Master Thesis

“QoS of Web Services in SOA Systems”

By student Georgios Papageorgiou

Advisor: Konstantinos Kokkinos

MSc in Network Computing Program

TEI Larissa & Staffordshire University 2014

Contents

Abstract

Web services are a major part of today's networks and internet based applications. There is a wide variety of web services and development of new ones takes place constantly. End users are called to select from a multitude of services, with their personal preferences imposing on the functionalities and properties of a candidate service. Selecting a service often involves evaluation from a group of end users with different preferences and priorities. This creates a challenge of finding the best tradeoff between the criteria and preferences the users may have.

Most Web services today follow the SOA (Service Oriented Architecture) as an architecture paradigm. The concept of SOA is combining a number of web services developed for a specific task, in order to get a fully operational software application. All services involved in such a system is characterized by attributes such as the task they were designed for, their availability at any point of time, response time, security policies etc. Such attributes provide information for the Quality of the Service (QoS) and are needed for a better selection of the appropriate web service of a SOA system.

The current thesis is focused on a development review, regarding Quality Attributes for web services in a systematic manner. We focus on Quality Models for Web services, Ontologies for web services with Quality of Service (QoS), defining web services with quality attributes, QoS Brokering architectures, QoS monitoring, Web Service Discovery and ranking based on QoS and QoS aggregation in composite web services.

In the second part we propose a tool for monitoring SOA systems capable of being used in several frameworks such as Self-Adaptive SOA Systems and Web Service Discovery Systems. The tool involves monitoring the QoS of a Service, will report the Quality information obtained and will report any Service Level Agreements (SLA) violations.

Keywords:

QoS, SOA, monitoring, quality model, ontology, web services

Abbreviations

API:	Application Programming Interface
DAML-S:	DARPA Agent Markup Language for Services
DARPA:	Defense Advanced Research Projects Agency
IEC:	International Electrotechnical Commission
ISO:	International Standardization Organization
OASIS:	Organization for the Advancement of Structured Information Standards
OWL-S:	Web Ontology Language for Services
QoS:	Quality of Service
REST:	Representational State Transfer protocol
SLA:	Service Level Agreements
SOA:	Service Oriented Architecture
SQuaRE :	Software product Quality Requirements and Evaluation
SWS :	Semantic Web Services
UDDI:	Universal Description, Discovery, and Integration
WS:	Web Service
WSDL:	Web Service Definition Language
WSMO:	Web Service Modeling Ontology
XML:	eXtensible Markup Language

Introduction

Quality of Service is a collection of technologies, techniques and characteristics of a product, that match the needs of service requestors with those of the service provider's based on the network resources available. Typically as part of QoS, a service monitoring system must be deployed, to ensure that services are performing at the desired level.

QoS refers to non-functional properties of Web services such as

- **Availability**: whether the Web service is present or ready for immediate use.
- **Accessibility**: the degree at which a service is capable of serving a Web service request. *Scalability* refers to the ability to consistently serve the requests despite variations in the volume of requests.
- **Integrity**: how the Web service maintains the correctness of the interaction in respect to the source.
- **Performance**: the quality aspect of Web service, which is measured in terms of throughput and latency.
- **Reliability**: the capability of a web service to maintain service and service quality
- **Regulatory**: the web services conformance with standards (SOAP, UDDI, and WSDL), and the established service level agreement.
- **Security**: the quality aspect of the Web service of providing confidentiality and non-repudiation.

QoS is important for services to ensure smooth operation of a Service Oriented System (SOA) system.

A service-oriented architecture is essentially a collection of services that communicate with each other in order to accomplish simple data passing or it could involve two or more services coordinating some activity. SOA provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services and allows the cooperation between these services. This is done through metadata with sufficient detail to describe the characteristics of these services through the Web Services Description Language (WSDL) document and also describing the data that drives them through the Simple Object Access Protocol (SOAP) or the more recent Representational State Transfer (REST) protocol. Use of the Universal Description, Discovery and Integration (UDDI) initiative, allows for service advertisement from the providers and service discovery from the consumer.

The objective of the thesis is focused on two subjects.

1. Development of a review of most significant available research, regarding quality attributes for web services in a systematic manner on the following aspects:
   - Quality models for web services
   - Ontologies for web services with QoS
   - Web service definitions of Quality attributes
   - QoS Brokering architectures
   - Monitoring
   - Web Service Discovery and Ranking with QoS
   - QoS aggregation in composite web services

2. Development of a monitoring tool for SOA systems capable of being used in several frameworks such as Self-Adaptive SOA Systems and for Web Service Discovery Systems for:
   - Monitoring of QoS of a service
   - Report the obtained Quality information
   - Report Service Level Agreements (SLA) violations

PART 1

1.1 Quality Models for Web Services

A Software Quality model is a structured set of Quality Attributes and Characteristics of Software, that are defined and grouped in a hierarchical way. It has the objective to describe, assess and/or predict quality.

Among the many Software Quality models that exist, ISO9126 [1] is one of the most acceptable and new proposed models are mostly based on it. ISO9126 provides a grouped set of characteristics for software quality as follows:

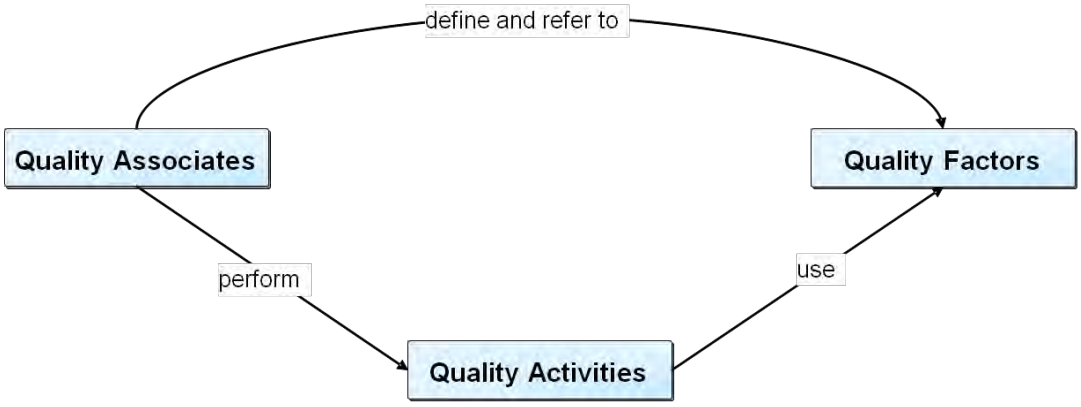
- **Functionality**: Suitability, Accuracy, Interoperability, Security, Compliance
- **Reliability**: Maturity, Fault Tolerance, Recoverability
- **Usability**: Understandability, Learnability, Operability, Attractiveness
- **Efficiency**: Time Behavior, Resource Utilization
- **Maintainability**: Analyzability, Changeability, Stability, Testability
- **Portability**: Adaptability, Installability, Co-existence, Replaceability, Conformance

ISO9126 was first published in 1991 and revised in 2001 as ISO/IEC 9126-1 to 9126-4 [2]. At the same time work on SQuaRE, has introduced the replacement ISO/IEC 25000 [3] and in 2011 the most recent ISO/IEC 25010 [4] which expands the set of Quality Characteristics as follows:

- **Functionality**: Suitability, Accuracy,
- **Reliability**: Maturity, Availability, Fault Tolerance, Recoverability
- **Performance**: Time Behavior, Resource Utilization
- **Operability**: Appropriateness recognizability, Ease of use, User error protection, User interface aesthetics, Technical learnability, Technical accessibility
- **Security**: Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity
- **Compatibility**: Co-existence, Interoperability
- **Maintainability**: Modularity, Reusability, Analyzability, Modifiability, Testability
- **Portability**: Adaptability, Installability, Replaceability

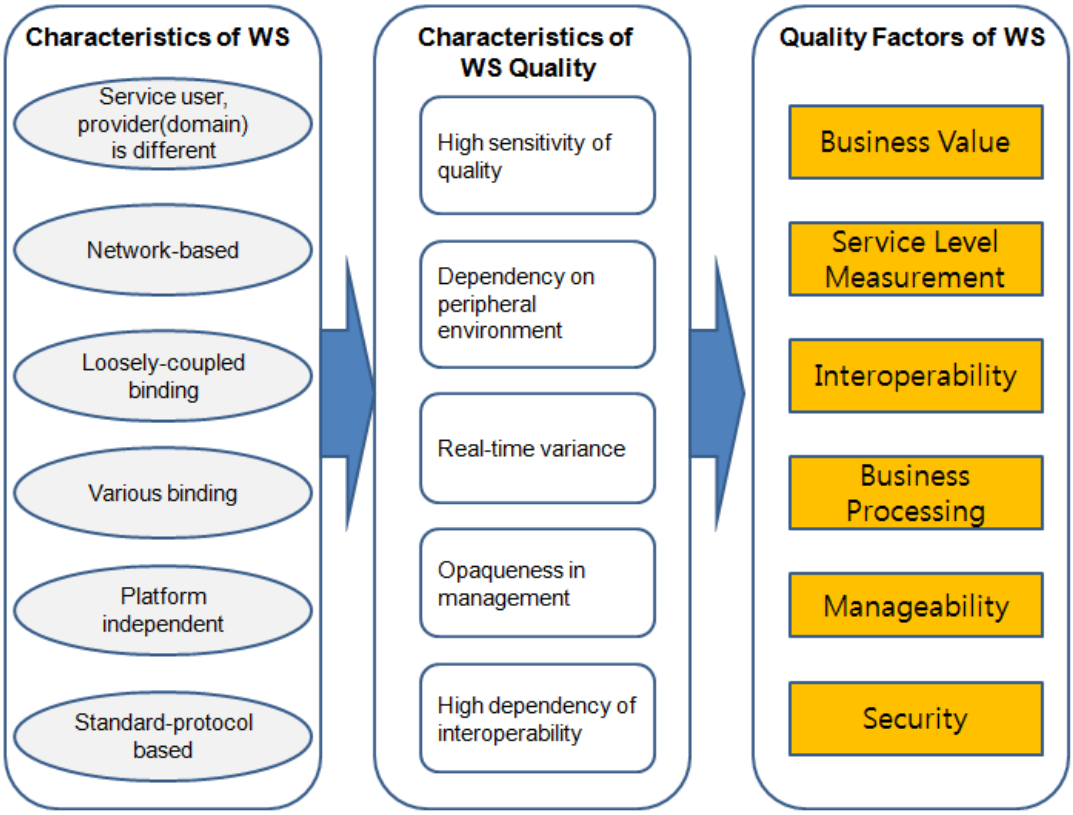
Since these models were designed for software in general, and not specifically for Web Services, some Attributes, such as Installability and Co-existence, do not always apply to Web Services.

OASIS has proposed another Web Services Quality model, based on ISO9126, named WSQM [5][6]. Apart from Quality factors WSQM is also dealing with Quality Activities and Quality Associates. As in shown in figure 1:



(Figure 1) WSQM parts schematic representation. [5]

- **Quality Factors:** Describes the characteristics, sub characteristics and attributes of Quality for Web Services
- **Quality Activities:** Are the activities performed during the lifecycle of a web service regarding to their QoS.
- **Quality Associates:** Are the persons or organizations that are involved in Quality Activities



(Figure 2) Extracting Quality Factors of WS [6]

The Quality Factors as described in WSQM are as follows:

- **Business Value Quality**
  - Price
  - Penalty and Incentive
  - Business Performance
  - Service Recognition
  - Service Reputation
  - Service Provider Reputation

- **Service Level Measurement Quality**
  - Response Time
  - Maximum Throughput
  - Availability
  - Accessibility
  - Successability

- **Interoperability Quality**
  - Standard Adoptability
  - Standard Conformability
  - Relative Proofness

- **Business Processing Quality**
  - Messaging Reliability
  - Transaction Integrity
  - Collaborability

