturally is subjective. We need to make sure the user input in the pair-wise comparison matrix is consistent. This process has to be performed for each single pair-wise comparison matrix. It eventually results in more accurate weights assigned to the criteria. As a consequence, the final result (in our case the set of optimal service offers based on the QoS properties) will be more accurate.

According to Saaty [29], the inconsistency level in the overall comparisons must not exceed 10% to be considered acceptable. Otherwise, the DM or the user has to revisit the comparisons made earlier and rethink about the judgments. The consistency ratio (CR) is calculated according to these two formulas:

$$CR = \frac{CI}{RI} \qquad (3\text{-}1)$$

$$CI = \frac{\lambda_{max} - n}{n-1} \qquad (3\text{-}2)$$

where *CI* is the consistency index, *RI* is from the random consistency index table proposed by Saaty, λ_{max} is the principle Eigen value obtained from the pair-wise comparison matrix, and *n* is the size of the matrix. Table 3.2 shows the random consistency index.

Table 3.2- The random consistency index [29].

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.52	0.89	1.11	1.25	1.35	1.40	1.45	1.49

3.5. Calculating the Weights of the Non-Functional Properties

3.5.1 Calculating the Default Weights

The weights of the criteria represent their importance levels to individual users when making the trade-off decisions. There are two types of weights; the system default weights, as well as the user defined weights based on their personal preferences. In order to account for the interdependency between various criteria, we choose the ANP model [38] as our weighting scheme to generate the default weight. The reason for this is that ANP method is well-defined and proven method that produces the criteria weights based on the dependency relationships among these criteria. It is more suitable than an equal weighting or arbitrary weighting methods. There are two major shortcomings of the ANP model: its time consuming procedure and its high demand on the user expertise.

In our selection system, we address the first issue by pre-calculating the ANP-based weights before the real-time interaction with the user starts. Therefore, it will not affect user's searching and selection experiences. The second issue is addressed by only involving the expert user in this process. To make this default weight more indisputable and not carrying too much of the expert user's subjective opinions, we only apply the ANP algorithm to non-functional criteria with dependencies which are objective, commonly agreed, clearly defined, and not changing for individual users. The performance related criteria usually have these characteristics and an expert on network and system performances would have the knowledge of defining their dependencies. For other criteria we assign a default weight to them.

3.5.2. Calculating the User Defined Weights

To increase the flexibility of the system, and make the selection results more customized, we also allow users to specify their own preferences over the QoS properties. Also the system

gives the detailed explanation and shows examples for each weighting method so that users would have a clear understanding of them. If the user has no sufficient knowledge or he is not willing to spend too much time and effort in presenting his preferences to our system, then he could use our simple weighting scheme. Using this weighting scheme, the user can rate the QoS criteria using a range of priorities of [1-10] in which 1 refers to the least preferred criterion and 10 refers to the most preferred criterion.

However, if the user is willing to spend a little more time and he has a better knowledge in the area of web service and searching over the network, then he could use the AHP method. In this procedure a questionnaire will be presented to the user to perform the required pair-wise comparisons based on his personal judgment in order to obtain a vector of the QoS property weights. The detailed steps of the AHP algorithm are stated in section 3.4.4. The rational for considering these two options of user-defined weighting methods is that we try to make our proposed framework more flexible to accommodate different degrees of the user experience and knowledge.

3.5.3. Combining Default Weights with User Defined Weights

At this point, two different weights are generated for the QoS criteria; the default weights and the user defined weights. The next step is to normalize the two weight vectors so that the sum of each weight vector is 1. We use these two formulas:

$$w_{iu} = \frac{p_{iu}}{\sum_{j=1}^{M} p_j} \qquad (3\text{-}3)$$

$$w_{id} = \frac{p_{id}}{\sum_{j=1}^{M} p_j} \qquad (3\text{-}4)$$

where w_{iu}, w_{id} represent the user defined weight and the default weight of the i-th criterion respectively, p_{iu} and p_{id} represent the user defined preference and the default value of the i-th criterion respectively, and M refers to the total number of criteria the system supports.

We propose that the final weight is the combination of the normalized default and user defined weights according to this formula [39]:

$$w_i = \alpha \cdot w_{id} + \beta \cdot w_{iu} \qquad (3\text{-}5)$$

where w_i represents the final weight of the i-th criterion, w_{id} represents its default weight and w_{iu} represents the user defined weight, α and β are the coefficients. During a selection process, if a user chooses no preference on any criteria, the value of β is set to zero; if a user only wants his own preferences as the weights without considering the system generated weights, the value of α is set to zero; or otherwise, the final weights are the combined ones. In our later case study, the values of α and β are set to 0.5, which means they are equally important.

3.6. The Selection Methods

3.6.1. The ANP/AHP Method

After calculating the QoS criteria weights, to proceed with the ranking process, the performance values of the service alternatives are first normalized. For this, the idealizing approach is used [36]. This approach has an important advantage to prevent the reversals phenomenon to occur in the final ranking when a new alternative is added. The idealizing approach is done on each QoS criterion by assigning 1 to the service offer with the highest value, and then dividing each offer's value by the value of this highest one. The process is repeated for all the QoS criteria. Next, the weighted sum is used to obtain the score of each service offer. It is performed by multiplying the weights vector with the normalized QoS values. Then the results are

summed up to obtain the final priority score of this service. Suppose we want to calculate the score of service s_i, it is shown below.

$$Score_{ANP/AHP} = \sum_{k=1}^{M} w_k.a_{ik}, \qquad (3\text{-}6)$$

where k represents the k-th non-functional property, M represents the number of QoS properties the system supports, w_k measures its weight, and a_{ik} represents the value of s_i on the k-th property.

This process is performed for all services. The final scores are absolute ranking scores; their sum is not equal to 1. They can be normalized (their sum is equal to 1) to obtain the relative priorities using formula 3.7. Based on these priorities, the services can be ranked from the best to the worst.

$$w_i = \frac{p_i}{\sum_{j=1}^{M} p_j} \qquad (3\text{-}7)$$

where w_i represents the QoS weight of the i-th criterion, p_i represents the user preference of the i-th criterion, and M refers to the total number of criteria the system supports.

3.6.2. The PROMETHEE Method

The PROMETHEE algorithm [40] is established based on the outranking concept and the dominance factor between each pair of alternatives with respect to a criterion. The DM still needs to perform the pair-wise comparison; however in PROMETHEE, he/she measures the deviation between the performance values of two alternatives with respect to all criteria. The best alternative is the one that outranks all or most of the other alternatives. There are three binary relationships defined in PROMETHEE: (1) preference "*P*": an alternative *a* is better than *b* with respect to the criteria *i* and *j* , (2) indifference "*I*": two alternatives *a* and *b* are equal with respect

to the criteria *i* and *j* , (3) incomparable "*R*": if an alternative *a* is better than *b* on the criterion *i* whereas *b* is better than *a* on the criterion *j*, it is impossible to perform the comparison.

One of the advantages of the PROMETHEE model is that it supports six types of pre-defined preference functions, which could cover the within-criterion relationships for most of the criteria (QoS properties, in our case). It could save user's effort of defining utility functions for each property. The other advantage is that PROMETHEE method does not have a fixed weighting scheme, and thus allows for the inclusion of any good weighting method. It is also a model which is simple on its concept and easy for users to understand and apply to solve the real problems [39].

The preference functions are [41]: *Usual criterion, U-shape criterion, V-shape criterion, Level criterion V-shape with indifference criterion and Gaussian criterion*. The main task in defining the preference function is to determine the preference and indifference thresholds (expressed as ***p***, ***q*** respectively). Some of the functions only consider one threshold (***p*** or ***q***), whereas the other functions require defining both of them. On the other hand, some functions can be used for the qualitative criteria and some for quantitative ones. The preference threshold is the minimum value above which the DM can assign the full preference relationship. The indifference threshold is the maximum value that the DM can consider the difference negligible when defining his/her preference with regard to a certain criterion. Defining the preference function and threshold values is one of the important and powerful features available in PROMETHEE, which ensures both accuracy and flexibility. Accuracy-wise, different preference functions provide multiple ways of evaluating alternatives.

In some cases, the dominating degree of one alternative over the other with respect to a specific criterion is very small. The DM has the option to neglect this small deviation and consider both alternatives to be equal in terms of this criterion, which is called the indifference relationship. In other cases when the deviation is large, the DM could define this as a full preference relationship. As for the flexibility, by allowing the DM to choose the appropriate preference and indifference thresholds, the algorithm could be customized to the problem the user is going to solve.

Below we explain the detailed steps of the PROMETHEE algorithm in the selection and ranking process [44].

Step 1. The DM begins the process from the service performance matrix. The service performance values are normalized as it is done in ANP/AHP method. The first task is to determine a preference function (among the six functions as explained before) for each QoS property.

Step 2. After the preference functions are specified, the pair-wise comparisons should be performed to find the deviations between each pair of service alternatives with respect to all the QoS criteria. Then the DM has to inspect these deviations with the help of the chosen preference functions and thresholds to determine how much one alternative is preferred over the other.

Given two services s_i and s_j, the priority of s_i over s_j on criterion k is defined by $P_k(s_i, s_j)$ as shown below,

$$P_k(s_i, s_j) = F_k(d_k(s_i, s_j)) \qquad (3\text{-}8)$$

$$d_k(s_i, s_j) = g_k(s_i) - g_k(s_j) \qquad (3\text{-}9)$$

where $d_k(s_i, s_j)$ measures the deviation between s_i and s_j on criterion k, $g_k(a)$ measures the value of a on k, and $F_k(.)$ is the pre-defined preference function if the criterion is to be maximized or the reversed function if the criterion is to be minimized.

The value of $P_k(s_i, s_j)$ is between 0 to 1 and can be represented as follow:

$$0 \leq P_k(s_i, s_j) \leq 1, \qquad (3\text{-}10)$$

$P_k(s_i, s_j) = 0$ implies that s_i and s_j are indifference. $P_k(s_i, s_j) \sim 0$ implies that s_i has a weak preference over s_j. $P_k(s_i, s_j) \sim 1$ implies that s_i has a considerable degree of preference over s_j. $P_k(s_i, s_j) = 1$ implies that s_i has a full preference over s_j.

Afterwards, these priority values are aggregated on all the QoS criteria. To measure the preference degrees between two alternatives, we use the following formulas,

$$\pi(s_i, s_j) = \sum_{k=1}^{M} w_k \cdot P_k(s_i, s_j) \qquad (3\text{-}11)$$

$$\pi(s_j, s_i) = \sum_{k=1}^{M} w_k \cdot P_k(s_j, s_i) \qquad (3\text{-}12)$$

where w_k is the weight of the k-th QoS criterion, M is the number of QoS criteria, and $\pi(x, y)$ measures how much x is preferred over y on all the QoS criteria.

Step 3. In this step, the positive and negative outranking flows are computed using the formulas below. The former refers to the outranking degree of an alternative over all the other alternatives. The latter refers to the degree of an alternative being outranked by all the other alternatives.

$$\emptyset^+(s) = \frac{1}{n-1} \sum_{x \in AS} \pi(s, x) \qquad (3\text{-}13)$$

$$\emptyset^-(s) = \frac{1}{n-1} \sum_{x \in AS} \pi(x, s) \qquad (3\text{-}14)$$

where n is the number of alternative services, and AS represents this set of services.

Step 4. The net outranking flow is a balance between the positive and negative flows and is calculated using this formula.

$$\emptyset(s) = \emptyset^{+}(s) - \emptyset^{-}(s) \qquad (3\text{-}15)$$

After it is computed for all services, a complete ranking of these services could be generated. Both preference and indifference relationships are kept in this model while the incomparability relationship is removed.

It should be noted that PROMETHEE has two commonly used versions (PROMETHEE I and PROMETHEE II). PROMETHEE I model provides a partial ranking by keeping all 3 relationships – preference, indifference, and incomparability. If two alternatives are incomparable, they are not included in the ranking list (partial ranking). Therefore, it is the DM's responsibility to decide what to do (e.g. he can discard them from the final ranked list). PROMETHEE II model could generate a complete ranking by combining the two flows. Although it might cause the information loss, it could save the human effort on making trade-off decisions. PROMETHEE II is the method we use for the service ranking because a simple and straightforward decision making process is normally preferred for service selection.