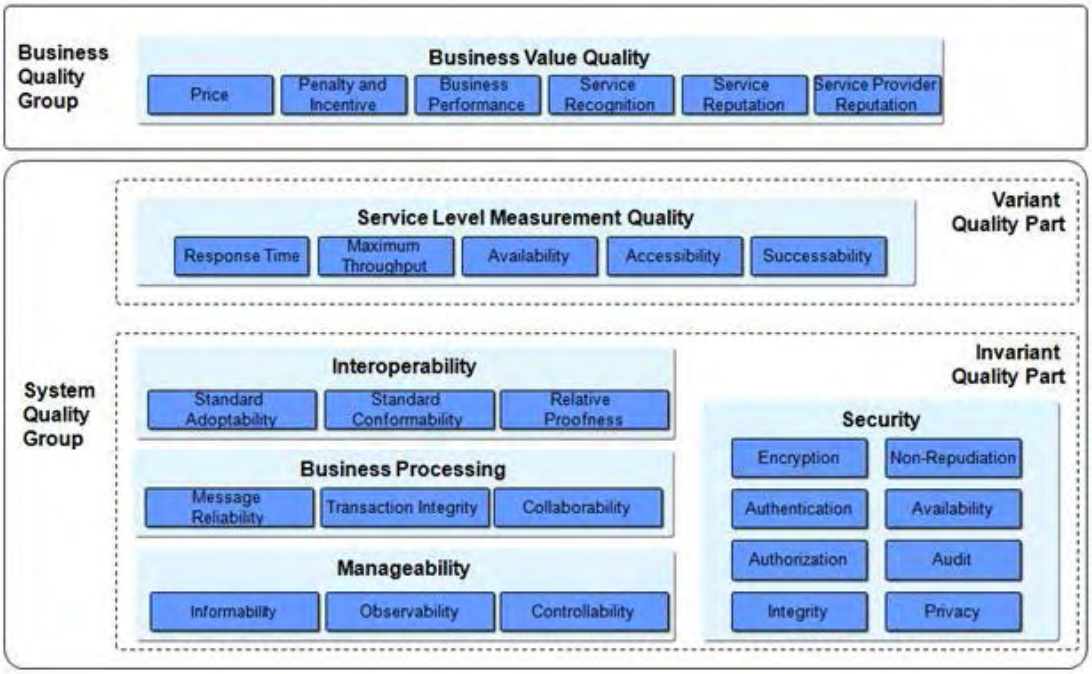
- **Manageability Quality**
  - Informability
  - Observability
  - Controllability

- **Security Quality**
  - Encryption
  - Authentication
  - Authorization
  - Integrity
  - Non-Repudiation
  - Availability
  - Audit
  - Privacy

While these models are complete, they are not widely established as a standard for web service models. ISO9126 is not specifically designed for Web Services, whereas ISO/IEC 25010 and WSQM are still in a working draft stage.

Another observation is that, different Quality model proposals use different names for the same Quality Attribute or Characteristic. For example *correctness* and *accuracy* are used to describe the same thing. Also some sub-Attributes are related to different Attributes such as Interoperability is under Functionality in ISO9126, whereas in ISO/IEC 25010 and WSQM is under Compatibility.

Due to lack of standardization for a Web Services Quality model, most research work on Quality models, takes into consideration only a few of the Quality Attributes used on the aforementioned models, some even use their own implementations. As such there is not a normalized interpretation of attributes.



(Figure 3) Graphical Structure of Web Services quality factor of WSQM [6]

1.2 Quality attributes and metrics

Depending on the perspective from which quality is researched, it can have different meaning [7].

- Quality can be an abstract, meta-physical ideal, which is unreachable but portrays the goal of where products aim.
- Quality from a manufacturer view that shows the compliance with the ISO 9001:1994 [8].
- Value based perspective, which depends on the stakeholder for whom it is defined.
- It can be the perspective of user attributes with the relevant context of use
- Product internal characteristics, based on measured attributes.

The main focus of the current QoS research lies mainly with the two latter views that deal with attributes and metrics.

Quality Attributes cannot be described by simple numbers, but each one consists from their own sub-set Quality Metrics. Quality Metrics are defined as quantitative measurable aspects of software that can provide the amount to which a Quality attribute is achieved for that software.

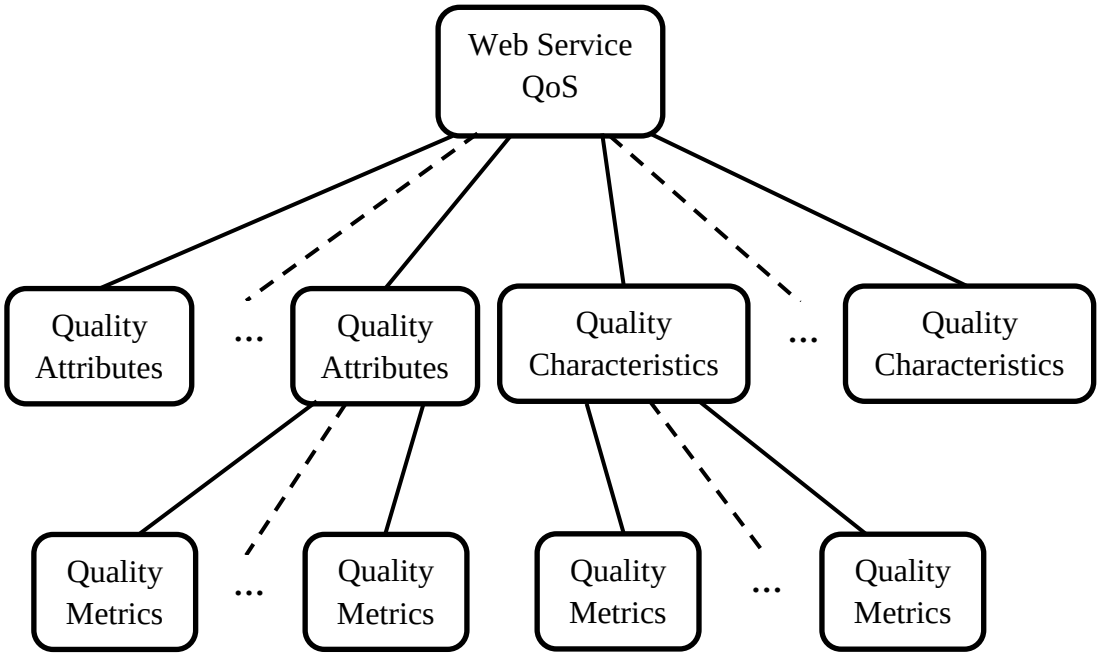
Apart from the taxonomy of quality attributes, attributes can be classified into

- Internal WS Quality
- External WS Quality
- WS Quality in use

Internal WS Quality, mainly aims to the Service Provider, and concerns attributes and parameters that are static and obtained by advertising. Such Attributes can be security, interoperability, usability, scalability and cost.

External WS Quality, is also mainly related to the Service Provider, and concerns mostly dynamic-observable metrics obtained through monitoring or probing the Web Service. Such attributes can be Availability, Response Time, but also include attributes from internal WS Quality in a more dynamic measurement approach.

WS Quality in use, concerns metrics and attributes provided by the user. Such characteristics can be based on the context of the user, such as weather forecast which is approached differently by someone who plans a long journey and in another way for planning his weekend. Quality in use is usually derived from the average of user opinions and can concern accountability and accuracy.



(Figure 4) Schematic hierarchy of quality components

1.3 Ontologies for Web Services with Quality of Service (Qos)

In the previous section we saw that quality models define quality attributes and characteristics for web services. In order to fully use quality attributes, a simple description and declaration of these attributes and metrics is not enough. We need to define their relations to one another in the same domain. To describe such information ontology is used that allows for a better description. Having a common base of description for the various metrics of a web service, is important in order to allow sharing this information between service providers, service clients and service brokers.

Until recently the approach of existing research work was around single attribute. In the last years, research increasingly focuses on a more semantic study of the quality factors and follows a more multi attribute approach. Thus ontology models become more and more important for the future of web services. Semantic web services will allow machines not only to read the data, but also provide some understanding of that data, allowing for better web service discovery, composition and invocation.

There are a lot of available ontologies for web services available but only a few of them allow for QoS accommodation. Thus various proposals of ontology models for web services have been proposed with QoS consideration.

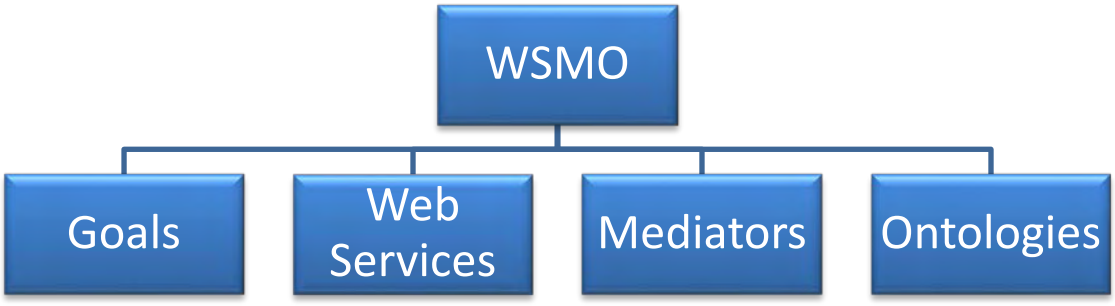
One of the most recent and promising approach is the Web Service Modeling Ontology (WSMO) [9]. It has been built with defacto Web standards for syntax addressing and communication protocols such as XML. Thus WSMO complements existing syntactic Web service standards by providing a model and language for the semantic markup of Web services.

Presentation and publication of machine readable declaratives can be used by developers and applications alike. This allows for easy interaction between software components and different platforms. Each term's syntax, static semantics and dynamic semantics, is defined across a number of documents using a Meta-Object Facility (MOF) [10] framework.

WSMO acts as a semantic meta-model, for services based on MOF. It defines an abstract Web Service Modelling Language (WSML) [11] and framework for specifying, constructing, and managing technology-neutral meta-models and consists of four core elements (figure 5)

- **Goals:** are descriptions of an objective, with the aim to fulfill it through the execution of a Web service, so that user desires can be satisfied.
- **Web Services:** the semantic description of the web services including their functional and non-functional properties, as well as information for interfacing with them.

- **Mediators:** describe the connecting entities between the different WSMO elements. They resolve heterogeneity problems of components of a web service on the data, process and protocol level.
- **Ontologies:** provide the terminology used by all other WSMO elements at a meta-level and support their description.



(Figure 5): The top-level elements of WSMO [9]

WSMO uses natural language for the descriptions which may lead to small deviations in the model definitions due to ambiguities in the way of representation of descriptions. This lack of precision in defining the semantics of WSMO can result in a difference in comprehension and hence interpretation from different users for the same WSMO model. Also the multitude of documents comprising the description of the model terms causes redundancy and possibly contradiction, making it difficult to extend and revise the descriptions.

In [12] the authors propose a more formal model of WSMO using Object-Z (OZ) [13]. Object-Z (OZ) is an extension of the Z formal specification language that accommodates object orientation and can enhance structuring in order to improve the clarity of the descriptions. Their proposed WSMO extension provides a single formal model for the syntax, the static semantics and the dynamic semantics, in place of the independent definitions in WSMO. The general approach is to define aspects of WSMO as Object-Z classes. WSMO syntax is captured by the Object-Z class attribute, Class Invariants define the static semantics of WSMO and Class Operation defines the dynamic semantics. And these three in turn are enclosed in a single class, that is much easier to utilize and revise.

QoS in WSMO is defined as non-functional information in the web service description but lacks the flexibility and expressiveness needed for QoS attribute definition. WSMO-QoS [14] has been proposed as an extension of WSMO, that gives a better rendition of the QoS class, by introducing a sub-class to it (Figure 6).

Class QoS **sub-Class** nonFuntcitonalProperties
hasMetricName **type** string
hasValueType **type** valueType
has MetricValue **type** value
hasMeasurementUnit **type** Unit
hasValueDefinition **type** logicalExpression
multiplicity = single-valued
isDynamic **type** bollean
isOptional **type** Boolean
hasTendency **type** {small, large, given}
isGroup **type** Boolean
hasWeight **type** string

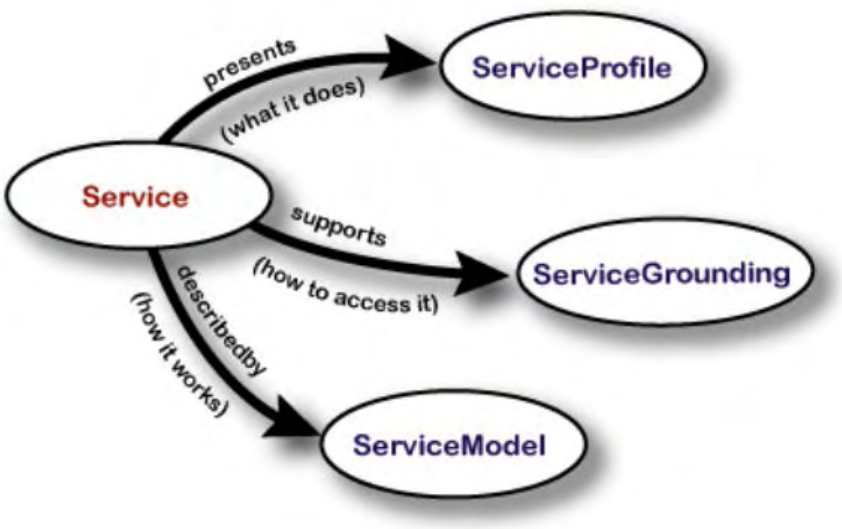
(Figure 6) QoS Ontology in WSMO-QoS

A different approach by Zhou et al. extending the DAML-S [15] ontology to incorporate QoS descriptions, proposed the DAML-QoS [16] ontology. Their proposal is made from three layers

- **QoS Profile Layer** has the purpose of matchmaking Service providers to Service Requesters
- **QoS Property Definition Layer** this layer defines the property name as well as its domain and limit constraints
- **QoS Metrics Layer** this layer is responsible for defining metrics for the QoS property and to provide semantic meanings for the measurements of metrics.

Since DAML-QoS is an extension of DAML-S, service providers and service requesters can use the strength of both ontology models utilizing DAML-S for semantic matchmaking and DAML-QoS for QoS based matchmaking.

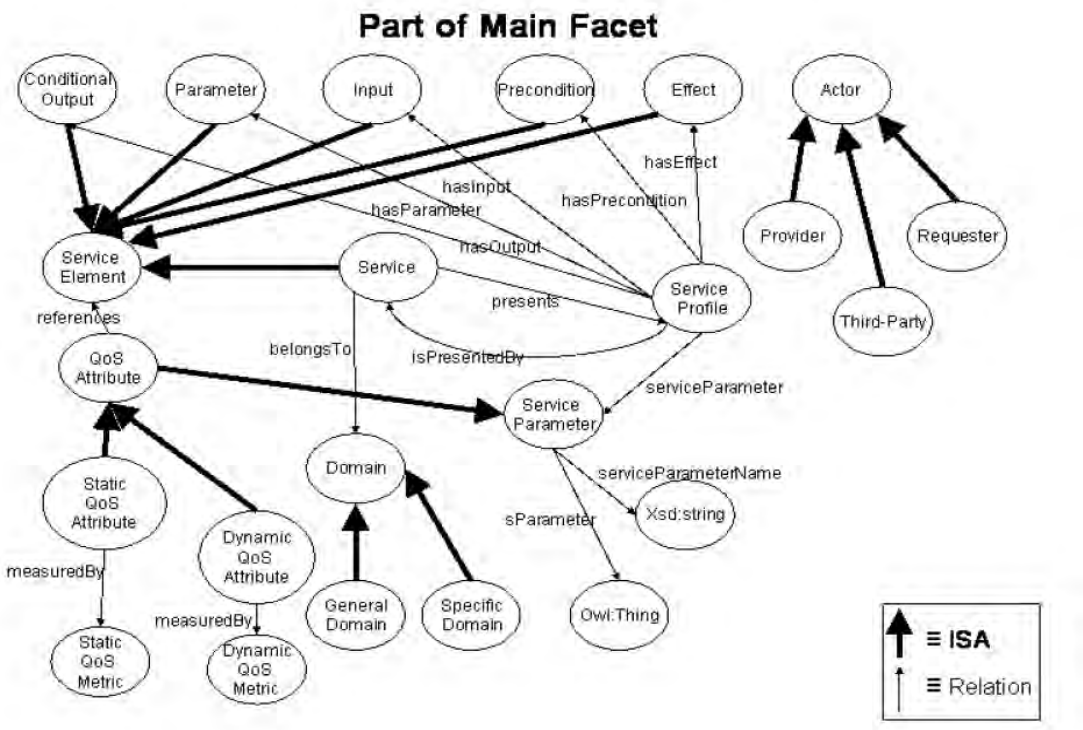
At the time DAML-QoS was proposed a newer version of DAML-S was introduced OWL-S [17] which has become a very popular ontology approach for Web Services and has essentially replaced the older DAML-S. OWL-S is based on the OWL[18] framework for semantic Web, for describing Web services. OWL-S provides three essential types of knowledge about a service (Figure 7). What the service does is described in the ServiceProfile, instructions of how to access the service in the ServiceGrounding and the inner workings in the ServiceModel.



(Figure 7) Top level of the OWL-S service ontology [17]

OWL-S similarly to WSMO does not incorporate detailed description for QoS. Some proposals have been made to make up for it by extending the OWL-S. kritikos et al. in [19] propose OWL-Q as an extension of OWL-S. OWL-Q is comprised from eleven facets each of which focuses on a specific part of the QoS description for the Web Service. These are:

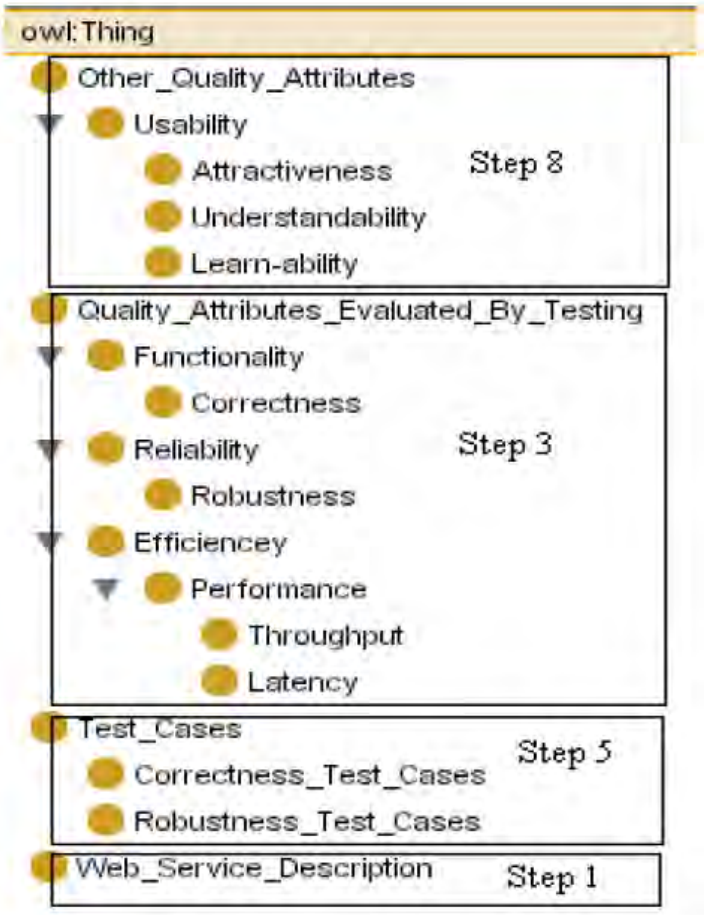
- **OWL-Q (main)** which is responsible for connecting OWL-S and OWL-Q and provide the high level QoS concepts
- **Measurement Directive** contains the specifics of measuring metrics.
- **Time** specifies a schedule for determining the frequency of complex metric computation
- **Goal** which defines QoS goals and constraints and mathematical formulas
- **Fuction** contains functions related to producing and checking metrics as well as functions that concern scale and transformation of one scale to another for metric comparisons.
- **Measurement** is where all concepts of the OWL-Q are defined to allow their storage and statistical processing by registries
- **Metric** defines the classes and properties of any QoS metric
- **Scale** controls the type of metric value and the type of operation allowed for metrics
- **QoSSpec** which defines QoS offers and requests
- **Unit** describes the unit of a ratio scale for a QoS metric
- **ValueType** defines the types that a QoS metric can take



(Figure 8) Part of Main Facet of OWL-Q [19]

G. Fortes proposes the QoS Model Ontology QoS-MO [20]

Some proposals do not follow OWL-S or WSMO ontologies but propose their own such as the proposal in [21]. As the authors mention, access to Web Services source code is limited to providers but service requesters do not have access to it. This lack of accessibility for the requester limits the testing techniques that can be performed. Their proposal allows the description of web services in a way that the requester can understand what the service can do, how to use it and what testing can be utilized on it, while taking QoS into consideration.



(Figure 9) A fragment of the taxonomy of the classes of the proposed ontology in [21]

1.4 Web Services definitions with quality attributes

For the definition of Web Services the used standard is Web Service Description Language WSDL [22]. WSDL is an XML based language proposed by the World Wide Web Consortium (W3C), that describes the functions of a web service and the data format of the messages it uses in an abstract way. Essentially WSDL is a mechanism used to publish a web service name and its XML schema (XSD) so that it can be used with the SOAP protocol. Although the description in the WSDL document describes the way to interface with the service, it does not give any QoS information for the service.

While WSDL is used to publish the information for a service, the UDDI (Universal Description, Discovery, and Integration) standard is used to store such information in an XML-based registry, where, publishers can advertise their services on the Internet using WSDL. Clients can then search this registry for the most suitable service by keywords. Since WSDL does not contain QoS information though the search in the UDDI might not return services that satisfy the QoS needs of the client.

Some proposals have been made in order to incorporate QoS information, by extending the WSDL document. The WSDL document contains an abstract description of the service interface operations with message exchange patters and parameter types. It also describes the bindings that constitute the connection type of the service interface. Finally endpoints associate these bindings to a network address through which the service can be invoked.

In [23] the authors propose extending the WSDL document, at service level rather than at interface level, for specifying non-functional properties. They add a new element in WSDL that contains the QoS details and parameters and use that to extend the endpoint of the service. A similar approach is used by Kang [24] by using annotations to extend WSDL for QoS description.

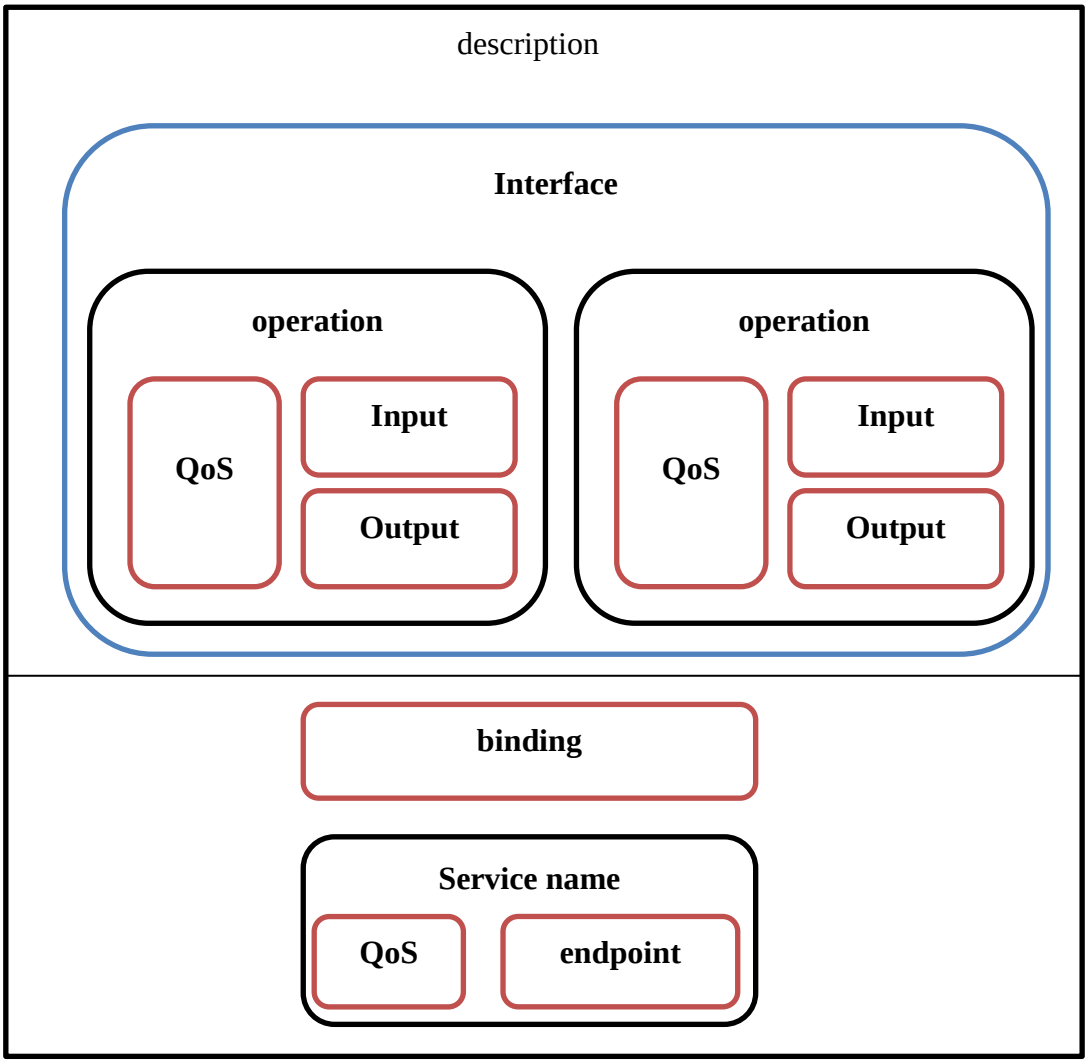
In [25] Xinmin Lin et al. propose a Dynamic QoS Management Model (DQMM), which allows for a dynamic domain QoS description with real time acquisition of QoS metrics. The model would allow for more accurate QoS information for domain related services.

```
<xs:complexType name="criteriaService">
  <xs:complexContent>
    <xs:extension base="qwdsl:ExtensibleDocumented">
      <xs:sequence>
        <xs:elementname="performance" minOccurs=0 maxOccurs ="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:NCName" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="performanceType">
  <sequence>
    <element name="ResponseTime" type="qwsdl:ResponseTimeType"
minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Offered" type="boolean"/>
</xs:complexType>
<xs:complexType name="ResponseTimeType">
  <attribute name="value" type="float"/>
  <attribute name="unit" type="string" fixed="sec"/>
  <attribute name="category" type="qwsdl:CategoryType"/>
</xs:complexType>
```

(Figure 10) QoS-Aware WSDL schema for Performance parameters [23]

A different approach of extension for WSDL was proposed by Joel Farrell with SAWSDL (Semantic Annotation for WSDL and XML Schema) [26][27], which defines how to add semantic annotations for the operations and interfaces described in a WSDL document. An ontology can be used to utilize the semantic annotations and thus allow Web Service discovery and composition. Most proposed frameworks offer their own extensions for WSDL QoS representation, but there are a few based on SAWSDL [28][29][30].