3.6.3. The CP Method

CP is a mathematical method used in different areas of scientific research and industry. It is used to solve problem with mathematical constraints. It is often used to model the optimization problem thus it can be implemented as a constraint satisfaction problem (CSP). The standard definition of CSP is shown below [42]:

CSP=(*V, D, C*); where *V* is a finite set of defined variables, *D* is a finite set of Domain for each variable belongs to *V*, *C* is a set of constraints on *V*.

In the QoS-based web service selection problem, the CP models the relationship between the offer (made by a web service provider) and the demand (set by the service user). The two

main concepts that need to be considered in CP methods are consistency and conformance. An offer and a demand are considered to be consistent if their corresponding CSP are satisfiable. And they are considered to be conformant if the solution space of the CSP of the offer is a subset of the solution space of the CSP of the demand and vice versa. For example, if an offer s_i has a response time as [0.2-0.9] in second and the demand is "response time is less or equal to 0.9", it is said that both the offer and the demand are consistent and conformant. This is because the values of each one of them are satisfiable and all values of [0.2-0.9] are in the subset of the "less or equal to 0.9".

The main steps that are required to select and rank the service offers based on their QoS properties following the CP method are as follows.

Step 1. We need to check the service performance against the QoS requirements for consistency and conformance. We can find the services that fully satisfy the QoS requirements and those who partially satisfy them. There are several CP algorithms used for this purpose such as the one described in [43].

Step 2. After we consider only the fully matched services we need to normalize their performance values. We use formulas 3-16 and 3-17 for this purpose. Suppose that the services' performances on the *k*-th non-functional property are $\{ a_{1k}, a_{2k}, \ldots a_{ik} \}$, a'_{ik} is the normalized value on the k-*th* property, a_{max} , a_{min} are the maximum and minimum values of the offers with respect to a QoS criterion.

If the tendencies of the performance values are *High* ,then:

$$a'_{ik} = (a_{ik} - a_{min}) / (a_{max} - a_{min}) \qquad (3\text{-}16)$$

If the tendencies of the performance values are *Low*, then:

$$a'_{ik} = (a_{max} - a_{ik}) / (a_{max} - a_{min}) \quad (3\text{-}17)$$

Here, the "High" tendency means that the higher value of an offer with respect to a QoS property is preferred, "Low" tendency means that the lower value of an offer is preferred.

Step 3. The overall priorities of the services will be computed using the weighted sum method. Finally we sort the services based on their final priorities from the best to the worst to obtain the final ranking order.

There are two types of QoS requirements. The first type specifies the hard constraints on services; the second specifies the soft constraints on services. For the first type services are required to satisfy all QoS requirements. For the second type, the users usually look for ideal values that can be achieved on certain constraints. These values could be compromised if necessary. The latter type of requirements is considered as an optimization problem in the QoS-based web service selection. The user's involvement is very significant in declaring their personal preferences on these criteria and how they want to trade-off between them.

Our main observation is that CP method is good when applied on hard constraints as service offers can be filtered out. In real life, users might be interested in some services that do not satisfy the soft constraints. Unfortunately the CP method treats both types of constraints the same way by excluding the service offers that partially match the QoS constraints. Therefore, this CP mechanism is not very suitable when dealing with the QoS-based web service selection problem. Our enhanced procedure to the CP method is introduced in the next section.

3.7. Improvement to the Selection and Ranking Methods

3.7.1. Defining a Penalty Function

The penalty functions have been widely used in the optimization problems that have constraints. Based on our literature research, all research work in this field agrees on the definition of the penalty function. The obvious aspect of it is to measure the distance between the alternative performance and the desired solution. The simple way to penalize a problem solution is to apply a certain penalty to a violating solution. It is up to the DM to define the penalty function that is appropriate to the problem's constraints [44].

In [45] the authors state that "the penalty technique transforms the constrained problem into an unconstrained problem by penalizing those solutions which are infeasible". The DM has to be careful when defining his/her penalty functions. The degree of the penalty should neither be too big nor too small. Being too large, this means that more information will be lost and some of the feasible solution could be discarded. On the other hand when it is too small, some of the infeasible solution could appear in the list of the optimum or feasible solutions.

In our specific problem, we measure the distance between the service performance and the user constraint (the requested QoS property). We also use the weight of the QoS property. In this context, it is worth to mention that with “distance” we only consider the gap between the offer value and the requested value. If an offer has a better value than the requested one then our proposed penalty function will not take this into consideration. Obviously, the higher the weight of the QoS criterion that a service fails to satisfy and the higher distance value is the higher the penalty would be.

Table 3.3- Using the penalty function on non-satisfying services.

QoS Criteria	Q1	Q2	Q3
QoS Constraint (user requirements)	≥ 5	≥ 0.98	≤ 3
weight	4	3.5	2.5
S1	3	0.97	3
S2	4	0.94	1
S3	5	0.93	2
S4	8	0.98	3

Suppose we have a set of services S = { $s_1, s_2, s_3, s_4 \dots, s_n$}, q_j is the user constraint on a criterion j, w_j is the weight of a QoS property j, d_j is the distance function, M is the number of QoS criteria, then $Pi(s)$ is the overall penalty function applied to service s_n as it is shown in formula 3.18.

$$Pi(s) = \sum_{i=1}^{M} d_j(s_{ij}, q_j) * w_j \qquad (3\text{-}18)$$

To illustrate how the penalty function can be applied in our problem, we show a simple example. Let's have four service offers $\{s_1, s_2, s_3, s_4\}$, and three QoS properties $\{Q_1, Q_2, Q_3\}$. We assign weights to the QoS properties as follow {4, 3.5, 2.5 respectively}, and the user constraints over them are $\{Q_1 \geq 5, Q_2 \geq 0.98, Q_3 \leq 3\}$. Table 3.3 shows all information.

As we can see from the table that s_1 does not satisfy Q_1constraint, s_2 does not satisfy $Q_1$and Q_2 and s_3 does not satisfy Q_2. The penalty function $Pi(s)$ can be applied on these services to penalize them for not satisfying the user requirements on these QoS criteria. Form the table as well, we notice that $s_1, s_2$and s_3 do not satisfy the same user requirement of $Q_2$with their values

of 0.97, 0.94, and 0.93 respectively. The penalties are determined by their distance values from the ideal values specified by the user and the weight of the Q_2 constraint.

3.7.2. Enhancement to the MCDM Based Service Selection Methods

Our improvements to the original MCDM algorithms are applied on the ANP, AHP and PROMETHEE methods respectively. When ranking services, the MCDM model will perform the pair-wise comparison routine to produce the criteria weights and to rank the alternatives.

In the case of ANP and AHP methods, the pair-wise comparison will be performed on the QoS criteria to determine their priorities (weights), then in the selection and ranking stage the weighted sum procedure will be applied to calculate the overall priorities of the services. Lastly, the final scores will be sorted from the best to the worst. The problem with this process is that it is completely neglecting the QoS requirements specified by the user, during the ranking stage.

In the case of PROMETHEE methods, the pair-wise comparisons of alternative services and their outranking flows are processed. If a service outranks all the other services, it is ranked the highest. One problem with this process is that it only considers the deviation of a service from the optimal value, without looking into the actual user requirement. Actually, no matter what the request is, as long as we have a same set of alternative services, they will be ranked the same. It is considered as a severe drawback of the most of the MCDM methods when it is used for QoS-based service selection.

For instance, if we have two services, one has the best overall quality value, but it does not satisfy a few user specified constraints on some QoS properties, another service has a worse overall value, but it satisfies all the constraints, the original MCDM model definitely ranks the first one better. The question is: will the user also prefer this service? If so, what is the point of

defining any constraint? For QoS-based service selection, it is necessary to consider the satisfaction degree of a service to the actual QoS request in the ranking process, besides optimizing all QoS criteria [39].

According to the above description, we can conclude that the two service selection models we use in our research - ANP/AHP and PROMETHEE have common major drawback. That is their complete neglecting of the user requirements on the QoS criteria. Below we describe our general proposal to address this drawback associated to both approaches.

We propose to include the actual QoS requirements from the user into the ranking process. Two improvements are made to the original model. The first one is to use the request to find a cut-off point so that services which are not good with respect to the request will be ranked last (will be stacked at the bottom of the returned list of services). The second improvement is to generate another ranking order based on how well the services could satisfy the user request through the use of a predefined penalty function [39].

Suppose we have a list of service alternatives $\{s_1, s_2, \ldots s_n\}$ and the user request is r, we first add r into the service set, and the new set becomes $\{ s_1, s_2, \ldots s_n, r\}$. Then we calculate the complete ranking order on this new set, and for all services ranked below r, we are going to put them to the bottom of the list because they are not preferred over the request considering their overall QoS optimization (ANP/AHP case) or net outranking flows (PROMETHEE case).

In our second enhancement, we introduce a new ranking score that is based on a penalty score calculated using formula 3.18. The latter is used to measures how much a service s is outranked by the request r. The rationale of considering this measurement is to add a penalty to a service if it could not satisfy part of the QoS requirement from the user. The smaller the value,

the less the penalty will be. With this calculation, we could have another ranking order on the alternative services to show how well they can satisfy the request.

With the two ranked lists of services, we could either present both lists to users and then leave for them to decide which ranked list to select, or use a simple approach such as calculating the weighted sum of two ranking scores to get a final score. Based on the latter a single list is produced and presented to users. The formulas for obtaining a single ranked list are listed below.

$$finalScore(s) = \alpha \cdot Score_{ANP/AHP}(s) + \beta . P(s) \quad \text{(3-19-a)}$$

$$finalScore(s) = \alpha \cdot \emptyset(s) + \beta . P(s) \quad \text{(3-19-b)}$$

Formula 3-19-a is used for the ANP/AHP approach and formula 3-19-b is used for the PROMETHEE approach.

If we want the ranking to emphasize more on how well a service can optimize the QoS criteria, we could choose a bigger α value, and if we want to emphasize more on how well a service can satisfy the user request, we could choose a bigger β value.

3.7.3. Enhancement to the CP Algorithm

We propose a significant and novel idea to improve the way the CP algorithm handle the soft constraints (partially matched offers) in order to suit the problem domain and objective. Our proposal is a layer-based filtering process. The enhancement steps are as follows.

Step 1. The service selection system compares the performance values of each service offer with the QoS constraints determined by the user. The idea is to categorize the services based on their satisfactory degrees to the user request on the QoS constraints.

There are two approaches that we can proceed with in terms of categorizing the services based on how the DM thinks about his/problem or based on the problem itself. In the first approach we organize the services in three layers based on their satisfactory degrees without considering the quantitative aspect of the satisfied constraints. In the second approach we consider the quantity of the satisfied constraints when placing the services in the layers; hence there could be multiple layers in this case.

Step 2.a. If the DM chooses the first approach, the services that fully satisfy the QoS constraints are placed in the upper layer (e.g. Layer 1). The services that partially satisfy the constraint; for instance they satisfy *1/2/.../M-1* constraints where *M* is the total number of the QoS constraints, are placed in the middle layer (e.g. Layer 2). Then the services that do not satisfy any of the constraints are placed in the bottom layer (e.g. Layer 3).

Step 2.b. If the DM chooses the second approach, the services are placed in layers based on the numbers of QoS constraints they satisfy. For instance, in (Layer 1) there would be the services that satisfy *M* constraints, Layer 2 will contain the services that satisfy *M-1* constraints and the last layer will have those which satisfy 0 constraints (they satisfy none of the constraints).

Step 3. After deciding upon which approach to consider, we follow the original steps to normalize the service performance values in each layer separately. Next we compute the overall priorities of the services per layer. Then calculate the final priorities in order to obtain the final ranking.

The outcome of this process is multiple ranked lists of services from each layer. Suppose that the alternative services are S = { $s_1, s_2, s_3, \ldots s_i$}, and suppose that the solution has three layers L = {Layer 1, Layer 2, Layer3}. Then their corresponding ranked lists are: S_i= {$s_1 \ldots s_i$}, S_j = {$s_1 \ldots$

$s_j\}$, $S_k = \{s_{1...}\ s_k\}$. In the last step we only need to concatenate the three lists to obtain the complete ranked list of the services, S = $\{s_{1..}, s_i, s_{1...}\ s_j, s_{1..}, s_k\}$.

The rationale for adopting this technique is to make sure that: (1) the services that partially satisfy the QoS constraints will not be excluded from the selection and ranking process hence they are still valid services, (2) the services that do not fully satisfy the user defined constraints are ranked after those which fully satisfy the constraints. In other word, using the layer-based approach we make sure that the non-satisfying services do not compete with fully satisfying ones towards the top of the set of the optimal services. After applying the layer-based technique the exact position of each service is determined by their overall performance (using the weighted sum). This mechanism is simple and includes no mathematical overhead. Moreover, it reflects our view in solving the problem by including the user requirements on the QoS properties in the ranking process.