Ravi Shankar Pandey expanding on work done by Wu [30] created a tool [31] that uses a simplified version of the WSDL meta-model proposed by D'Ambrogio [32]. He uses this meta-model to add QoS information for the service as well as the operation into the WSDL document that describes a service, while retaining the structure of the document (Figure 11).



(Figure 11) WSDL structure with QoS proposed by Pandey [31]

Other approaches trying to extend WSDL are using Q-WSDL [33] to achieve automated discovery based on QoS. In [34] X-WSDL an extension for the WSDL tries to add QoS criteria information as a new xml element in the service description to allow better service discovery depending on requirements.

OASIS have also presented a working draft version for a WSDL extension, the Web Service Quality Description Language (WSQDL) [35]. Based on their proposed WSQM [5] model, they incorporated the quality attributes using WSQLD. The WSQDL schema can be found at [36]. The goal of WSQDL is to provide a standardized form for representing quality metrics and characteristics.

Some UDDI extension approaches have also been made though not so popular as WSDL extensions. The authors in [37] present a broker architecture that allows service selection based on QoS parameters. A UDDI extension using a tModel

(Technical Model) for the QoS information enables the broker system to make service selection depending on functional and non-functional requirements.

A sample of the proposed tModel as mentioned in [37] shows the QoS information of a categoryBag object with the four metrics considered by their model.

```
<tModel tModelKey = "somecompany.com: Stock
QuoteService:PrimaryBinding:QoSInformation">
  <name>QoS Information for Stock Quote Service</name>
    <overviewDoc>
     <overviewURL>
       http://<URL describing schema of QoS attributes>
     </overviewURL>
    </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="uddi:uddi.org:QoS:Price"
      keyName="Price
      keyValue=" 0.01" />
    <keyedReference
      tModelKey="uddi:uddi.org:QoS:ResponseTime"
      keyName="ResponseTime"
      keyValue="0.05" />
    <keyedReference
      tModelKey="uddi:uddi.org:QoS:Availability"
      keyName="Availability"
      keyValue="99.99" />
    <keyedReference
      tModelKey="uddi:uddi.org:QoS:Throughput"
      keyName=" Throughput"
      keyValue="500" />
  </categoryBag>
</tModel>
```

OASIS and Adam Blum propose a tModel [38][39] that combines two existing techniques for QoS to be added to UDDI. Using of external resources and Multiple categories for QoS attributes. Their approach is to include a multitude of categories for the various QoS attributes in the tModel and reference to tModel that will have a taxonomy for the categories. This will allow better manageability in case of too many categories, as well as standardise the location of external reference.

Another UDDI extension approach is the UDDIe [40] that adds meta-data dynamic properties for describing QoS of services. UDDie is one of the most important contributions regarding QoS incorporation in approaches of UDDI extension.

Parimala's and Saini's approach extends both WSDL and the UDDI by introducing the X-WSDL [41] and X-UDDI [42]which is also based on UDDIe. In X-WSDL two elements are added in WSDL that contain information for QoS criteria and their description. Then this new WSDL schema is mapped to the X-UDDI.

The author of [43] extends with quality criteria the *Service Implementation Document* part of WSDL while also adding a quality server for managing UDDI QoS characteristics.

1.5 QoS Brokering architectures

Apart from expanding WSDL and UDDI another method to use QoS in service description is using brokers. Brokers act as a middle man between providers and clients and their role is to find services that have the criteria requested by the users. They require a repository of advertised services like UDDI or some alternative service registry, where they can look to find the service matching the criteria. This method is the dominant discovery procedure of today's systems.

Brokers relieve the clients from the task of messaging each server application even if these are Web Services. As they collect the data from the servers, they form a single output XML that has the specifications of the XSD of the service and this is send to the client as a response to the client request.

The authors in [37] Use UDDI extensions to create a broker architecture that is tasked with verifying and certifying the QoS information provided by the publisher. Once the task has completed then it register them in the UDDI registry.

A model that can take requirements into consideration during Web Service discovery was proposed by Diamadopoulou et al. [44] which can add QoS information in the WSDL or from the database of a service selection system that acts as a broker.

Rathore et al. [45] created a broker based model for web service composition. Their model assigns to the broker the tasks of verification, certification, service registration and publishing provided they match the quality criteria that are set. It has a function to monitor the QoS of the service in order to verify if the advertised QoS is valid.

Yu et al [46] propose a broker approach that is not based on UDDI but has their own service registry approach. They define a QoS broker mechanism for managing QoS of composite Web services.

Chen et al [47] presented a broker solution to alleviate the problem of using API for accessing backend servers. Their approach relieves the backend servers from considerate overload since the broker shares information between services. The brokers also maintain a copy of service information including QoS in its own registry, which further reduces traffic to backend servers.

Vadivelou et al propose a delegator broker system that emphasizes in balancing the load for service selection. It uses WSDM (Web Service Distributed Management) to monitor the quality parameters that the user specifies in order to select the best service that matches the requested service quality and dynamically register them to the UDDI registry.

According to the authors in [49] the current approaches are not scalable enough regarding service selection. Their heuristic approach tries to cover for that drawback by introducing distributed brokers.

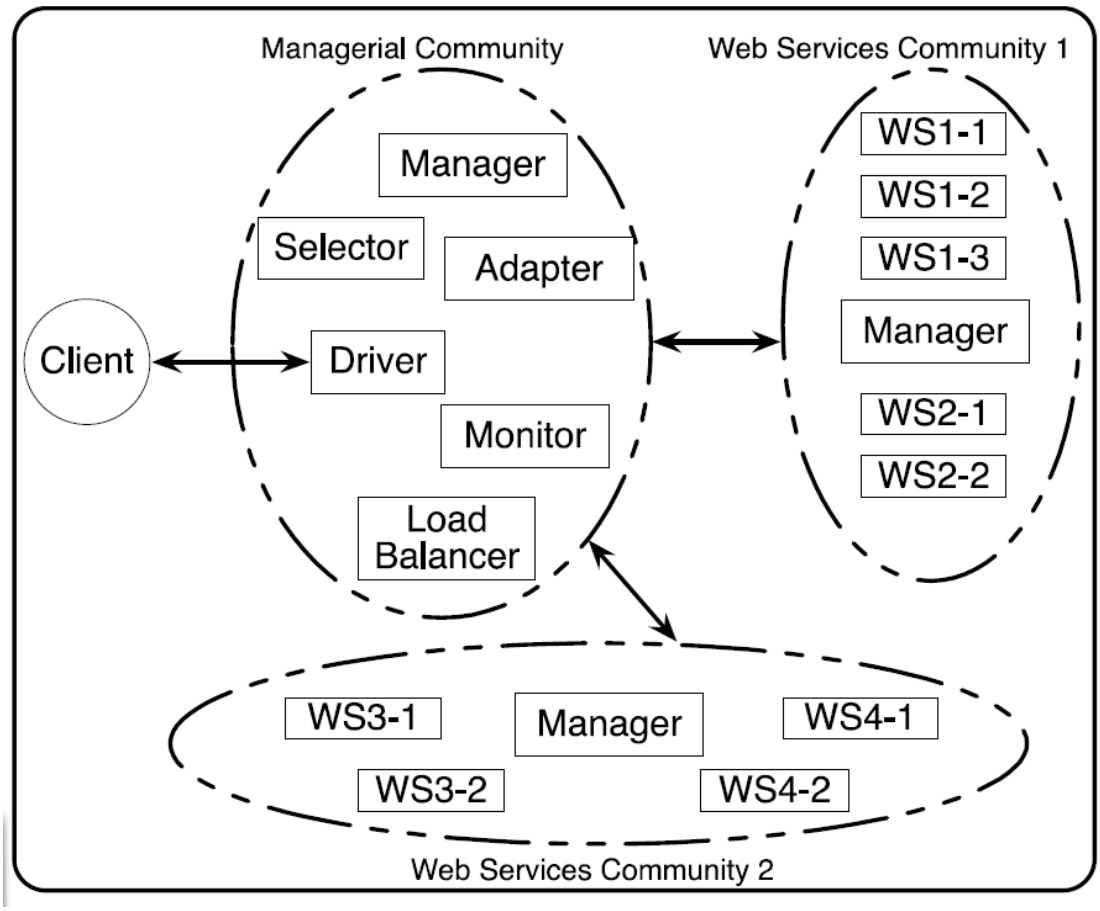
Almost all the proposals include a monitor component in order for the brokers to obtain real time quality information of the services that are members of their web service registries. The brokers use the QoS details in order to evaluate the services based on user requirements so that the service best matching the request can be selected or in some cases a group of services to compose a complex service.

1.6 Monitoring

As mentioned previously, broker architectures use monitoring in order to obtain quality measurements in real time for non-static attributes. Such monitoring usually takes place during service discovery or in service selection. Monitoring can take place in two different ways. Passive and Active.

Passive monitoring mostly observes the messages that services exchange between them or with clients. Thus the monitor only gets quality information that relate to real operations of the service. This means that the operation of the service is not hindered in any way as it uses the already ongoing traffic exchange to get quality metrics. The drawback of passive monitoring is that there is no control on when or how the measurements will be taken. Passive quality monitors are focused on specific services for which they are developed. As such they only operate within the boundaries of these systems and are hard to expand for other systems. This makes them more cumbersome in favor of more realistic quality measurements and dependable on what has been declared as testable.

Active monitoring also known as online monitoring, uses testing methods and as such it requires sending requests to the tested service in order to get responses and determine quality of service. Measurements are taken from the transaction based on its performance, like response times and throughput. Active monitoring allows better control of when and what functionality to test and thus is usually used for operation evaluation of functional requirements in order to determine robustness and behavior of a system in extreme condition scenarios. These tests are used to predict or detect the operational limits of a system. Due to the intrusive nature of active tests, the measurements are not as accurate as in passive monitoring but on the other hand provide total control for the conditions of the tests. Active tests are especially useful during development of services due to the fact that they can predict inadequacies that could result in service failure.



(Figure 12) Managerial Community and two Normal Communities [50]

The authors in [50] propose a managerial community of services with QoWS quality properties. Web Services are monitored by observation of the client request by the monitor. During the observation the service is checked for any inconsistencies of the expected information and reports it to the WSM (Web Service Manager) component of the proposed model architecture.

Arlitt et al [51] present a passive crowd-based monitoring system that gathers data from visited services in order to acquire measurements. Similarly in [52] the monitor tool proposed is tracking the packets exchanged between the server and the client to acquire quality information. They offer two different placement possibilities one at the common input output service port as a service and the other as a server component in which case it might compete for cpu cycles with the server itself.

This raises the question of where to apply passive monitoring in a SOA system. As we saw until now there are some possible integration options for monitoring tools. They can be part of the SOA system, they can be applied on the service side or similarly on the client side. There is also the possibility of being applied on the broker side which is becoming more popular due to central service bus used by modern distribution systems. Regarding the server side monitoring there is the issue of tampering with the

quality attributes so in this area [51] and [52] that are based on collaborative or crowd based QoS information gathering, coupled with some ranking broker mechanism can give more reliable results. The alternative being, the client side monitoring which is more trustworthy but limited in range of service ranking.