3.8. The Time Complexity Analysis

In order to compute the time complexity, we use the big O notation for this purpose. It is the calculation of the worst case in terms of the time the algorithm takes to accomplish its function. We compute the time complexity for the original methods and for our proposed enhanced procedures.

3.8.1. Time Complexity Analysis for the Original Algorithms

3.8.1.1. The ANP Method

We explain the time complexity analysis of the ANP algorithm:

- The first computation step is to find the unweighted matrix based on the dependency matrix. The Eigenvector is used to find the relative weights with respect to a single cluster. The

process is repeated for the whole matrix. The time complexity of finding the Eigenvector is O (*n^3*) [9].

- Finding the cluster matrix: this includes computing the Eigenvector where the dependency relationship exists among the clusters themselves. Computing the Eigenvector takes O (*n^3*) time.
- Finding the weighted matrix: this includes several multiplication operations between each number in the cluster matrix and a corresponding block in the unweighted matrix. The latter can be formed as a vector. Since one iteration loop is used for each multiplication operation, the time complexity is O (*n*) for each single operation. The time complexity for the whole matrix is O (*n^2*).
- Finding the limiting matrix: this includes raising the weighted matrix to large powers until all columns become identical. The best that the multiplication of two dimension matrices takes, is O (*n^2.3*) time. Since there is a chain of matrix multiplications, an outer iteration will be used hence the time complexity for this process is O (*n^3*).
- Normalizing the values of the QoS properties: the matrix contains the services' performances. The matrix dimension is (*n, M*), where *n* is the number of the web services, *M* is the number of the QoS properties. A column represents the offers under each QoS property. The time complexity for normalizing one vector that represents the performance values with respect to a QoS property is O (*2n*) which is equal to O (*n*). The reason is that two non-nested loops are required to perform this operation. The time complexity of this process for the whole matrix is O (*n * M*).
- Synthesizing the weights and the performance values to obtain the overall service score: this encompasses a multiplication operation of two matrices *(n, M), (M, 1)*. The first matrix

represents the normalized performance values and the second matrix represents the weights vector. The time complexity here is O (*n* * *M*) since the second inner iteration loop will be a constant as the second matrix is one dimension.

- Sorting the services' scores: the time complexity for sorting the overall services' scores is O (*n log(n)*).

Based on the above analyses the worst time the ANP algorithm takes is O (*n^3*).

3.8.1.2. The AHP Method

In AHP, we can divide the process into three main stages:

- Calculating the relative weights in each group within each level in the hierarchy: this is done by computing the Eigenvector of the QoS criteria of the group. The time complexity of finding the Eigenvector is O (*n^3*) [9].
- Normalizing the weight vectors: in each level of the hierarchy, the absolute weights within each group in a level have to be normalized to obtain the relative weights with respect to the parent group in the next immediate level. In this process a single parent group weight is multiplied by the weights vector that includes multiple QoS properties. That is multiplying $w_{k,i-1} * w_{(1,2,3\,..j)}$; where *w* is weight value, *k* is the group number, *i-1* is the level number and $(1,2,3\,..j)$ represents the number of the values in the weights vector. The time complexity to normalize a single group of QoS properties is O (*n*). To normalize all groups in a level with respect to its parent level, two nested loops are required and the time complexity is O (*n^2*). The latter is the worst case; if the problem domain is defined to be a single group of QoS properties then the time complexity is O (*n*).

- The subsequent steps are similar to ANP algorithm (i.e. normalizing the performance values, synthesizing the weighs and sorting the services' scores)

Based on the above analysis the worst time the AHP algorithm takes is O (*n^3*).

3.8.1.3. The PROMETHEE Method

- Assigning a preference function to each criterion: this is the first step in the PROMETHEE algorithm. With respect to each QoS property, the performance values of the services have to be compared to determine the two threshold values of preference and indifference (*p, q*). To perform the comparisons between each pair of performance values, two nested loops are required. The time complexity is O (*n^2*).
- Assigning weights to the QoS properties: this requires O (*n*) since an iteration loop is used to visit each property in the QoS properties vector. The weights are pre-computed based on the DM preferred weighting algorithm.
- Computing the preferences formula $\pi(a, b)$ requires O (*n^3*) because three nested iterations are used for comparing each pair of service performance vectors with respect to all QoS properties and finding the deviation values. The outer iteration visits each service vector to perform the next comparison.
- Computing the positive and negative flows ($\emptyset^+$, $\emptyset^-$)**:** this step can be performed inside the previous procedure by first checking the comparison result for each pair of services (whether it is positive or negative) and then computing the sum of the total positive flows and the sum of the total negative flows. Therefore no extra time is computed in this step.
- Computing the net flow by subtracting the $\emptyset^-$ from $\emptyset^+$. This requires O (*n*) for the services vector.

- Sorting the services' scores: the time complexity for sorting the overall services' scores is O (*n log(n)*).

Based on the above analysis, the worst time the PROMETHEE algorithm can take is O (*n^3*).

3.8.1.4. The CP Algorithm

- Normalizing the services performance values: this requires finding the maximum and the minimum values in each vector with respect to a QoS property. Then the normalization is performed using formulas 3.16 and 3.17. This requires a time of O (*n*) for one vector and a time of O (*n^2*) for the entire matrix.
- The rest of the process (synthesizing the weighs and sorting the services' scores) is similar to the other aforementioned algorithms.

Based on the above analysis the worst time the CP algorithm takes is O (n^2).

3.8.2. Time Complexity Analysis for the Proposed Enhancements

3.8.2.1. The MCDM Based Algorithms

Our improvement to the original MCDM algorithms is efficient and easy to implement. This is illustrated in this analysis through new required operations.

- Adding the user request of the QoS properties will increase the performance matrix by one additional row; this has no effect on the time complexity.
- To implement formula 3.18 two iteration loops are needed. The inner loop is to iterate through the service performance vector that has unsatisfying values and compare these values with the user request vector and multiply the results with the corresponding QoS weight then sum them up. The outer loop is used to iterate through the entire matrix. The time complexity is O (*n^2*).

Based on the above description, the time complexity of our enhanced procedure is below the time the original algorithm takes. As a result, the time of the complete procedure of the MCDM method will not be increased.

3.8.2.2. The CP Algorithm

The entire enhancement procedure will not affect the original time complexity. To organize the services by layers based on their satisfactory degrees to the user request, two iteration loops are required. The inner loop is to check each value in the service performance vector if it does not satisfy the user request for a specific QoS property. It depends on the approach; the checking procedure can result in three layers or multiple ones. The outer loop is required to go over each performance vector and repeat the same above procedure. The time complexity is O (n^2). In the next step, we calculate the overall priority for each service within each layer by multiplying the QoS weight with the service performance vector. The time complexity is O ($n * M$). Then we sort the services in each layer. The time complexity for sorting the services is O ($n \log(n)$). Thus the in the worst case the time complexity is O (n^2). Based on the above description, the time complexity of our enhanced procedure is below the time the original algorithm takes. As a result, the time of the complete procedure of the CP method will not be increased.

3.9. The Sensitivity Analysis

Choosing an appropriate MCDM method to solve a multi-criteria problem becomes a challenging task. For decades, several important methods have been proposed and introduced to the literature. In general, MCDM methods play major roles in solving and helping DMs to make appropriate decisions related to a wide range of real-world problems. Not all methods handle multi-criteria problems in the same way; however, they all share the same goal that is obtaining an optimal solution among a set of alternatives. Choosing the best MCDM to solve a problem is

always associated with its accuracy in representing the DM's preferences and optimizing the criteria. The preferences adopted by a DM are subjective to his view. Therefore, it has been difficult to define a method to measure the accuracy of a MCDM method. Using a second method to compare and measure the accuracy of the first one could result in a frustrating conclusion [46] [47]. One way to systematically compare and evaluate particular MCDM methods is to conduct a sensitivity analysis test procedure in which we can identify what method can provide a more stable and robust result.

In our research, we provide a procedure to perform the sensitivity analysis or the "what if" assessment. We measure how a ranking method is stable with respect to the changes that might happen to the QoS criteria weights. There are also, other different approaches to perform the sensitivity analysis. One approach is by considering the changes in service offers values. Another approach is by increasing/decreasing the number of the criteria and/or the alternatives.

In below procedure we increase the QoS weight by consistent and small fraction to know how the smallest change in the weight will affect the overall ranking. It could be considered a systematic and proven way to measure the accuracy and the robustness of each algorithm [48]. The procedure below illustrates the sensitivity analysis in which the deviation degree of each selection algorithm is computed.

Step 1. The input to the process is a services' performance matrix with a fixed and pre-calculated weights for the QoS criteria. In our illustrating case study shown in next chapter, we will use the combined weights of ANP-based and simple weights.

Step 2. Calculating the overall priority scores P of the services $S= \{S_1, S_2, S_3, S_4 \ldots S_n\}$ with respect to the QoS properties or criteria $C= \{C_1, C_2, C_3 .. C_M\}$ using the fixed weighting scheme.

For the ranking process, we choose the desired ranking algorithms. The outcome is a ranked list of services based on each of the selected ranking method. We consider these lists as a baseline to use for the sensitivity analysis.

Step 3. For a specific criterion C_M that has a weight *w* (for simplicity we begin with the highest weight), we specify an upper bound *u (u >= w)* and an increment *i* so that $w \leq w + i \leq u$. Here we make sure that numbers of the increments are equal when processing each one of the criteria.

Step 4. We increase *i* for the chosen criterion once at a time. We normalize the weights of the criteria based on the new increment using formula 3.7. We re-calculate the overall scores of the services *P'* and find the services ranking, where *P'* represents the ranking after the increments. We then compute the deviation score between *P'* and *P* by counting the changes in the service ranking orders in *P'* (we give one score for each single change in service position in the ranked list). If no change has occurred, the score will be 0. The process is repeated until *(w + i) ==u*.

Step 5. The step#3 is repeated for each criterion using the same ranking method. The total deviation scores are calculated. The resulted scores of all criteria are summed up to obtain the final deviation score based on the selected ranking method.

Step 6. The steps #3, #4 are repeated for each ranking method. The less the total score the method has the better it is in terms of its sensitivity to the changes of the criteria weights (it is more stable and robust). As a consequence, the method with a less score should be used for ranking the services.

3.10. Summary

In this chapter we define our web service selection framework. This includes describing all components and their functionalities. The latter is needed to achieve our ultimate goal to select

and rank the optimal service offers based on the user's non-functional requirements and his/her preferences.

Our service selection framework is customizable, flexible and it takes into account the efficiency aspect when processing the user request and his/her information. Our proposed framework mainly considers the user non-functional requirements and the user's view in defining the preferences on the QoS criteria. We introduce our enhancements to the original algorithms and we thoroughly define our improvement procedures. We point out the significance that our improvement makes to the entire selection process. The methods that we use include the MCDM based such as ANP, AHP and PROMETHEE methods, and the CP method. We explain the process flow starting from the user request through the criteria weighting then the service selection and ranking towards producing robust and accurate results. We also compute the time complexity of the original methods and our improvement procedures. We design system architecture and a sequence diagram using UML 2.0 for this purpose.

CHAPTER 4

ILLUSTRATION AND EVALUATION

In this chapter, we provide a detailed case study that illustrates the weighting, and the selection and ranking process of our QoS-based web service selection framework. A sensitivity analysis was also performed on the generated results of the enhanced versions of the selection and ranking methods. The final results clearly illustrate the importance of our improvements to the entire process.

4.1. Case Study General Setting

In our illustrating case study and for simplicity, we chose to include nine service alternatives (service offers) and nine QoS criteria (QoS properties). Table 4.1 includes the complete information needed to perform our illustrating example. This information includes: (1) the QoS properties, (2) the service alternatives, (3) the service performance values, (4) the QoS tendency, (5) the units used to measure the QoS metrics, and (6) the user request on each QoS property (non-functional requirements).

The value of the QoS tendency is [High, Low]; “High” means the higher value of an offer with respect to a QoS property is preferred, “Low” means the lower value is preferred. Some of the QoS properties could be unitless and a percentage or scaling number can be used to interpret the values whereas others have units. The reliability is measured by the Mean Time Between Fail (MTBF) which is usually represented by the hours of the service functionality. The security is measured using an ordinal enumeration value taken from the set {none, very poor, poor, medium, average, high, very high, extremely high, and excellent}. For the response time and

throughput, a range value type is used. To convert the interval data into a single value, for the former an average was calculated to obtain the final value whereas for the latter the final value was calculated by subtracting the lower bound from the upper one.