A proposal by Jurca et al [53] combines the logic of these solutions by performing client side monitoring which in turn are passed to a reputation repository where data from multiple clients are gathered. The data are compared and the final ranking is advertised to a service registry like UDDI.

Zeginis and Plexousakis propose a method of monitoring a service using SLA [54] in order to check service QoS. This method allows to better determine what goes wrong with a system by reporting SLA violations. The exact reason of what goes wrong is more easily identifiable and pinpointed to a subsystem.

IBM has proposed a monitoring solution [55] that uses an independent from the client and provider monitor and works similarly to a broker. The client in order to get monitor information makes the request to the monitor which in turn is responsible of making the appropriate calls to the service. This works similar to brokers being the middle man in service discovery and thus is a very popular logic used by a lot of researchers.

Due to the middle man similarity with brokering some proposals incorporate the monitor functions in the service discovery broker so that it has a dual role of acquiring QoS and passing the service requests.

The authors of [56] propose a monitoring system that can be applied as an active monitor on the client side, or as a passive monitor on the server side. The proposed framework can additionally monitor SLA violations and report them. The duality of the monitor being a client-server implementation can give a more complete image of the system that is monitored.

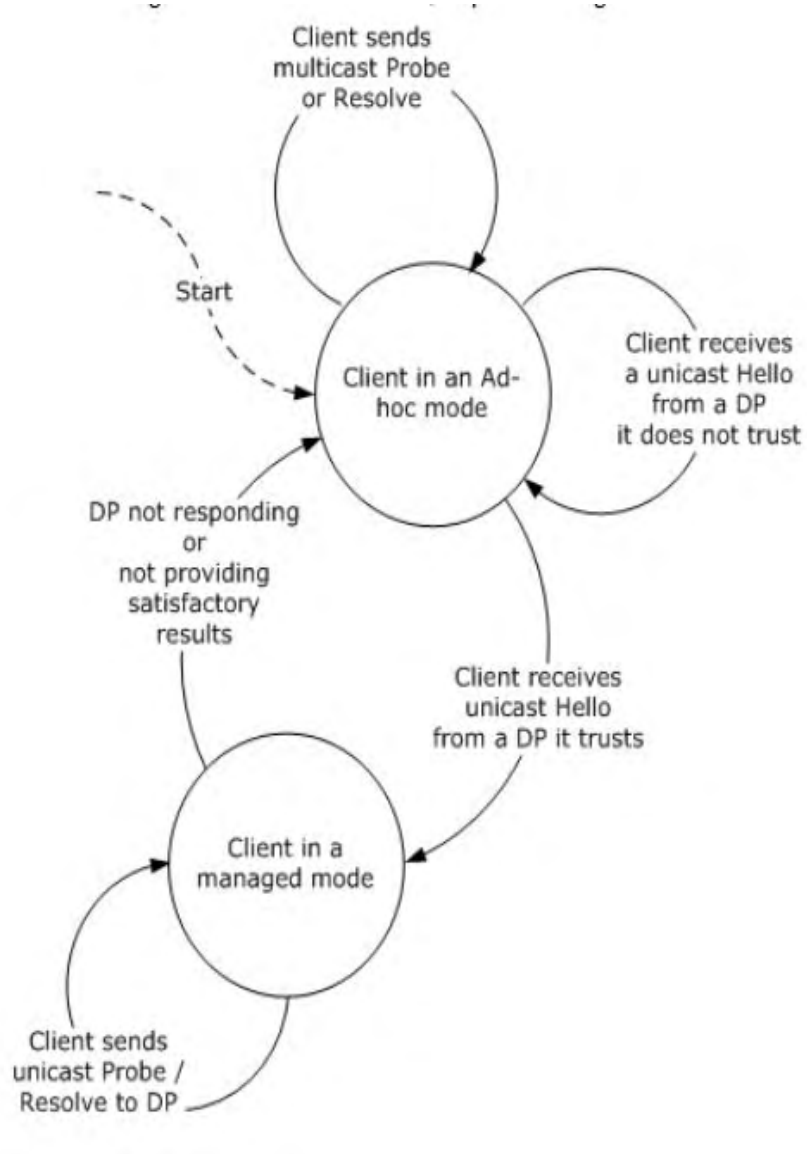
Zhou et al. [16] proposed a DAML-QOS based monitor tool that does not need to be included in the service code but is automatically generating the tool by interfacing with the engine of the service. This brings forth the issue of monitoring tools demanding language specific instructions that are depending on the services technology.

This issue was tackled with by Zeng in [57] that proposed using ECA rules (Event Contition Action). According to the authors since ECA does not enforce low level mechanics to the users it is easier for them to perform monitoring.

Expanding on the solution are by Zeng is the use of ECE (Event Condition Expectation) [58]. The authors propose a variation of the traditional Event-Condition-Action rule pattern.

1.7 Web Service Discovery and ranking with QoS

OASIS proposed a specification for a protocol used to locate services. The protocol works in an ad hoc mode, in order to find a service based on the type of the service, and a scope that the service resides. The client sends a probe message to a multicast group of services and any service matching it responds to the client. The other method is to find a service with a specific name. As before the client sends a message to the multicast group and the service that matches the name sends a response to the client. Alternatively to the ad-hoc scenario they propose a managed method using a proxy to send the messages. The protocol can switch between the two modes at any time.



(figure 13) State transitions of a Client configured to dynamically switch operational modes. [59]

Zhou et al. [33] propose a framework for Web Service discovery by using an agent. Users gather data as they use new services and they sent these data to an agent. The agent then publishes the information to a UDDI. The authors introduce an algorithm that allows better selection of services without the need to visit the registry services. The agent gets the user requested characteristics for the service and matches it with the stored descriptions. Then an optimization takes place to find the most appropriate services matching user demands.

The authors in [31] based on their WSDL extension propose a method for Web Service discovery using MDS and adding QoS information support. The QoS properties can be registered to aggregators for discovery from the MDS which then passed the information to a transaction tool for retrieving the service data before sending them back to the user.

1.8 QoS aggregation in composite Web Services

Composite services are services that are composed by other services. Essentialy they are the representative of SOA systems. In order to create such services first the needed services must be discovered based on criteria. Such services must be evaluated for their functional requirements and their QoS requirements. An algorithm is needed thus to make this classification.

The authors in [61] expand on their previous proposal of consumer-centric QoS-aware selection, QCMA that used a fuzzy opinion similarity and QoS preference with a number of QoS attributes as defined by W3C. However this approach does not cover some aspects of web service selection such as:

- The opinion group does not take into consideration the diversity of the group population in terms of background and perception. This makes QoS criteria ineffective for web service selection.
- The single group consensus in QCMA does not provide adequate QoS opinion similarity and preference order when dealing with multi-group opinions, thus needing further criterion review and analysis.
- Multi-attribute based identification is more difficult than single-attribute outlier identification. This leads to the need of reclassifying the single-attributes to better relate to other group opinions.
- SAM used in QCMA is very complex with a large number of inputs. It requires a vast number of calculations for the similarity analysis of QoS opinions, which can lead to problematic system scalability.

Expanding on QCMA the authors propose the FMG-QCMA that uses fuzzy clustering, SAM analysis and RMGDP QoS preference order analysis for multi-group consensus and quality of grouping measurement.

FMG-QCMA allows for greater similarity of member opinions within the same group, than it is between members of different groups. Also the similarity spread will be lower. This allows for reduced computational complexity and better web service selection.

Before FMG-QCMA takes action it needs to get the initial Fuzzy QoS opinion for each consumer. For this to be done each consumer has a profile that contains details for his selections in the past, and his disposition towards the 13 QoS attributes defined by W3C. For each attribute the consumer provides his preference order as well as 4 values on a scale of 0 to 10 to describe his QoS requirements. These are:

- Absolute minimum attribute value services below that are ignored
- Preferred minimum value of attribute
- Preferred maximum of attribute value
- Highest attribute value, above witch service performance is not needed

The fuzzy QoS opinions parameters can be changed and then reevaluated by the system in later stages if needed.

FMG-QCMA then proceeds to create subgroups based on the similarity of consumer QoS opinions. Opinions can belong to multiple subgroups depending on the similarity between these groups. If an opinion belongs in a single group no further analysis for it will take place as it has correctly been classified. If it does not fully belong to a certain group, then groups that have similarities are taken into consideration. Thus by using FMGSAM algorithm further attempts to fully classify the opinion are taken until the most appropriate group is found or a preset calculation limit has been reached. The procedure is also done when changes in the customer profile take place, altering his opinion boundaries, raising the need for reclassification.

After grouping has reached a satisfying level, preference order of attributes can be taken into account via RMGDF algorithm.

Finally the system is ready to propose the most appropriate services to the consumers based on the classification of their QoS preferences.

1.9 Review conclusions

After studying the existing work presented in this survey the following results have been reached.

On the topic of Quality model there is not a standardised choice that sets a globally accepted solution to define QoS for Web Services. There is a multitude of proposals that have been made but none of them is yet accepted as a standard by the community. On that regard the most promising solution is currently presented by the OASIS group with their WSQM proposal. While it is not wholly accepted yet, parts of it are already used by many of other work related to Web Service QoS. Coupled with their proposals in other areas of Web Services for SOA systems that are based on their model it is the most promising proposal currently.

A step after determinitng QoS attributes must also be taken so that the quality metrics set in a quality model can actually be usable. This requires an ontology that will allow quality attributes and characteristic to be given meaning. There are some languages that allow a semantic perspective for viewing quality attributes. In the past, work usually dealt with single attributes but since modern systems have to take into consideration a multitude of quality attributes ontologies are the only way to relate such attributes to one another in a semantic way. The most used language for ontologies is OWL-S but since it lacks QoS information support many extensions have been developed as well as some alternatives. The most promising and complete ontology on this aspect currently is in my opinion WSMO which has already been accepted by W3C.

Once the semantic description of quality attributes has been determined the next step is to incorporate these to the Web Service description itself. Thus the WSDL and UDDI must be extended to include QoS information. There is a multitude of proposals for extending WSDL and UDDI by adding new elements to them although none is yet widely accepted. OASIS proposed their own extension based on their Quality model WSQM and considering that they cope with both areas of research I find it to be the most promising option together with SAWSDL currently available for WSDL. SAWSDL is used in a lot of research by using annotations for QoS integration which is a commonly used technique on the subject. In a similar regard UDDIe is a proposal for adding QoS to the UDDI. Due to the fact that it uses tModels which is the preferred method of extending UDDI by the research community UDDIe is a really promising technology.

The most widely used method for acquiring QoS information from Web Services as well as for service discovery and selection is brokering. These delegate between client and provider and incorporate monitoring techniques in order to get the most accurate QoS information for the services and select the most appropriate according to the client needs.

The need to acquire real time QoS information is demanded in order to help in service discovery. There are many Monitoring solutions mostly of the passive philosophy but since they lack extensibility for multiple framework support, the research community focuses more into active testing, or an intermediary of the two using cloud consensus logic.

Service discovery is needed to find the appropriate services that would match the user demands. This requires ranking od QoS for Services. On this subject there are not enough proposals the little existing not being recognized or seeing any use. This is still an open for research field.

Together with Service discovery comes the question of Composite Web Service QoS. The subject of aggregation and consensus of QoS is the hottest topic for researchers currently as well as the newest of the Web Service areas of research. As such although there are many groups working on the subject the availability of their work is still not readily available and mostly relate to mathematical algorithm approaches.

PART 2

2.1 System Requirements

2.1.1 Functional Requirements

The proposed monitoring tool has a series of functional requirements that it needs to fulfill. These are characteristics of the system that relate to operations and inner workings (eg. types of tests, automations etc)

It must be able to connect to any service to monitor its performance. In order to meet this objective, the application will query the service for its public methods and attempt to execute them, in order to take measurements of the quality metrics.

It should be able to monitor the quality metrics in real time in order to get realistic values of them. Additionally acquisition of such measurements should be achieved through passive monitoring or through active testing. These tests will be performed not only once, but there will be an iteration of the testing procedure so that the monitoring tool will be able to yield a firm result with respect to the service's performance.

The application should be able to derive more complex metrics that depend on measurable basic quality metrics.

The resulting metrics should be able to be utilized from web service discovery systems. Results should be machine readable so that other systems can incorporate them in their functionality.

2.1.2 Non-functional requirements

The non-functional requirements of the system describe those characteristics of the system that do not relate to specific operations, but rather involve somewhat subjective features, such as usability, reliability, etc.