Table 4.1- The user requests and a list of functionally matching services.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Tendency	High	High	High	High	Low	High	Low	High	High
Unit	%	Hour	%	%	Second	Kbps	USD	-	-
Request	>=0.90	>=10000	>=0.85	>=0.85	<=[1-11]	>=[512-982]	<=5	>=7	>=average
S1	97	16000	0.90	0.95	[0.3-6.7]	[512-982]	5	7	Very High
S2	96	11000	0.87	0.92	[1-11]	[512-990]	4	10	High
S3	90	10900	0.66	0.85	[2-14]	[590-1010]	8	7	Extremely High
S4	98	14000	0.94	0.93	[0.1-3.9]	[513-990]	7	9	Average
S5	88	9900	0.80	0.78	[1.5-12.5]	[569-1024]	6	6	medium
S6	89	9800	0.88	0.70	[0.1-5.9]	[544-1024]	2	8	Very High
S7	94	13000	0.89	0.85	[0.5-9.5]	[515-1000]	5	8	High
S8	88	10000	0.96	0.87	[1-11]	[512-984]	4	8	Excellent
S9	89	9800	0.79	0.75	[1.5-13.5]	[566-1024]	6.5	6	medium

In Chapter 3, we define the QoS properties that we used in our case study. We also illustrate the ANP-based network model as a starting point of obtaining the default QoS weights in our proposed framework. The model includes three main clusters (High Level QoS, Low Level QoS, and User &Management QoS). The network model is flexible and extensible to include more properties and to have different structures in terms of organizing the QoS properties in multiple clusters.

Furthermore, the ANP and PROMETHEE algorithms have been implemented using software packages (SuperDecison 2.0.8 and D-Sight 2.0.4 respectively) [49] [50]. Java language

was used to implement the intermediate steps to calculate the QoS weights and generate the ranked lists of the optimal service offers.

4.2. System Assumption and Limitation

Below are the assumptions we have made and the restrictions of our evaluation setup.

- The web services, involved in our selection and ranking system, are functionally matching offers after the matchmaking process on the functional requirements.
- The user request on the QoS criteria needs to be submitted every time a user uses our service selection system.
- The user is required to submit his/her request on all QoS criteria. If the user has no particular request on a specific criterion, he/she can leave this option blank.
- Using the AHP and ANP methods to evaluate the QoS criteria the user is limited to use a range of values from 1 to 9 to measure the importance degree between the criteria.
- Using the PROMETHEE II method in our selection system, the incomparable relationship is not taken into account. We assume that all services will have their ranking in the final ranked list.
- Fuzzy values for the service performance values are not considered in our system.
- Our selection system does not consider the semantic approach in defining the QoS properties.
- The accuracy of the multiple MCDM methods was not measured. However, we performed a sensitivity analysis process to measure the stability of each method by checking how sensitive the selection method is to the changes that have been made to the QoS weight values.

4.3. Calculating the Weights of the QoS Properties

In this section, we illustrate the process of calculating the default weights and the user defined weights of the QoS properties. The first weighting is based on the ANP method and the second weighting is based on a simple weighting method or the AHP method.

4.3.1. Calculating the Default Weights

The ANP algorithm was used to generate the relative weights for the objective QoS criteria in our service selection framework. Then their average weight was calculated and assigned to the subjective ones. We assumed that the judgments and hence the resulted default weights were based on the expert user's experience. His/her main responsibility was to perform the pair-wise comparisons that led to determine the QoS criteria weights. The first step was to build the dependency-based network model of the objective QoS properties and to create the dependency table as shown in Table 4.2.

Table 4.2- The dependency table is partitioned into multiple blocks.

	QoS Criteria	AV	RL	CM	SC	RT	TP	SU
High Level QoS	AV		1		1		1	1
	RL	1			1		1	1
	CM	1	1		1		1	1
	SC							1
Low Level QoS	RT		1		1			1
	TP	1	1		1	1		1
User& management QoS	SU			1	1			

The purpose of this table was to determine the influences of a QoS property or the whole cluster on another one with respect to a control criterion or a control cluster. Starting from this

table, we performed the pair-wise comparison procedure for the QoS criteria and the clusters towards obtaining the final default weights. The table was partitioned into multiple blocks that represented these relationships. The pair-wise comparisons of the QoS criteria were performed within each cluster in terms of their influences with respect to the control criterion (listed in the topmost of the dependency table). In our dependency table, there are 9 blocks that we wanted to process in order to construct the unweighted matrix according to the ANP algorithm. The process included: (1) performing the pair-wise comparisons, (2) finding the Eigenvectors in order to obtain the relative priorities of the QoS properties with respect to their control criteria in each block, (3) and checking our judgement consistency to ensure a highest degree of accuracy.

As illustrated in Table 4.2, there were 3 blocks within each one of the 3 clusters. In the first block of the High Level QoS, three pair-wise comparisons were required. The first comparison was performed between reliability and composability with respect to availability. The second comparison was performed between the availability and composability with respect to reliability. The last comparison was performed among the availability, reliability and composability with respect to scalability. In the second block, one pair-wise comparison was performed among the availability, reliability and composability with respect to throughput. In the third block, one pair-wise comparison was performed for the relative importance of the availability, reliability, composability and scalability with respect to security.

Within the fourth block in the Low Level QoS cluster, two pair-wise comparisons were performed. The first comparison was performed between the response time and throughput with respect to reliability, and the second comparison was between the same criteria with respect to scalability. In the fifth block, no pair-wise comparison was required. In this block, the dependency relationships (influences) with respect to the control criteria were not enough. In the

sixth block, a pair-wise comparison was required between the response time and throughput with respect to security.

Within the seventh, eighth and ninth blocks the pair-wise comparisons were not required. The reason is that, in our specific case study, only one criterion exists in the User& Management cluster (i.e. security).

To illustrate the pair-wise comparison process, the Eigenvector and the consistency ratio calculations, we choose the third block in the High Level QoS cluster. In this particular block, a pair-wise comparison process was performed with respect to the security control criterion. According to DM's judgement that availability, reliability and scalability were moderately more important than composability thus three values of 3 were entered in the corresponding cells of the availability, reliability and scalability in the pair-wise comparison matrix. The three reciprocal values 1/3 were entered in the cells that are correspondent to the composability. Also the DM has decided that availability, reliability and scalability were equally important with respect to security thus three values of 1 and their reciprocal values were entered in the matrix. This is shown in Table 4.3. Next, we wanted to find the normalized principal Eigenvector that represented the priorities of the compared QoS criteria with respect to security.

We summed up each column, and then each value was divided by the total. Then, the average of each row was obtained that represents the priority of the corresponding criterion. The last column contained the priority vector of the QoS properties which was normalized, thus its sum was equal to 1.

Table 4.3- A pair-wise comparison with respect to security within the High Level QoS cluster.

QoS Criteria	AV	RL	CM	SC
AV	1	1	3	1
RL	1	1	3	1
CM	1/3	1/3	1	1/3
SC	1	1	3	1

We illustrate the calculation of the QoS priorities:

The priority of the availability criterion: (3/10 + 3/10 + 3/10 + 3/10) /4 = 0.3.

The priority of the reliability criterion: (3/10 + 3/10 + 3/10 + 3/10) /4 = 0.3.

The priority of the composability criterion: (1/10 + 1/10 + 1/10 + 1/10) /4 = 0.1.

The priority of the scalability criterion: (3/10 + 3/10 + 3/10 + 3/10) /4 = 0.3.

Table 4.4 shows the final results.

Table 4.4- The pair-wise comparisons result with respect to security.

QoS Criteria	AV		RL		CM		SC		QoS Priority (weights)
AV	1	3/10	1	3/10	3	3/10	1	3/10	0.3
RL	1	3/10	1	3/10	3	3/10	1	3/10	0.3
CM	1/3	1/10	1/3	1/10	1	1/10	1/3	1/10	0.1
SC	1	3/10	1	3/10	3	3/10	1	3/10	0.3
Total	10/3	1	10/3	1	10	1	10/3	1	1

Before considering the results of the pair-wise comparison, it was necessary to check if the judgments (i.e. the pair-wise comparisons) were consistent. Usually it is done for each pair-wise comparison matrix in the whole network model whose size is $n > 2$. If it is consistent then the priority vector can be considered for the next step, otherwise, the DM has to rethink about his/her judgements and the pair-wise comparison is repeated. We used the two formulas 3-1 and 3-2 (listed in Chapter 3) for this purpose. As we stated that the consistency ratio should be less than 10% so that our judgment can be considered as consistent.

First, we calculated the consistency index *CI* using formula 3-2. For this we wanted to find the principal Eigen value λ_{max} . It can be calculated by multiplying each value in the priority vector (weight) by the sum of the columns obtained from the pair-wise comparison.

$\lambda_{max} = 10/3 * (0.3) + 10/3 * (0.3) + 10 * (0.1) + 10/3 * (0.3) = 4.$

We knew that the size of the comparison matrix (n) is equal to 4.

Then, $CI = (4– 4) / 3 = 0.$

Next, we calculated consistency ratio. We knew that the random index of a matrix with size 4 is 0.89 (as shown in Chapter 3).

Then, $CR = 0 / 0.89 = 0$, this is less than 10% and our judgment was consistent.

In the same way, we have calculated the relative priorities of the QoS criteria in the rest of the blocks. In the next step of the ANP algorithm, we entered the priority vectors (relative weights), obtained from the series of pair-wise comparisons, in the corresponding cells of the unweighted matrix as shown in Table 4.5.

Table 4.5- The unweighted matrix.

	QoS Criteria	AV	RL	CM	SC	RT	TP	SU
High Level QoS	AV	0.00	0.8	0.00	0.44	0.00	0.333	0.3
	RL	0.25	0.00	0.00	0.44	0.00	0.333	0.3
	CM	0.75	0.2	0.00	0.11	0.00	0.333	0.1
	SC	0.00	0.00	0.00	0.00	0.00	0.00	0.3
Low Level QoS	RT	0.00	0.5	0.00	0.5	0.00	0.00	0.5
	TP	1	0.5	0.00	0.5	1	0.00	0.5
User &Management QoS	SU	0.00	0.00	1	1	0.00	0.00	0.00

The next step was to construct the cluster weight matrix. As we noticed in the network model that the clusters, too, influenced each other. To construct the cluster weight matrix we first looked into the network model and the dependency table. We wanted to figure out the dependency relationships and how strong they are (e.g. by counting how many 1s in the corresponding block). This helped make our judgements more accurate in terms of the influences among the network clusters. The pair-wise comparison and Eigenvector calculation processes were similar to the ones performed for the QoS criteria and included the three clusters of the network model. To illustrate the process we choose the High Level QoS cluster.

We wanted to calculate the clusters' priorities (weights) with respect to the High Level QoS cluster. The expert has decided that High Level QoS cluster was moderately to strongly more important than Low Level QoS cluster. Also, High Level QoS cluster was moderately more

important than User & Management QoS. The Low Level QoS was equally as important as User & Management QoS. Table 4.6 shows the result of the pair-wise comparisons.

Table 4.6- A pair-wise comparison with respect to the High Level QoS cluster.

Clusters	High Level QoS	Low Level QoS	User & Management QoS
High Level QoS	1	4	3
Low Level QoS	1/4	1	1
User&Manage-ment QoS	1/3	1	1

Next, we wanted to calculate the Eigenvector of this matrix to obtain the priorities of the clusters with respect to the High Level QoS cluster. Table 4.7 shows the priority vector of the High Level QoS criteria.

Table 4.7- The pair-wise comparison result with respect to the High Level QoS cluster.

Clusters	High Level QoS		Low Level QoS		User & Management QoS		Cluster's Priority (weights)
High Level QoS	1	12/19	4	2/3	3	3/5	0.633
Low Level QoS	1/4	3/19	1	1/6	1	1/5	0.174
User&Manage-ment QoS	1/3	4/19	1	1/6	1	1/5	0.192
Total	19/12	1	6	1	5	1	1

Before considering the final weights we had to check the judgment consistency. Following the calculation steps mentioned earlier, the value of *CR* is 0.005 and it is less than 10% which means that our judgment was consistent.

After obtaining the priority vectors of all clusters with respect to each other we formed the cluster weight matrix by entering the resulted vectors in the corresponding columns as shown in Table 4.8.

Table 4.8- The cluster weight matrix.

Clusters	High Level QoS	Low Level QoS	User & Management QoS
High Level QoS	0.633	0.833	0.833
Low Level QoS	0.174	0.166	0.166
User & Manage-ment QoS	0.192	0	0

Next, a weighted matrix was constructed by multiplying each value in the cluster matrix by the corresponding block in the unweighted matrix. For example, the value of 0.633 from the cluster matrix would be multiplied by all values from the first block of the High Level QoS in the unweighted matrix. In our illustrating example, the resulted weighted matrix is shown in Table 4.9. In the weighted matrix, we wanted to make sure that each column was stochastic. For this we might have to normalize it to the sum of 1. To perform the normalization we used formula 3-3. For example the columns under the availability, reliability and throughput of the weighted matrix were obtained by normalizing the original values in order to make them stochastic. The rest of the columns were already stochastic (their sum is 1).

Table 4.9- The weighted matrix.

	QoS Criteria	AV	RL	CM	SC	RT	TP	SU
High Level QoS	AV	0.00	0.627	0.00	0.281	0.00	0.333	0.25
	RL	0.588	0.00	0.00	0.070	0.00	0.333	0.083
	CM	0.196	0.156	0.00	0.281	0.00	0.333	0.25
	SC	0.00	0.00	0.00	0.00	0.00	0.00	0.25
Low Level QoS	RT	0.00	0.108	0.00	0.087	0.00	0.00	0.083
	TP	0.216	0.108	0.00	0.087	1	0.00	0.083
User &Management QoS	SU	0.00	0.00	1	1	0.00	0.00	0.00

The next step was to raise the weighted matrix to a high and limited power until all of its columns converged to become identical. The resulted matrix is called a limiting matrix which is shown in Table 4.10. We took one column as the final weights for the criteria as shown in Table 4.11.

Table 4.10- The limiting matrix.

	QoS Criteria	AV	RL	CM	SC	RT	TP	SU
High Level QoS	AV	0.208	0.208	0.208	0.208	0.208	0.208	0.208
	RL	0.151	0.151	0.151	0.151	0.151	0.151	0.151
	CM	0.212	0.212	0.212	0.212	0.212	0.212	0.212
	SC	0.056	0.056	0.056	0.056	0.056	0.056	0.056
Low Level QoS	RT	0.036	0.036	0.036	0.036	0.036	0.036	0.036
	TP	0.109	0.109	0.109	0.109	0.109	0.109	0.109
User &Management QoS	SU	0.225	0.225	0.225	0.225	0.225	0.225	0.225

Table 4.11- The weights of the objective QoS criteria based on ANP method.

QoS Criteria	AV	RL	CM	SC	RT	TP	SU
Weights	0.208	0.151	0.212	0.056	0.036	0.109	0.225

At this point, all objective QoS properties had their default weights. To assign weights to the subjective properties the average weights of the objective ones were calculated:

$$AVO_{wi}=\frac{\sum_{i=1}^{M} w_i}{M} \qquad (4\text{-}1)$$

where AVO_{wi} refers to the weights average of the objective properties, w_i is the weight values of the QoS properties, M is the number of the objective properties.

$\boldsymbol{AVO_{wi}}$ = 0.208 + 0.151 + 0.212 + 0.056 + 0.036 + 0.109 + 0.225 = 0.9972 / 7 = 0.142. Table 4.12 shows the complete default weights for all QoS properties.

Table 4.12- The ANP based weights of all QoS criteria.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Weights	0.208	0.151	0.212	0.056	0.036	0.109	0.142	0.142	0.225

4.3.2. Calculating the User Defined Weights

In our illustrating case study, we chose to use the simple weighting scheme to generate the QoS weights for the user preferences. Let's assume the user has submitted his/her

preferences through the user interface of our service selection framework. The preferences are shown in Table 4.13.

Table 4.13- User preferences on the QoS properties.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Weights	8	8	5	5	8	8	8	6	7

Based on the user preferences the service availability, reliability, response time, throughput and service cost are the most important criteria followed by service security then service reputation and finally the service composability and scalability.

4.3.3. Calculating the Final QoS Weights

At this stage we wanted to normalize the two weighting vectors, the default weights and the user defined weights. Then we combined them to obtain the final weights using formula 3-5. For normalizing the default weight vector we used formula 3-3. Table 4.14 shows the normalized default weights.

w_{id} = 0.208 + 0.151 + 0.212 + 0.056 + 0.036 + 0.109 + 0.142 + 0.142 + 0.225= 1.2812.

For normalizing the user defined weight vector we use formula 3-4. Table 4.15 shows the normalized user defined weights.

w_{iu} = 8 + 8 + 5 + 5 + 8 + 8 + 8 + 6 + 7 = 63.

Table 4.14- The normalized default weights.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Weights	0.162	0.118	0165	0.044	0.028	0.085	0.111	0.111	0.175

Table 4.15- The normalized user defined weights.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Weights	0.127	0.127	0.08	0.08	0.127	0.127	0.127	0.095	0.111

The combined weights are shown in Table 4.16. These weights were later used in the selection and ranking step.

Table 4.16- The final combined weights of the QoS properties.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Weights	0.144	0.122	0.122	0.062	0.077	0.106	0.119	0.103	0.143

4.4. Selecting and Ranking the Best Web Services

4.4.1. Using ANP/AHP Methods

In order to rank the service offers using this approach, the weighted sum method was used. Below we show the calculation of the services' scores using formula 3-6.

Service 1 = (0.144 * 0.898) + (0.122 * 1) + (0.122 * 0.937) + (0.062 * 1) + (0.077 * 0.571) + (0.106 * 0.970) + (0.119 * 0.4) + (0.103 * 0.7) + (0.143 * 0.75) = 0.816.

Following the same procedure, we calculated the scores for the rest of the service offers. Then we sorted the services based on their scores from the best to the worst to have the final ranking as shown in Table 4.17.

Table 4.17- The service offers ranking based on the ANP/AHP method.

Ranking Order	Services	Total Score
1st	Service 6	0.829
2nd	Service 1	0.816
3rd	Service 4	0.812
4th	Service 8	0.799
5th	Service 2	0.778
6th	Service 7	0.763
7th	Service 3	0.693
8th	Service 5	0.635
9th	Service 9	0.628

4.4.2. Using PROMETHEE Method

The first step was to choose an appropriate preference function among the six ones available to the DM. In our case study we chose to use the *V-shape with indifference criterion* preference function (shown in Figure 4.1) for all our criteria. This is because this particular function deals with the quantitative criteria and it considers two thresholds so that the DM can accurately define the preference and indifference boundaries for the value ranges of the criteria. It could also benefit the decision making process in terms of the flexibility needed in the sensitivity analysis phase.

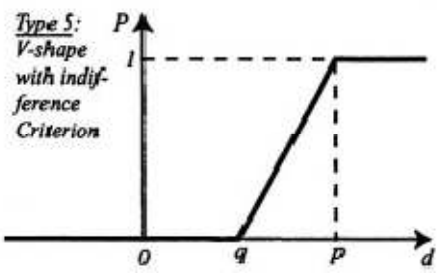

Figure 4.1- The V-shape with indifference criterion preference function [41].

This preference function is defined as follows [44].

$$P(d) = \begin{cases} 0 & d \leq q \\ (d-q)/(p-q) & q < d \leq p \\ 1 & d > p \end{cases}$$

The values of indifference and preference thresholds *q, p* were required for this function and we know that *d* is the deviation value between two performance values. After choosing the preference function we had to determine the two thresholds (*p, q*) for all QoS criteria based on the services' performance values on them. As we described, the DM can determine the values of the thresholds in the way that it is suitable to the problem and the information available to him/her. Table 4.18 shows the values of *p* and *q* that we specified in our illustrating case study.

Table 4.18- The two thresholds (p, q) values for each QoS property.

QoS Criteria	AV		RL		CM		SC		RT		TP		CO		RP		SU	
Threshold Values	q	p	q	p	q	p	q	p	q	p	q	p	q	p	q	p	q	p
	0.008	0.009	0.004	0.005	0.008	0.009	0.008	0.009	0.04	0.05	0.002	0.003	0.2	0.3	0.08	0.09	0.115	0.120

For simplicity we chose the threshold values so that we always have a full preference (i.e. *P (d)* = 1) except for one case that happened in the throughput criterion where the deviation value between service 2 and service 4 is neglected based on formula 3-9. In our case study the value of $d_{TP}(s_2, s_4)$ is equal to 0.002.

To illustrate the ranking process, we consider three services (e.g. service 2, service 3 and service 4). First, we calculate $\pi(s_2, s_3)$, $\pi(s_2, s_4)$ and $\pi(s_3, s_4)$ using formulas 3-11 and 3-12 as shown in Tables 4.19, 4.20 and 4.21.

Table 4.19- The calculated preference degrees between service 2 and service 3.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU	TOTAL
$\pi(s_2, s_3)$	0.0088	0.0007	0.0267	0.0045	0.0064	0.0214	0.0297	0.0309		0.1290
$\pi(s_3, s_2)$									0.0357	0.0357

Table 4.20- The calculated preference degrees between service 2 and service 4.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU	TOTAL
$\pi(s_2, s_4)$						0	0.0255	0.0103	0.0178	0.0536
$\pi(s_4, s_2)$	0.0030	0.0229	0.0089	0.0006	0.0513	0				0.0867

Table 4.21- The calculated preference degrees between service 3 and service 4.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU	TOTAL
$\pi(s_3, s_4)$									0.0536	0.0536
$\pi(s_4, s_3)$	0.0118	0.0236	0.0356	0.0052	0.0577	0.0212	0.0041	0.0206		0.1800

Second we computed the positive and negative flows ($\emptyset^+$ and $\emptyset^-$) of the three services using formulas 3-13 and 3-14.

$\emptyset^+(s2) = 0.1290 + 0.0536 = 0.1826$; $\emptyset^-(s2) = 0.0357 + 0.0867 = 0.1224$

$\emptyset^+(s3) = 0.0357 + 0.0536 = 0.0893$; $\emptyset^-(s3) = 0.1290 + 0.1800 = 0.3090$

$\emptyset^+(s4) = 0.0867 + 0.1800 = 0.2667$; $\emptyset^-(s4) = 0.0536 + 0.0536 = 0.1072$

Then we calculated the outranking net flow of the service offers using formula 3-15. Later the net flow values were used to rank the services.

$\emptyset(s2) = 0.0602$; $\emptyset(s3) = -0.2197$; $\emptyset(s4) = 0.1596$

Finally we ranked the services based on their $\emptyset$ values as shown in Table 4.22.

Table 4.22- The final ranking order of the service offers based on PROMETHEE method.

Ranking Order	Services	$\emptyset^+$	$\emptyset^-$	$\emptyset$
1st	Service 4	0.700	0.262	0.438
2nd	Service 1	0.673	0.248	0.425
3rd	Service 2	0.650	0.289	0.361
4th	Service 7	0.624	0.295	0.329
5th	Service 8	0.580	0.334	0.246
6th	Service 6	0.534	0.396	0.138
7th	Service 3	0.302	0.626	- 0.324
8th	Service 9	0.119	0.813	- 0.694
9th	Service 5	0.095	0.822	- 0.727

4.4.3. Using CP Method

According to our case study, there were only 3 services that fully satisfied the QoS requirements; namely service1, service2 and service7. The rest of the services were excluded from the selection and ranking process based on the CP algorithm. We normalized their values

using formulas 3-16 and 3-17. Table 4.23 shows the normalized values of the three services. Then we calculated the overall score (priority) of each service using the weighted sum as shown in Table 4.24.

Table 4.23- The user request and the matching service offers based on the CP algorithm.

QoS Criteria	AV	RL	CM	SC	RT	TP	CO	RP	SU
Tendency	High	High	High	High	Low	High	Low	High	High
QoS Weights (ANP+Simple)	0.144	0.122	0.122	0.062	0.077	0.106	0.119	0.103	0.143
QoS Requests	>=90	>=10000	>=0.85	>=0.85	<=6.0	>=470	<=5	>=7	>=average
S1	1	1	1	1	1	0	0	0	1
S2	0.666	0	0	0.7	0	0.533	1	1	0
S7	0	0.666	0.666	0	0.4	1	0	0.333	0

Table 4.24- The ranking order of the service offers based on the CP algorithm.