The proposed tool should be expandable. Incorporating new metrics, tests and/or functions to it must be easily done, if needed.

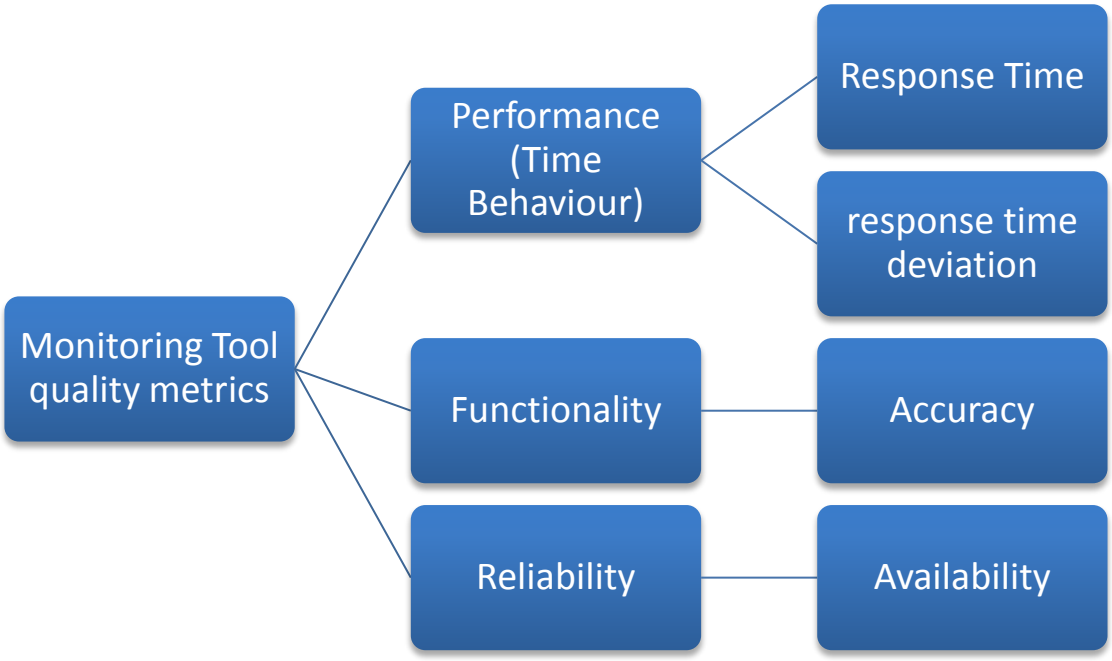
It should utilize existing standards, so that it will be possible to incorporate into other service frameworks. Thus standards such as WSDL and SOAP should be compliant with the monitoring tool.

It should be user-friendly. A graphical user interface (GUI) should be implemented, in a way that will facilitate all the user's tasks, while at the same time be easy to use. The user should be able to easily connect to the chosen web service, with appropriate error messages in case the web service cannot be tested (e.g. service is down or does not conform to the framework standards).

Test results should be presented in an understandable way so that the user can easily interpret them. Presented should be both in raw form (e.g. a table of the response times or of time errors) as well as in graphic form.

2.1.3 Metrics

For determining which quality metrics the monitoring tool will take measurements of, the ISO/IEC 25010 [4] model was considered. Since Operability, Security, Compatibility and Portability are not directly measurable, only the following were taken into account during the development of the tool.



(Figure 14) Quality attributes hierarchy used in the monitoring tool.

The characteristics and quality metrics used are explained below

- Time Behaviour
  - response time

    The total time that passes from the moment a request is made to the service until the moment a reply returns, represented in milliseconds.
  - Stability

    How stable and consistent is the service regarding response time.
- Accuracy

Refers to the capability of the service to provide correct results. This is hard to measure at all times, since it would require the expected response of a certain service request to be known.

- Availability
  This tells if the service is up and running and able to provide a reply.

2.2 Development information

Since the monitoring tool must be able to perform tests on any WCF service, this requires the first stage was to develop an interface that the service must implement in order to enable the monitor tool to access (and test) its methods properly. The interface will declare the methods that should be used in order to check for the acceptable response times of each method and for a certain validity check of the results. Once this interface has been implemented on the service, the monitoring tool will be able to connect to the service, execute its methods and test the responses' quality and timeliness against the standards set forth by the developer of the service. This means that the proposed system shows a great degree of usability on virtually any system, where the developer has allowed for monitoring under the principles set forth in the requirements. This significantly increases the potential of the system as a widely used monitoring tool.

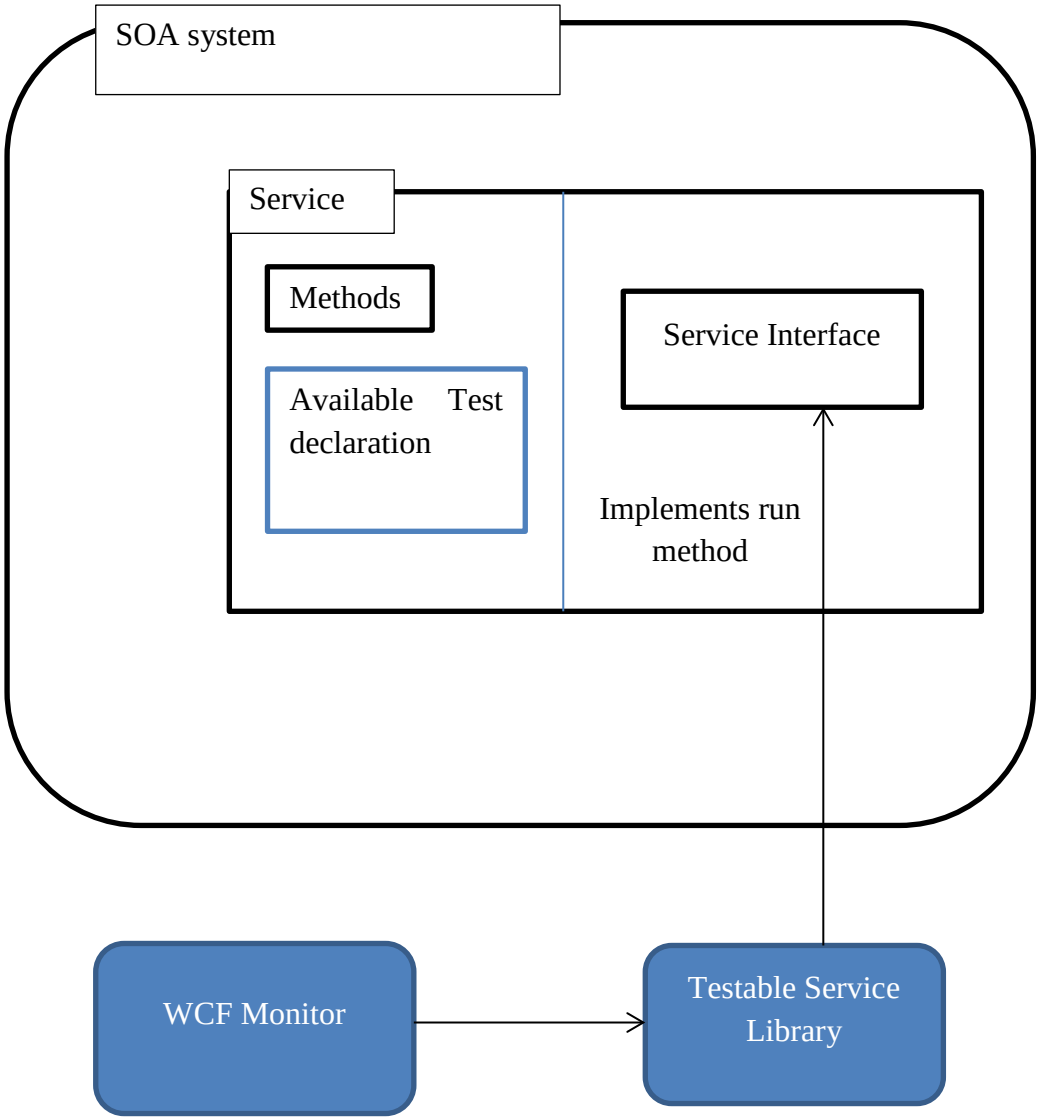
Furthermore, a sample WCF service is required in order to test for performance, with the monitor tool. Common netiquette forbids performing the aforementioned tests on someone else's service without permission. Such an action could well be considered by the service's administrators as a denial-of-service attack and would probably, in the best case scenario, result to our IP address being blocked by the server. Apart from ethical or legal issues, this would not allow us to perform the necessary tests and complete our work. Consequently, developing and hosting our own testable service is required, which will implement the principles set forth earlier, in order for it to be testable under our monitoring system.

The system was developed using Microsoft Visual Studio 2010 and the latest .NET Framework. Also the zedgraph library [62] was used for the graphical representation of measurements. The system used during development was a 64bit windows 7 system.

Testing was done by hosting the sample testable service on Microsoft Internet Information Services server version 7. The application conforms to the general design standards of today's applications regarding WSDL and SOAP coupled with WCF services.

2.3 The Proposed System Architecture

The proposed system consists of three parts, The *Testable Service Library* which is the basis of the entire system, *monitor service* part that allows the incorporation of the system to a service and the *monitor application* that connects all the other parts.



(Figure 15) Monitoring tool deployment in SOA system

2.3.1 The TestableServiceLibrary

The components of the TestableServiceLibrary, are the basis of the entire system. This interface class must be implemented in all services that allow testing under the proposed framework. The interface, described in the ITestableService class, requires that all testable WCF services implement two methods. First, the generic method RunMethod is implemented that allows the monitoring tool to run methods on the services, calling them by their name. This is needed since at design time the methods of the testable service are not always known. Second, a method called AvailableTests is implemented by the testable service. This will return an array of tests that can be run for that given service.

The testable service's developer is responsible for setting the service's tests and the expected performance according to the specifications of the service. The test standards may in the future be set by a reference document, but at this point, the tool can only test whether the behavior of the tested service meets the specifications of its developer.

Types of test are also described in the TestableServiceLibrary as can be seen in (Figure 16). A GenericTest class that includes the main characteristics common to all tests is declared which each specific test type extends by adding its individual characteristics and metrics necessary for the testing process. Of special note here is that the test implementation includes XML markup commands that will allow the results to be saved to an XML-type file.

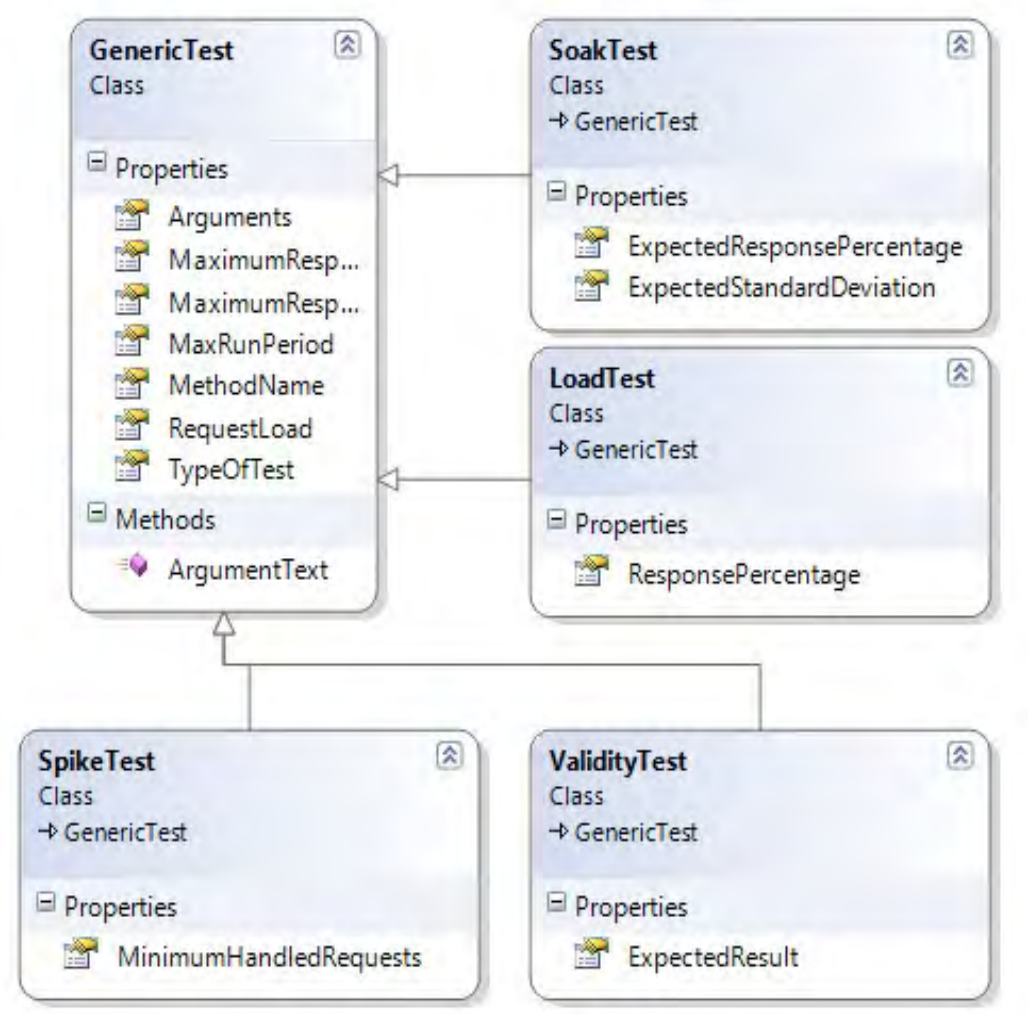
2.3.1.a Test types

The framework currently supports four different types of tests. Load, Soak, Spike and Validity test. In more detail the Load test measures whether a system is able to handle a typical number of requests per second. Of note is that the various methods that possibly exist in a service do not have the same amount of complexity. Thus the Load test has slightly different parameters for each method regarding response time threshold and the minimum required percentage of responses within the threshold.