Ranking Order	Services	Total Score
1st	Service 1	0.421
2nd	Service 2	0.379
3rd	Service 7	0.333

4.5. Applying our Improvements on the Original Algorithms

4.5.1. Improving the ANP/AHP and PROMETHEE Approaches

First, we included the QoS request vector as a new service offer in the set of the service alternatives and we call it Service10. Then, we calculated the complete ranking of the services following the steps of the original algorithm. To take into account the user requirements on the

QoS criteria, we marked the newly added service (i.e. QoS request vector) as a cut-off point. The services whose overall optimizations to the QoS criteria above the cut-off point are good with respect to the user request. The services located below cut-off point are considered as poor with respect to the user request. The results are shown in Tables 4.25 and 4.26 with respect to ANP/AHP and PROMETHEE algorithms respectively.

Our second enhancement is using the penalty function. We have calculated the penalty scores on each service that does not completely satisfy the user requirements on the QoS properties. From Table 4.1 we noticed that we had six services (i.e. Service3, Service4, Service5, Service6, Service8, and Service9) that do not satisfy all the user requirements. We used formula 3-18 to compute the total penalty scores applied on these services. The complete calculations of the penalties are illustrated below.

Table 4.25- Using the user request as a cut-off point with the ANP/AHP approach.

Ranking Order	Services	Total Score
1st	Service 6	0.829
2nd	Service 1	0.816
3rd	Service 4	0.812
4th	Service 8	0.799
5th	Service 2	0.778
6th	Service 7	0.763
Request	Service 10	0.692
7th	Service 3	0.693
8th	Service 5	0.635
9th	Service 9	0.628

Table 4.26- Using the user request as a cut-off point with the PROMETHEE method.

Ranking Order	Services	$\emptyset^+$	$\emptyset^-$	$\emptyset$
1st	Service 4	0.700	0.262	0.438
2nd	Service 1	0.673	0.248	0.425
3rd	Service 2	0.650	0.289	0.361
4th	Service 7	0.624	0.295	0.329
5th	Service 8	0.580	0.334	0.246
6th	Service 6	0.534	0.396	0.138
Request	Service 10	0.334	0.527	- 0.193
7th	Service 3	0.302	0.626	- 0.324
8th	Service 9	0.119	0.813	- 0.694
9th	Service 5	0.095	0.822	- 0.727

Service3 failed to satisfy *composability, response time, throughput and cost* criteria, thus:

$P_3(service3)$ = [(0.885 – 0.687) * 0.122] + [(0.333 – 0.25) * 0.077] + [(0.970 – 0.866) * 0.106] + [(0.4 – 0.25) * 0.119] = 0.056.

Service4 failed to satisfy *cost* criterion, thus:

$P_4 = (service4)$ = [(0.4 – 0.285) * 0.119] = 0.0135.

Service6 failed to satisfy *availability, reliability, scalability,* thus:

$P_6(service6)$ = [(0.918 – 0.908) * 0.144] + [(0.625 – 0.612) * 0.122] + [(0.894 – 0.736) * 0.062] = 0.0128.

Service8 failed to satisfy *availability* criterion, thus:

$P_8(service8)$ = [(0.918 – 0.898) * 0.144] = 0.0029.

Service5 failed to satisfy all QoS properties, thus:

$P_5(service5)$ = [(0.918 – 0.898) * 0.144] + [(0.625 – 0.618) * 0.122] + [(0.885 – 0.833) * 0.122] + [(0.894 – 0.821) * 0.062] + [(0.333 – 0.285) * 0.077] + [(0.970 – 0.938) * 0.106] + [(0.4 – 0.333) * 0.119] + [(0.7 – 0.6) * 0.103] + [(0.5 – 0.375) * 0.143] = 0.0578.

Service9 failed to satisfy all QoS properties, thus:

$P_9(service9)$ = [(0.918 – 0.0908) * 0.144] + [(0.625 – 0.0612) * 0.122] + [(0.885 – 0.823) * 0.122] + [(0.894 – 0.789) * 0.062] + [(0.333 – 0.266) * 0.077] + [(0.970 – 0.944) * 0.106] + [(0.4 – 0.307) * 0.119] + [(0.7 – 0.6) * 0.103] + [(0.5 – 0.375) * 0.143] = 0.064.

The result of the above calculations shows that Service3, Service4, Service5, Service6, Service8, and Service9 have different penalty scores that we can use to rank the services. However the penalty scores of Service1, Service2 and Service7 are all equal to 0 because they satisfy all the user requirements on the QoS criteria. For these services we looked into their overall optimization values (the ANP/AHP case) or their net flow values (the PROMETHEE case) to determine their final ranking. This is shown in the second column of Tables 4.27 and 4.28.

The result of our two enhancements was two ranked lists of the service offers for each ANP/AHP and PROMETHEE approaches. We can present them both to the user as shown in Tables 4.27 and 4.28 (the first two columns). Or we produce a single list and present it to the user. In our case study we combined them using formula 3-19.

Using formula 3-19-a and 3-19-b, we chose to assign a value of 0.5 to α and β which indicates that we equally emphasized on the service overall optimization to the QoS criteria and

how well the service satisfies the user request. In both Tables 4.27 and 4.28 we also show the final ranking score of each service (the third column) considering the values that we assigned to α and β. The single ranked lists based on the enhanced ANP/AHP and enhanced PROMETHEE approaches are shown in Tables 4.29 and 4.30 separately.

Table 4.27- The two ranking orders of the service offers based on the enhanced ANP method.

Ranking Order based on the Overall Optimization	Penalty based Ranking	Final Ranking Score	Services	Total Score	Penalty Score
1st	5th	3	Service 6	0.829	0.0128
2nd	1st	1.5	Service 1	0.816	0
3rd	6th	4.5	Service 4	0.812	0.0135
4th	4th	4	Service 8	0.799	0.0029
5th	2nd	3.5	Service 2	0.778	0
6th	3rd	4.5	Service 7	0.763	0
Request			Service 10	0.692	
7th	7th	7	Service 3	0.693	0.056
8th	8th	8	Service 5	0.635	0.058
9th	9th	9	Service 9	0.628	0.064

Table 4.28 – The two ranking orders based on the enhanced PROMETHEE method.

Original PROMETHEE based Ranking	Penalty based Ranking	Final Ranking Score	Services	$\emptyset^+$	$\emptyset^-$	$\emptyset$	Penalty Score
1st	6th	3.5	Service 4	0.700	0.262	0.438	0.0135
2nd	1st	1.5	Service 1	0.673	0.248	0.425	0
3rd	2nd	2.5	Service 2	0.650	0.289	0.361	0
4th	3rd	3.5	Service 7	0.624	0.295	0.329	0
5th	4th	4.5	Service 8	0.580	0.334	0.246	0.0029
6th	5th	5.5	Service 6	0.534	0.396	0.138	0.0128
Request			Service 10	0.333	0.526	- 0.193	
7th	7th	7	Service 3	0.302	0.626	- 0.324	0.056
8th	8th	8	Service 5	0.119	0.813	- 0.694	0.058
9th	9th	9	Service 9	0.095	0.822	- 0.727	0.064

Table 4.29 – The single ranked list based on the enhanced ANP/AHP method.

Final Ranking Order	Services
1st	Service 1
2nd	Service 6
3rd	Service 2
4th	Service 8
5th	Service 4, 7
6th	Service 3
7th	Service 5
8th	Service 9

Table 4.30- The single ranked list of the service offers based on PROMETHEE method.

Final Ranking Order	Services
1st	Service 1
2nd	Service 2
3rd	Service 4, 7
4th	Service 8
5th	Service 6
6th	Service 3
7th	Service 5
8th	Service 9

4.5.2. Discussions of the Enhanced ANP/AHP and PROMETHEE Methods

In Sections 4.4.1 and 4.4.2, we illustrated the service ranking process using the original ANP/AHP and PROMETHEE algorithms and based on the information in Table 4.1. The original algorithms of ANP/AHP and PROMETHEE do not consider the user requests when selecting and

ranking the services. Our results illustrated how some services that do not satisfy the user request could be ranked first whereas those who are good and satisfy the user request could be ranked last. In our example, using the ANP/AHP ranking approach, Service 1, Service 2 and Service 7 that satisfy all the user requirements were only ranked 2nd, 5th and 6th respectively. The services that do not satisfy all user requirements such as Service 6, Service 4 and Service 8 had better ranking with a ranking order of 1st, 3rd and 4th respectively. Using PROMETHEE approach, the Service 4 that does not satisfy all user requirements (e.g. it failed to satisfy the Cost criterion) was ranked first, whereas Service 1, Service 2 and Service 7 were ranked behind it.

These results obviously do not meet the user's expectation. As the users clearly stated their requirements on the QoS criteria, they should obtain the services that satisfy their demands especially if we consider the fees that are applied on services.

Using our enhanced procedure, on one hand, the users can clearly identify the services that satisfy his/her requirements on the QoS based on their overall optimization. According to our improvement, the service selection and ranking system will display a cut-off point in the returned ranked list of services that represents the user requirements on the QoS criteria; we treat the user requirements as an additional service offer. The cut-off point is to inform the user that all services listed above this point are good with respect to his/her submitted requirements whereas all other services that are listed below the cut-off point are not good with this respect. This mechanism clearly shows to the user what services whose overall optimization scores are not good compared to his/her request. This is very helpful: (1) to determine the best offers that the user can consider, (2) to know the services that do not satisfy his/her request then he would think what to do with these services. Without considering the user request in the selection and ranking process, the user could be confused in terms of which ones of the returned services that are good

or poor with respect to his submitted request. In other word, there would be no clear boundary between the two sides so that the user can feel that his request has been respected by the service selection system.

On the other hand, calculating the penalty scores on the non-satisfying services measures how well a service could satisfy the user request on a particular QoS property. The lower the penalty score, the better the service offer will be. The best value is 0, which implies that the service could completely satisfy the request. In both ANP/AHP and PROMETHEE cases, and according to our improvement (calculating the penalty scores), the ranking order of Service 1, Service 2 and Service 7 was significantly improved. They would be ranked at the top of the ranked list leaving the non-satisfying services at the bottom of the list. The penalty scores of these three services were equal to 0 meaning that they completely satisfied the user request. In order to put them in order we simply considered their overall optimization to the QoS criteria. The higher optimization value the better it is so that Service 1 was ranked the first followed by Service 2 then Service 7. The other group of services did not completely satisfy his request. These services were Service 3, Service 4, Service 6 and Service 8. They were ranked according to their penalty scores; the lower score a service had the better its ranking was. Based on the enhanced ANP/AHP and PROMETHEE approaches, Service 8 was ranked the fourth, just after Service 7 followed by Service 6, Service 4. For the services that satisfy none of the user request, their ranking order was at the bottom of the list considering their overall optimal values. We conclude that the ranking orders of the services that could not satisfy all the user requirements on the QoS were noticeably affected.

Our results reflect and illustrate the user's view in terms of the QoS request he/she made when submitting the query. At this point the system could provide the user with two ranked lists

of services. The first list emphasizes more on the overall optimal value. The ranking order from the best to the worst was: Service 6, Service 1, Service 4, Service 8, Service 2 Service 7, Service3 Service 5 and Service 9 using ANP/AHP approach. The ranking order using PROMETHEE algorithm was Service 4, Service 1, Service 2, Service 7, Service 8 Service 6, Service3 Service 5 and Service 9. The second list emphasizes more on how well a service satisfies the user request. In both the enhanced ANP/AHP and PROMETHEE approaches the ranking order of the second list was: Service 1, Service 2, Service 7, Service 8, Service 6, Service 4, Service3 Service 5 and Service 9. We could also provide a single list by combining the two ranked lists.

Through the improvement to the selection and ranking process, we provide the user with a customizable and flexible way to emphasize on any aspect he/she wants- how well the services optimize the QoS criteria and how well they satisfy the user request. This can be performed by adjusting the values of the two coefficients in the combination formula 3-19. From the above description, these new features could largely improve the selection and ranking performance.

4.5.3. Improving the CP Selection Method

First we checked the performance values of the service offers against the user requirements on the QoS criteria for the consistency and conformance to determine the fully, partially and the non satisfying offers. In our improvement procedure, the most important step was to categorize the service offers based on their satisfaction degrees. Table 4.31 shows the service offers organized in three layers. Table 4.32 shows the offers organized in multiple layers based on the number of the satisfied QoS criteria.

Table 4.31- The layer-based service ranking with three categories.

Layers	Services	Description
Layer 1	Service1, Service2, Service7	Satisfy all QoS criteria
Layer 2	Service3, Service4, Service6, Service8	Satisfy some of the QoS criteria
Layer 3	Service5, Service9	Satisfy none of the QoS criteria

Table 4.32- The layer-based service ranking considering the number of the satisfied constraints.

Layers	Services	Description
Layer 1	Service1, Service2, Service7	Satisfy all QoS criteria
Layer 2	Service4, Service8	Satisfy all QoS criteria but one
Layer 3	Service 6	Satisfy all QoS criteria but three
Layer4	Service 3	Satisfy all QoS criteria but four
Layer5	Service5, Service9	Satisfy none of the QoS criteria

In our case study and for simplicity, we chose to continue with the first option (considering the satisfaction degrees of the offers without looking into the number of the satisfied criteria). In the next step, using a weighted sum, we calculated the overall score (priority) of each service within each layer separately. In section 4.3.3 we illustrated the selection process and showed the ranking order of the fully satisfying offers that composed Layer 1, using formulas 3.15 and 3.16. In the same way we ranked the service offers included in Layer 2 and Layer 3. Table 4.33 shows the ranking order of the service offers within each layer.

In our last step we concatenated the three lists of the service offers to form a single ranked list that could be presented to the user. It is important to mention that the final ranking of the

service offers is based on two criteria: (1) their satisfaction degrees to the user requirements between each layer, (2) their overall optimization to the criteria within the layer they belong to. Table 4.34 shows the final ranking based on our improvement to the CP algorithm of the QoS-based web service selection and ranking.

Table 4.33- The ranking order of the service offers within each layer.

Layers	Ranking Order	Services	Total Score
Layer 1	1st	Service 1	0.421
	2nd	Service 2	0.379
	3rd	Service 7	0.333
Layer 2	1st	Service 4	0.746
	2nd	Service 8	0.571
	3rd	Service 6	0.516
	4th	Service 3	0.208
Layer 3	1st	Service 9	0.446
	2nd	Service 5	0.306

Table 4.34- The single ranked list of the offers based on the enhanced CP method.

Final Ranking Orders	Services
1st	Service 1
2nd	Service 2
3rd	Service 7
4th	Service 4
5th	Service 8
6th	Service 6
7th	Service 3
8th	Service 9
9th	Service 5

4.5.4. Discussions of the Enhanced CP Selection Method

We have illustrated the way that the CP algorithm selects and ranks the web services based on their QoS properties. In our case study, only three services that satisfy all the user requirements have been included in the selection and ranking process (i.e. Service 1, Service 2 and Service 7). The rest of the services were excluded because they failed (either partially or fully) to satisfy the user requirements on the QoS properties. As we explained that in real life the users might still be interested in such services and consider them in his/her service searching process. Therefore we introduced our new selection and ranking strategy to overcome this dilemma.

In our illustrating example we illustrated how our enhancement procedure could handle this problem and provide a promising solution. This was through a new strategy to select and rank the service offers with considering the user request. There are obviously two major advantages of our proposed improvement. The first advantage was that, none of the services that failed to partially or fully satisfy the user request would be excluded from the selection process. For this, we categorized the services based on their satisfaction degrees to the user request into multiple layers. For example, Service 1, Service 2 and Service 7 were placed in the upper layer indicating that they were the best in terms of satisfying the user requirements as they fully satisfied the user request. Service 3, Service 4, Service 6 and Service 8 were placed in the middle layer indicating that they were the second best services as they partially satisfied the user requirements. Finally, Service 5 and Service 9 were placed in the bottom layer indicating that they were the worst as they satisfied none of the user requirements on the QoS property.

The second advantage was that we took the user's personal non-functional requirements on the QoS criteria as a central aspect in ranking the service offers and. In the same time, we considered the service's overall optimization to determine its ranking order within the layer it belongs to. For example, in the middle layer the ranking order of the included services was (Service 4, Service 8, Service 6, and Service 3). Service 4 is the best within this layer because its optimal value is the best compared to the other services' values.

We did observe that the partially-satisfying services such as those in the middle layer never compete with the fully-satisfying services in the upper layer to obtain a better ranking order. This is very important with respect to the user's view and personal preferences. The main idea of using the CP algorithm should be to satisfy the user constraints that were set in the beginning of the process. In the mean time, the selection algorithm should provide a clear strategy to include the services that can not satisfy all the user constraints. The complete exclusion of these services will cause a frustrating reaction from the user especially if the number of the partially and none satisfying services is high. For these different reasons, our improvement to the CP selection and ranking algorithm is extremely important and can be considered as a novel idea in our field of research.

4.6. Measuring the Stability of the Enhanced Selection Algorithms

We followed the sensitivity analysis procedure (as described in Chapter 3) to measure the stability of the ANP/AHP approach, PROMETHEE algorithm and the CP algorithm. Since the outcome of an algorithm is associated to the true weighting, we took it as a true evaluation when conducting the sensitivity analysis test. Then we increased each QoS weight by 0.01 for 10 times, and the entire QoS weights vector was normalized. Next we calculated the overall score of

the service offers based on the new normalized weights. Here we considered our enhancements of the selection and ranking process. We have produced 10 ranked lists based on the ten increments of one QoS weight. We repeated the process for every single QoS property which led to 90 ranked lists (in our example). The result was the total number of the changes that happened in the services' positions in their ranking order in each ranked list. The sensitivity analysis test was performed for the three algorithms mentioned above. The more score (the number of changes in the ranking order) the algorithm has the less stable it is. The idea is that we wanted to know the most valid outcome that has the minimum disagreement or deviation from the true one.

In Table 4.35 we provide the total sensitivity score of each algorithm according to our tests. We noticed that the CP algorithm with considering our enhancement over the service selection and ranking process (the layer-based approach) was more stable than the other approaches we used in our tests (i.e. enhanced ANP/AHP and enhanced PROMETHEE algorithms). Consequently our service selection framework could return to the user the ranked list of service offers produced by the enhanced CP algorithm for its stability and robustness in terms of the changes in the QoS weights.

Table 4.35- The sensitivity analysis results of the three selection algorithms.

Criteria	**AV**	**RL**	**CM**	**SC**	**RT**	**TP**	**CO**	**RP**	**SU**	Sensitivity Degree
Enhanced ANP/AHP	0	22	0	19	29	0	10	17	24	123
Enhanced PROMETHEE	6	10	6	10	8	16	30	12	22	120
Enhanced CP	0	7	0	0	16	0	13	0	0	36
Criteria Sensitivity Scores	6	39	6	29	53	16	53	29	46	279

On the other hand, we noticed that response time and cost properties are the most critical criteria with sensitivity scores of 53 for each. The less affecting criteria are availability and composability with scores of 6 for each.

To illustrate the sensitivity analysis procedure, we take the enhanced CP algorithm using our weighting scheme in the service selection framework. We show the impact of the weight increments of the reliability property on the ranking order of the service offers. In this example, 'S' refers to 'Service', '>' means 'more preferable' and '=' means that two services are equally preferred.

First, we consider the outcome based on the true weighting:

S1>S2>S7>S6>S8>S4>S3>S5>S9

Then we increased the weight of the reliability criterion (i.e. $w = 0.122$) by 0.01 for 10 times and generated the new rankings as follows.

@ Weight of 0.132, the ranking order is: S1>S2>S7>S6>S8>S4>S3>S5>S9.

@ Weight of 0.142, the ranking order is: S1>S2>S7>S6>S8>S4>S3>S5>S9.

@ Weight of 0.152, the ranking order is: S1>S2>S7>S6>S8>S4>S3>S5>S9.

@ Weight of 0.162, the ranking order is: S1>S2>S7>S6>S8>S4>S3>S5>S9.

@ Weight of 0.172, the ranking order is: S1>S2>S7>S6>S8>S4>S3>S5>S9.

@ Weight of 0.182, the ranking order is: S1>S2>S7>S6>S8>S4>S3>S5>S9.

@ Weight of 0.192, the ranking order is: S1>S2>S7>S6>**S8=S4**>S3>S5>S9.

@ Weight of 0.202, the ranking order is: S1>S2>S7>S6>**S4>S8**>S3>S5>S9.

@ Weight of 0.212, the ranking order is: S1>S2>S7>S6>**S4>S8**>S3>S5>S9.

@ Weight of 0.222, the ranking order is: S1>S2>S7>S6>**S4>S8**>S3>S5>S9.

As we notice here that at weight of 0.192 both Service 4 and Service 8 were equally preferred because the former has moved one position forward. Thus, one point was added to the sensitivity degree of the used algorithm. In the subsequent weight increments the two services have swapped their ranking orders, thus two points were added. Since this happened in three increments, then 6 pints were added. The total sensitivity degree of reliability was 7. By repeating this process for every single QoS criterion we obtained the total sensitivity degree of the enhanced CP algorithm which is 36.

4.7. Summary

In this chapter, we illustrate how our service selection system would implement the weighting, and the selection and ranking processes using a case study example. First we explain our weighting scheme that includes calculating the default weights using the ANP method. Second we illustrate the steps of calculating the user defined weights for the QoS criteria using a simple and flexible method based on his/her personal preferences.

In the selection and ranking stage, we illustrate the steps of each algorithm to generate the final ranked list of the optimal services. Next we emphasize, in our illustration, on the significance of our enhancements to the selection and ranking process. We explain how the improved selection and ranking algorithms can provide the user with the results that reflect his/her perspective and preference in terms of the requested QoS values.

In our sensitivity analysis process we compare the three selection and ranking approaches we use in our selection framework with the consideration of our enhancements to them. We illustrate how we could generate a single ranked list of services that is considered as a stable result. This is performed by showing the sensitivity degrees of each algorithm in terms of the changes that have been made to the weight of each QoS criterion.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

In our thesis we reviewed the existing research work in the field of QoS-based web service selection. We addressed some of the significant issues and shortcomings that characterise these works. Different MCDM methods along with the CP method have been integrated in our proposed service selection framework. In our research work we proposed a customizable and flexible weighting scheme. We improved the selection and ranking process in a way that reflects the user view of selecting and ranking the best services in order to meet his/her expectation regarding the QoS requirements. We designed our customizable and flexible service selection framework for this purpose.

Our proposed framework has the following features:

- It provides a systematic and flexible weighting scheme to the QoS criteria. To reduce the user's workload an expert was allowed to handle the process of defining the default weights. To make the system more flexible and customizable we give the user the opportunity to define his/her preferences on the QoS criteria in a simple method. Moreover, the system efficiency will not be affected because the predefined weights could be saved as default weights. The user will be able to retrieve his preference information in later usages.
- The user requirements on the QoS values are considered as a principal factor to determine how well a service satisfies the user request. Based on its satisfaction degree, the service will have its rank in the final ranked list.

- The interdependency relationships among the QoS properties were taken into account when generating the default weights of the objective QoS criteria. This characteristic exists in real life scenarios; and it is often not considered in most of the existing research work.
- Our service selection framework provides the user with the most stable and robust result through performing a sensitivity analysis on our enhanced selection algorithms.

5.2. Future Work

The field of QoS-base web service selection is expanding and growing to include different theories and to be applied in different domains. From this point, several paths can be taken into account for future research. We list some of them:

- We have used two main strategies to utilize our decision process; they are penalty function based and layer based. We can investigate other decision strategies that can be exploited towards solving our problem. Reference [51] can be a good resource for this purpose.
- Cloud computing can be employed in QoS-based web service selection. Reference [52] can be a good resource for this purpose. Service selection is a key function to be used in cloud computing, users can search services from cloud, in a way; it could help more users use the services on cloud.
- Further sensitivity analysis research needs to be performed to generate the most suitable result. Our presented procedure was based on a preliminary study that we have conducted. There are different approaches that can be adopted. Reference [53] can be a very good resource for this purpose.
- Although several scholars addressed the problem of computing the accuracy of the MCDM based selection algorithms, we still could conduct a thorough research study and try to come up with a proven procedure to compare the accuracy of multiple algorithms.

- We can further consider a semantic web service selection approach in our service selection framework. Also, fuzzy values can be considered for the services performance on the QoS properties.
- Finally, a GUI-based application can be developed according to our framework prototype and test it on real data.

REFERENCES

[1] M. Papazoglou, *Web Services :Principle and Technology (1st ed.)*, Harlow: Pearson Education Limited, ISBN: 978-0-321-15555-9, 2008.