Soak tests have a goal of testing a system performance for prolonged periods of time in order to find inconsistencies in operation and determine stability of the system. Given the relatively limited time available for this project soak test is performed for a limited period of time (e.g. a few hours) and test the system with a sustained load of requests. A check is made for how steady the response times are (small standard deviation.

Spike testing is meant to test a system against bursts of traffic, on small periods of time with intense traffic. Spike tests determine if a system is robust and can cope with heavy traffic without data loss or performance deterioration. Requests in a spike test must be a multitude of the typical load and the system should be able to handle a certain percentage of these requests in a small period of time (e.g. in a minute).

Validity tests are meant to check a system for correct responses and if it operates as expected during multiple requests.



(Figure 16) Class diagram of the available Tests

Finally in case of a service that does not implementing the proposed interface, only "free tests" are available. It is a simplified version of Load test that gets some parameters from the user and simply gathers the response times of the requests. This type of test can be applied on any exposed method of the tested service.

This capability was described in the functional requirements presented earlier.

Additionally, the code includes directives that allow the classes to be used in a WCF service environment.

2.3.1.b Any Service

Another important component is the AnyService class which is a generic service declaration containing a constructor. This instantiates the class using only the URL of a service and tries to parse the WSDL document to discover the methods exposed by that service and the arguments they require. The method is based on an XML-to-Linq technique [63], that given the XML scheme of a standard WSDL document, it queries the XML elements of the document.

For this project the focus is on WCF services. WCF tents to split the WSDL document into different files, while older web services (ASMX) had a single WSDL file. Considering that the goal is for the tool to be used in any web service it is compliant with both WSDL approaches, using the query command single WSDL. Similar to the format of the web services, the constructor method tries to parse the document in both formats. The constructor creates a stream to the WSDL document and runs queries against the document elements to discover the exposed methods and the arguments that these may require.

The class includes a property which is a Dictionary using the methods' names as key and another Dictionary as value. The value Dictionary includes the parameters of the method, using the parameter name as key and the parameter type as value, whereas if the method does not take any parameters, the value Dictionary is null.

2.3.1.c TestManager

The final component set is the system that performs the tests on the target service. The main class is the TestManager, which accepts the test to be executed and the service that the test is to be executed on. This class then starts a Timer and sets its interval according to the Requests per second value of the given test[1]. On each tick of the Timer object the method starts a new Thread and uses the Thread to execute the required method on the service and wait for its response.

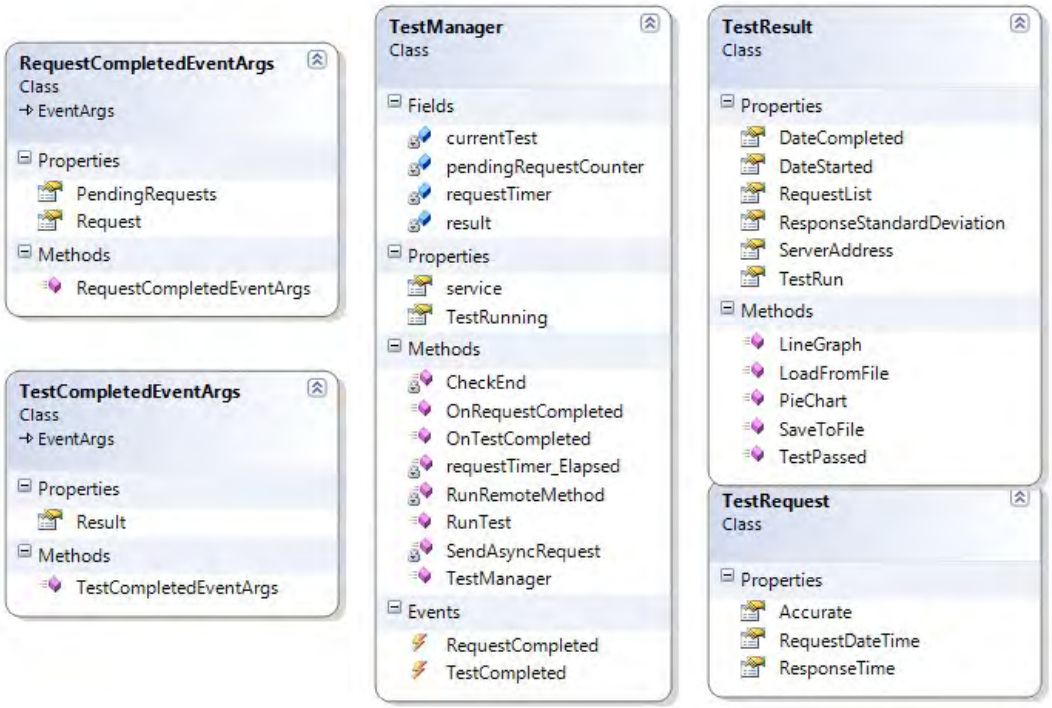
The information of this procedure is stored in a TestRequest object, along with the accuracy of the response, in the case of a validity test. The list of all TestRequest objects are stored in the XML-serialisable class TestResult, which can be saved to a

[1] The formula which sets the Interval in milliseconds is *1000 / RequestLoad*

TRF (Test Results File), which is an XML based fill. This option enables portability of the test results. A sample of a trf file can be seen below.

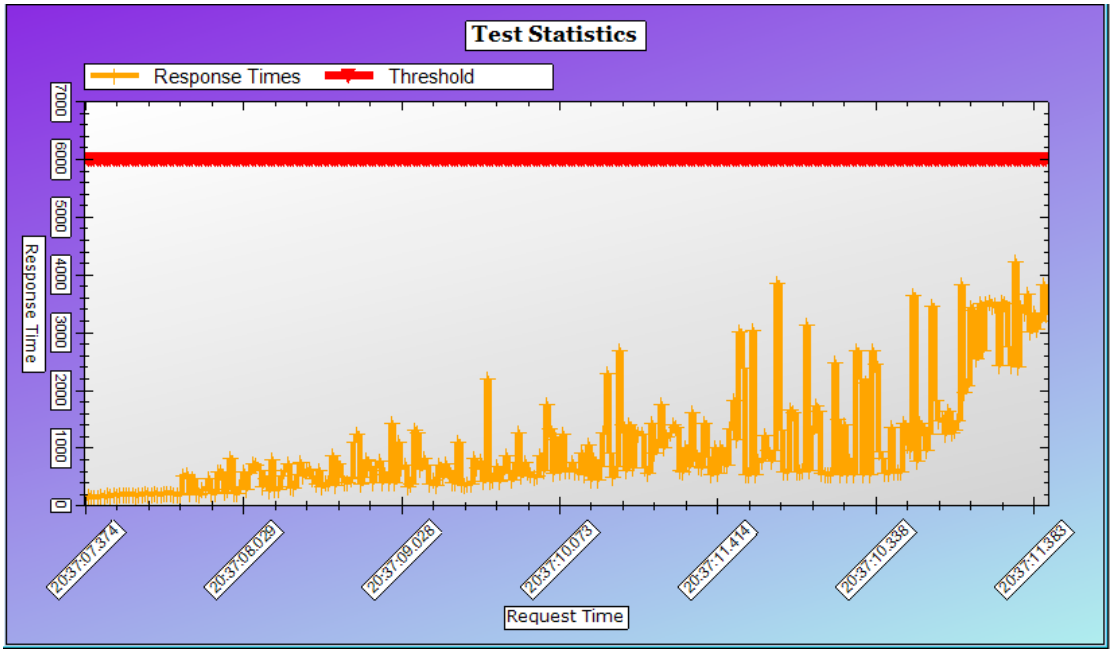
```
<?xml version="1.0" encoding="utf-8"?>
<TestResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <TestRun xsi:type="ValidityTest" TypeOfTest="ValidityTest" RequestLoad="1"
MaximumResponseTime="60000000" MethodName="PerformAddition">
  <MaxRunPeriod />
  <Arguments>
   <anyType xsi:type="xsd:int">1</anyType>
   <anyType xsi:type="xsd:int">2</anyType>
  </Arguments>
  <ExpectedResult xsi:type="xsd:int">3</ExpectedResult>
 </TestRun>
 <DateStarted>2014-06-01T20:24:05.6835885+03:00</DateStarted>
 <DateCompleted>2014-06-01T20:24:20.8944585+03:00</DateCompleted>
 <RequestList>
  <TestRequest ResponseTime="412.02360000000004" Accurate="true">
   <RequestDateTime>2014-06-01T20:24:06.7106472+03:00</RequestDateTime>
  </TestRequest>

      ⋮

  <TestRequest ResponseTime="172.00990000000002" Accurate="true">
   <RequestDateTime>2014-06-01T20:24:19.8804005+03:00</RequestDateTime>
  </TestRequest>
  <TestRequest ResponseTime="302.01730000000003" Accurate="true">
   <RequestDateTime>2014-06-01T20:24:20.8944585+03:00</RequestDateTime>
  </TestRequest>
 </RequestList>
</TestResult>
```

The TestRun element contains the information of the type of test in this case ValidityTest which is also the type of the test run also, and it contains the parameters set for the test as well as which method is tested (performAddition). Then the element arguments contains the input data for the method that is tested. Finally the resulted measurements for each request made to the tested method are presented each in its own element containing the metrics acquired.



(Figure 17) Classes of the Testable Service Library

The information in the test results can be seen in a graphical form (Figures 18, 22 and 23), with a line graph and a pie chart being used to display the data to the user. This option is consistent with the non-functional requirements presented in the requirements section. The graph option is implemented using the ZedGraph [62] library, which has become the standard for open source graphing of data series. Finally, the TestResult class uses a TestPassed method responsible for checking the metrics acquired and comparing with test goals to determine if the service performed in the required parameters set in the test. It returns true if the test results satisfy the conditions and also calculates a statistic as an output variable. This statistic is particular to the test type. For example, in the Soak test it is simply the standard deviation, while in the Load test, it is the percentage of requests handled with acceptable response times. Note that in the case of a Free test, the statistic is simply the average response time, which the user can interpret accordingly.