[2] T. Erl, *Service-Oriented Architecture, Concepts, Technology, and Design*, Prentice Hall, Indiana, ISBN-10: 0-13-185858-0, 2006.

[3] M.P. Singh, M.N. Huhns, *Service-Oriented Computing, Semantics, Processes, Agents*, John Wiley&Sons Ltd.,West Sussex, ISBN: 0-470-09148-7, 2005.

[4] M.P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions", in *Proceedings of the Fourth International Conference on Web Information Systems Engineering,* Rome, Italy, pp.1-10, 2003.

[5] IBM Services Architecture Team, "Web Services Architecture Overview", http://www.ibm.com/developerworks/webservices/library/w-ovr/#author1, Last retrieved at July 2011.

[6] A.F. M. Huang, C. Lan, S.J.H. Yang, "An Optimal QoS-based Web Service Selection Scheme", *Information Science*, Vol. 179, Issue 19, pp. 3309-3322, 2009.

[7] B. Roy, "Paradigms and Challenges", in *Multiple Criteria Decision Analysis—State of the Art Annotated Surveys,* Vol. 78, Figueira J., Greco S., Ehrgott M., Ed. Stanford University: Springer, ISBN: 0-387-23067-X, pp.3-18, 2005.

[8] C.A.C. Coello, G.B. Lamont, D.A.V. Veldhuizen, "Multi Criteria Decision Making", in *Evolutionary Algorithms for Solving Multi-Objective Problems* (2nd ed.), Vol. 8, Springer Science, ISBN: 978-0-387-33254-3, pp.415-445, 2007.

[9] V.X. Tran, H. Tsuji, R. Masuda, "A New QoS Ontology and its QoS-based Ranking Algorithm for Web services", *Simulation Modelling Practice and Theory*, Vol.17, Issue 8, pp. 1378–1398, 2009.

[10] M. Godse, R. Sonar, and S. Mulik, "Web Service Selection on Analytical Network Process Approach", in *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, Yilan, Taiwan, pp. 1103-1108, 2008.

[11] C. Herssens, I.J. Jureta, and S. Faulkner, "Dealing with Quality Tradeoffs during Service Selection", in *Proceedings of the International Conference on Autonomic Computing*, Illinois, USA, pp. 77- 86, 2008.

[12] Y.J. Seo, H.Y. Jeong, and Y.J. Song, "Best Web Service Selection Based on the Decision Making between QoS Criteria of Service", in *Proceedings of the 2nd International Conference on Embedded Software and Systems*, Xian, China, pp. 408-419, 2005.

[13] M. Godse, R. Sonar, and S. Mulik, "The Analytical Hierarchy Process Approach for Prioritizing Features in the Selection of Web Service", in *Proceedings of the Sixth European Conference on Web Services*, Ayia Napa, Cyprus, pp. 41-50, 2008.

[14] C. Marchairs, J. Springael, K. De Brucker, A. Verbeke, "PROMETHEE and AHP: The Design of Operational Synergies in Multicriteria Analysis. Strengthening PROMETHEE with Ideas of AHP", *European Journal of Operational Research*, Vol. 153, Issue 2, pp.307-317, 2004.

[15] R.U. Bilsel, G. Buyukozkan, D. Ruan, "A Fuzzy Preference-Ranking Model for a Quality

Evaluation of Hospital Web Sites". *International Journal of Intelligent Systems,*Vol.21, Issue 11, pp.1181–1197, 2006.

[16] H.R. Yazgan, S. Boran, K. Goztepe, ," An ERP Software Selection Process with Using Artificial Neural Network Based on Analytic Network Process Approach", *Expert Systems with Applications*, Vol.36, Issue 5, pp.9214-9222, 2009.

[17] J.A.W. Mulebeke, L. Zheng, ,"Analytical Network Process for Software Selection in Product Development: A Case Study", *Journal of Engineering and Technology Management*, Vol.23, Issue 4 ,pp.337-352, 2006.

[18] F. Rossi, P.V. Beek, T. Walsh, Handbook of Constraint Programming (Foundations of Artificial Intelligence), Elsevier, The Netherlands, ISBN-10: 0-444-52726-5, 2006.

[19] M. A. Zemni, S. Benbernou, M. Carro,"A Soft Constraint-Based Approach to QoS-Aware Service Selection", in *Proceedings of the Eighth International Conference on Service Oriented Computing*, CA, USA, pp. 596-602, 2010.

[20] A. Ruiz-Cortés, O. Martín-Díaz, A.D. Toro, and M. Toro, "Improving the Automatic Procurement of Web Services Using Constraint Programming", *International Journal on Cooperative Information Systems*, Vol.14, Issue 4, pp. 439-468, 2005.

[21] J.M. Garc´ıa, D. Ruiz, A. Ruiz-Cort´es, "On User Preferences and Utility Functions in Selection: A Semantic Approach", in *Proceedings of the Fifth International Conference on Service-Oriented Computing*, Sydney, Australia, pp. 105–114, 2008.

[22] E.M. Maximilien, and M.P. Singh, "A Framework and Ontology for Dynamic Web Service Selection", *IEEE Internet Computing*, Vol. 8, Issue 5, pp. 84-93, 2004.

[23] C. Hang, M. Singh, "From Quality to Utility: Adaptive Service Selection Framework", in *Proceeding of the Eighth International Conference on Service-Oriented Computing,* CA, USA, pp. 456-470, 2010.

[24] J. Li, D. Ma, J. Han, and X. Long, "Toward Trustworthy Semantic Web Service Discovery and Selection", in *Proceedings of the 6th International Conference on Automatic and Trusted Computing*, Brisbane, Austria, pp. 209-220, 2009.

[25] L. Vu, M. Hauswirth, K. Abrerer, "QoS-based Service Selection and Ranking with Trust and Reputation Management", in *Proceedings of the Cooperative Information System Conference*, Ayia Napa, Cyprus, pp. 466-483, 2005.

[26] K. Kritikos, and D. Plexousakis*,"* Mixed-Integer Programming for QoS-Based Web Service Matchmaking*"*, *IEEE Transaction on Services Computing*, Vol. 2, Issue 2, pp.122-139, 2009.

[27] Q. Yu, and A. Bouguettaya, "Computing Service Skyline from Uncertain QoWS", *The IEEE Transactions on Services Computing*, Vol.3, Issue 1, pp.16-29, 2010.

[28] D. Booth, "*Web Services Architecture*", W3C Working Group Note 11 February 2004, http://www.w3.org/TR/wsa-reqs/, Last retrieved at July 2011.

[29] L. T. Saaty, L. G. Vargas, " Analytical Network Process", in *Decision Making with the Analytical Network Process, Economic, Political, Social and Technological Applications with Benefits, Opportunities, Costs and Risks*, Vol.95, Springer, ISBN-10: 0-387-33859-4, pp. 1-26, 2006.

[30] L.T. Saaty, "Analytic Hierarchy and Analytic Network Processes for the Measurement of Intangible Criteria and for Decision-Making", in *Multiple Criteria Decision Analysis—State of*

the Art Annotated Surveys, Vol. 78, Figueira J., Greco S., Ehrgott M., Ed. Stanford University: Springer, ISBN: 0-387-23067-X, pp. 346-406, 2005.

[31] K. Kritikos, D. Plexousakis, "Requirements for QoS-based Web Service Description and Discovery", *IEEE Transactionnns on Services Computing*, Vol. 2, pp. 320-337, 2009.

[32] S.W. Choi, J.S. Her, and S.D. Kim, "Modeling QoS Attributes and Metrics for Evaluating Services in SOA Considering Consumers' Perspective as the First Class Requirement", in *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, Tsukuba Science City, Japan, pp. 398-405, 2007.

[33**]** Y.T. Liu, A.H.H. Ngu, L.Z. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection", in *Proceedings of the International Conference on World Wide Web*, New York, USA, pp. 66-73, 2004.

[34] S.W. Choi, J.S. Her, S.D. Kim, "QoS Metrics for Evaluating Services from the Perspective of Service Providers", *IEEE International Conference on e-Business Engineering*, Hong Kong, China, pp. 622-625, 2007.

[35] J. Li, D. Ma, J. Han, and X. Long, "Toward Trustworthy Semantic Web Service Discovery and Selection", in *Proceedings of the 6th International Conference on Automatic and Trusted Computing*, Xian, China, pp. 209-220, 2009.

[36] T.L. Saaty, *Theory and Applications of the Analytic Network Process: Decision Making with Benefits, Opportunities, Costs, and Risks*, RWS Publications, Pittsburgh, PA, ISBN: 1-888603-06-2, 2005.

[37] T.L. Saaty, "Decision Making with the Analytical Hierarchy Process", *International Journal of Services Sciences*, Vol.1, No.1, pp. 83-98, 2008.

[38] M.P. Nimira, T.L. Saaty, “An Analytic Network Process model for financial-crisis forecasting”, in *Decision Making with the Analytic Network Process:Economic, Political, Social and Technological Applications with Benefits, Opportunities, Costs, and Risks*, Springer Science+Business media, Boston, ISBN-10: 0-387-33859-4, 2006.

[39] R. Karim, C. Ding, “An Enhanced PROMETHEE Model for QoS-Based Web Service Selection”, in *Proceedings of the Eighth International Conference on Services Computing* (Accepted), Washington DC, USA, pp. 1-6, 2011.

[40] J. Brans, and B. Mareschal, “PROMETHEE Methods”, in *Multiple Criteria Decision Analysis: State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott, Ed., Springer Verlag, Boston, ISBN: 0-387-23067-X , pp. 163-196, 2005.

[41] E. Ocelikova, D. Klimwsova, “Using PROMETHEE Method for the Ranking of Multidimensional Data”, in *Proceedings of the Eighth IEEE International Symposium on Applied Machine Intelligence and Informatics*, Herlany, Slovakia, pp. 93-96 , 2010.

[42] J. M. Garc´ıa, D. Ruiz, A. Ruiz-Cort´es, O. Mart´ın-D´ıaz, M. Resinas,“An Hybrid, QoS-Aware Discovery of Semantic Web Services Using Constraint Programming”, in *Proceedings of the Fifth International Conference on Service Oriented Computing*, CA, USA, pp. 69–80, 2007.

[43] Q. Ma, H. Wang, Y. Li, G. Xie, and F. Liu, “A Semantic QoS-aware Discovery Framework for Web Services”, in *Proceedings of the IEEE International Conference on Web Services*, Beijing, China, pp. 129-136, 2008.

[44] V. Mousseau, L.C. Dias, J. Figueira, “Dealing with Inconsistent Judgments in Multiple Criteria Sorting Models”, *A Quarterly Journal of Operations Research*, Vol. 4, pp. 1-17, 2005.

[45] J. A. Joines, C. R. Houck, “On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA’s”, in *Proceedings of International Conference on Evolutionary Computation*, Florida, USA, pp. 579-584, 1994.

[46] V. Ozemoy, “Choosing the Best Multiple Criteria Decision-Making Method”, *Information Systems and Operational Research* ,Vol. 30, Issue. 2, pp. 159-172, 1992.

[47] D. L. Olson, H. M. Moshkovich, R. Schllenbberger, A. I. Mechitov, “Consistency and Accuracy in Decision Aids: Experiments with Four Multiattribute Systems”, *Decisions Sciences*, Vol. 26, Issue 6, pp. 723-748, 1994

[48] Y. Chang, C. Yeh, “Evaluating Airline Competitiveness Using Multi-Attribute Decision Making”, *The international Journal of Management Science*, Vol. 29, Issue 5, pp. 405-415, 2001.

[49] SuperDecision Software for Decision-Making, http://www.superdecisions.com/, Last retrieved at July 2011.

[50] D-Sight Multi-Criteria Software, http://www.d-sight.com/, Last retrieved at July 2011.

[51] J.W. Payne, J.R. Bettman, “Adaptive Strategy Selection in Decision Making”. *Journal of Experimental Psychology: Learning, Memory, and Cognition* , Vol.14, Issue 3, pp. 534-552, 1988.

[52] S. Wang, Z. Zhang, O. Sun, H. Zou, F. Yang, “ Cloud Model for Service Selection”, *IEEE INFOCOM Workshop on Cloud Computing*, Shanghai, China, pp. 1-6, 2011.

[53] E. Triantaphyllou, “A Sensitivity Analysis Approach for Some Deterministic Multi-Criteria Decision Making Methods”, *Decision Sciences*, Vol.28, Issue 1, pp. 151-194, 1997.