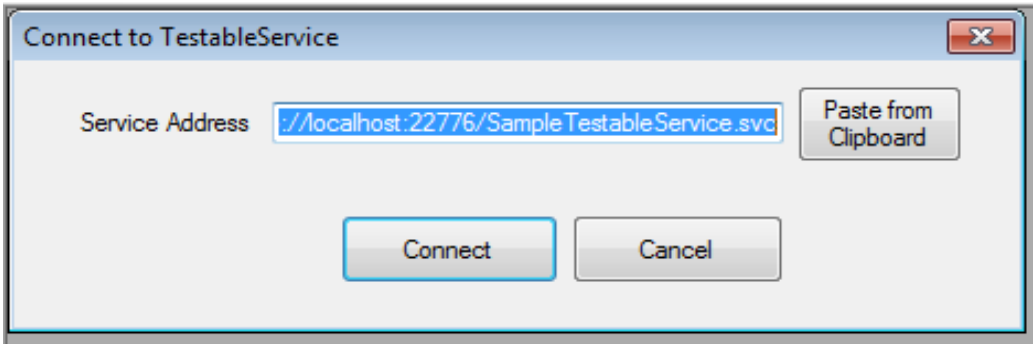


(Figure 18) Example of the graphical representation of a spike test results

2.3.2 The Testing Tool

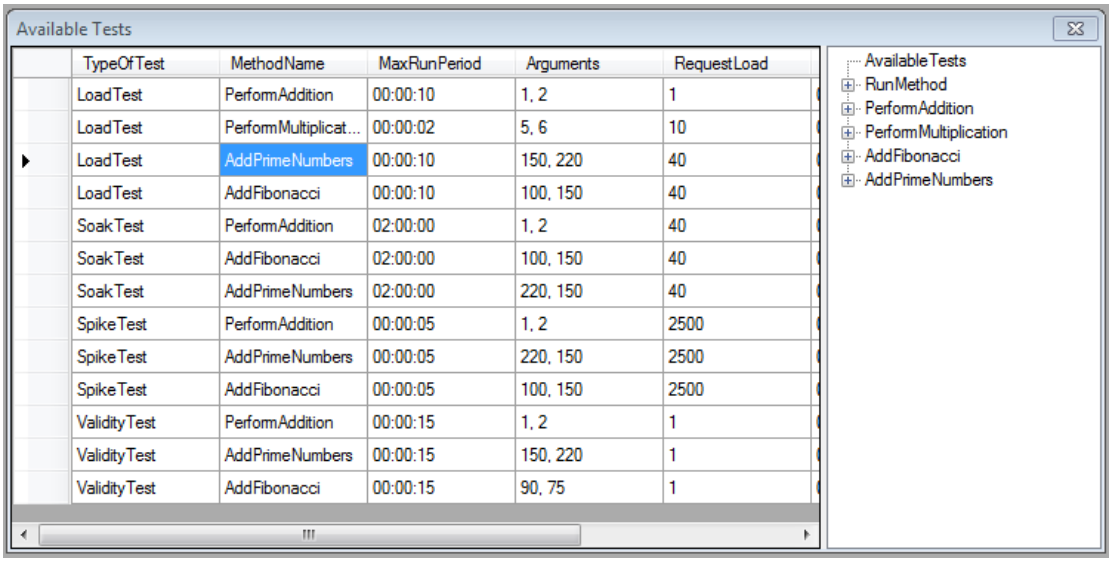
The monitoring tool developed, uses the TestableServiceLibrary to execute tests primarily to services that implement the ITestableService interface, but can additionally execute methods on any service that exposes its methods on a WSDL document and allows execution of its services by name.

The application uses a simple menu format to allow the user to execute its functions. On the Tools menu, in the connection form, the user can enter the URL of the service to be tested. This form defaults to http://localhost:22776/SampleTestableService.svc which is the default URL of the SampleTestableService assigned during development but an alternate service URL can be given. Provided the service in said URL complies with the previously mentioned WSDL standards, or implements the ITestableService interface a message informs the user for the successful connection. In case the service is not available an appropriate message appears informing the user about the failure. In case the service does not implement the framework interface, the user is informed that only free tests are available.



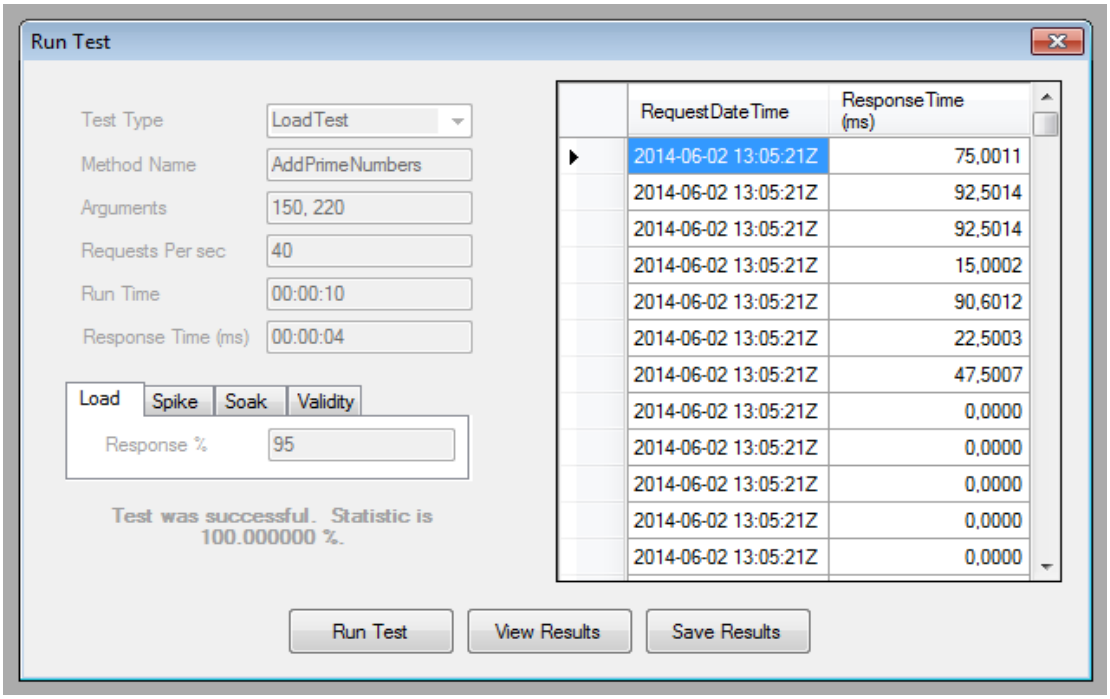
(Figure 19) Connection menu

Once the connection to a testable service is made, the user can use the Test Panel to view the tests the service allows. In case of a service implementing the Testable Service Library this window shows the tests that have been declared by the developer for the service along with their running parameters. On the right hand side all the available methods of the service are shown. These are drawn dynamically from the exposed WSDL of the service upon connection of the tool. In case of a service that does not implement our framework, only the methods are displayed and the user can perform a free test and declare the test parameters.



(Figure 20) List of available implemented tests in the Sample Service

Double clicking on any of the tests, opens the Run Test panel. Here the parameters of the selected test are shown on the left (or can be given by the user). Also the test goals are shown for the declared test (free tests do not yet implement setting threshold goals. Running the test will return an array of raw data of the results on the right side and a status notification about achieving or not the test goals that were set.



(Figure 21) Raw Results of a Load test

Additionally, the services methods are shown in a tree view, which the user can double click to start a Free Test on that method. If the test is of Free type, the user can enter the options for the request load (requests per sec) and the length of run time, as well as the methods parameters, if any, separated with a comma or a semicolon. If the test is a Validity Test, an additional column is displayed with the results, which displays True if the response of the method was the expected one, as registered by the test specifications.

Using the options that the zed graph library [62] offers, the Reporting form allows the user to display the results in a line diagram or a pie chart. If the test type includes some form of a threshold value, this is represented clearly on the line diagram, using a red line. Also the test results can be saved to an XML-type document of type TRF so that they can be possibly used by some decision or alerting system.

2.3.3 The Sample Service

Our sample service, SampleWCFService, is built to comply with the requirements of the ITestableService interface. It exposes a list of the available tests that the developer permits on the service and it implements the RunMethod method which the tests use to call the test methods. A sample of how a Soak test is declared in the service can be seen below.

```
GenericTest primeSoakTest = new SoakTest();
primeSoakTest.TypeOfTest = TestType.SoakTest;
primeSoakTest.MethodName = "AddPrimeNumbers";
primeSoakTest.MaxRunPeriod = new TimeSpan(2, 0, 0);
primeSoakTest.Arguments = new object[2] { 220, 150 };
primeSoakTest.RequestLoad = 40;
primeSoakTest.MaximumResponseTime = new TimeSpan(0, 0, 0, 0, 7);
((SoakTest)primeSoakTest).ExpectedStandardDeviation = 0.4;
((SoakTest)primeSoakTest).ExpectedResponsePercentage = 75;
retVal.Add(primeSoakTest);
```

We can see that the test is declared as a generic test of the subtype soak test. Then the method for which the declaration is made is declared followed by the values for the attributes of the generic test class. At the end the specific to the soak test parameters for the Expected standard deviation and Expected response percentage in the threshold are given values.

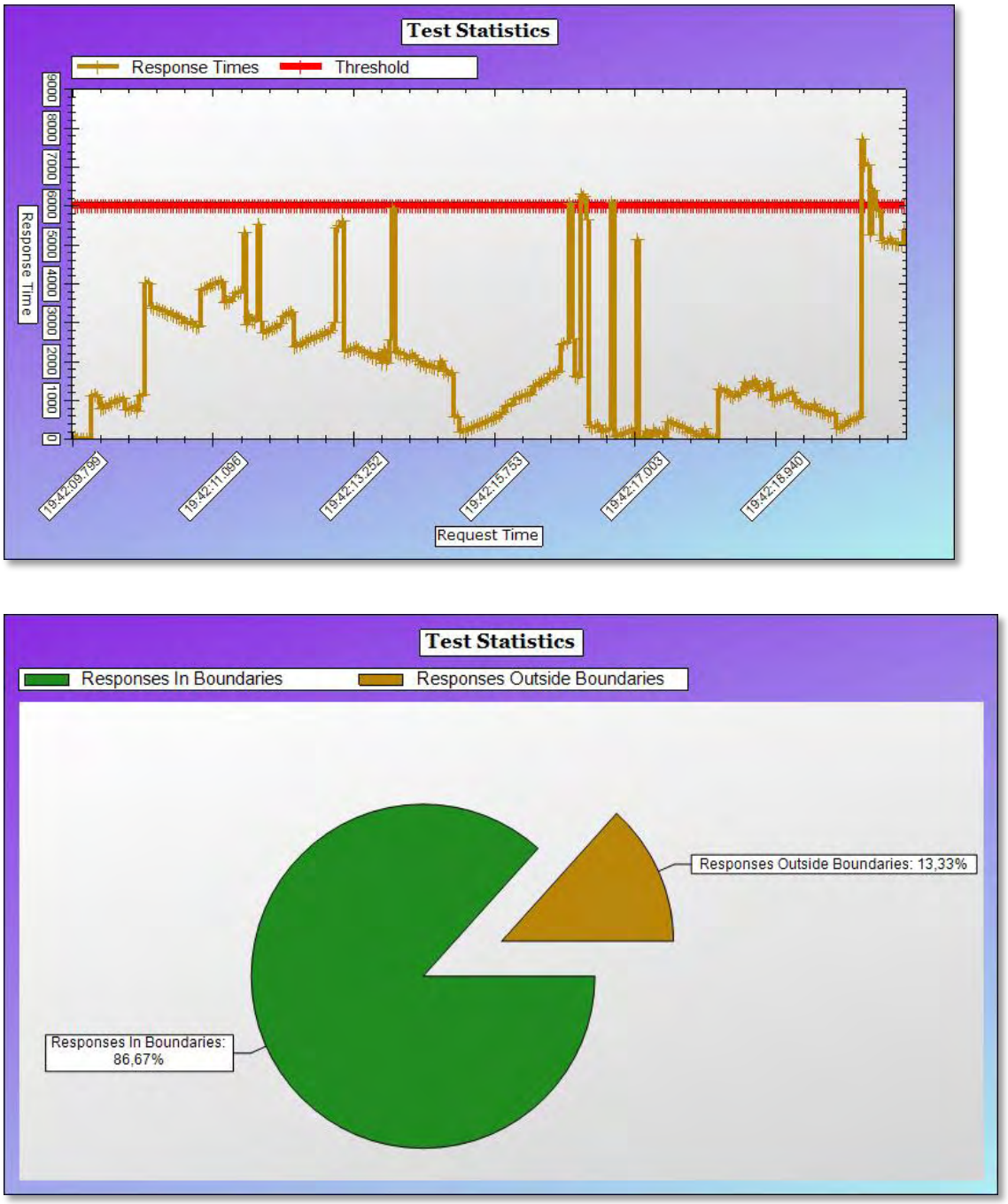
The service includes four methods, two simple ones the PerformAddition and the PerformMultiplication that take two parameters and as their names imply return the sum and the product of their parameters. Also there are two more complex methods that also take two parameters and the first AddFibonacci returns the sum of two n^{th} fibbonacci numbers and a similar method AddPrimeNumbers that returns the sum of two n^{th} prime numbers. In both cases the n is taken from the parameters of the methods.

For testing purposes, in PerformAddition and PerformMultiplication code that creates artificially a 40% chance of a five second overhead delay in the response and a 20% chance of producing a wrong response was included. These "faults" were needed for testing of the service during the developer server of the visual studio IDE, in order to manage the correctness of the test result analyzer. The code in the final version has been disabled but left in the file for future reference.

2.4 Testing the Service

For testing the framework first the SampleWCFService was hosted on the IIS server on a machine. Then the monitoring tool was run on machines connected to the server machine and a lan level, as well as through the internet.

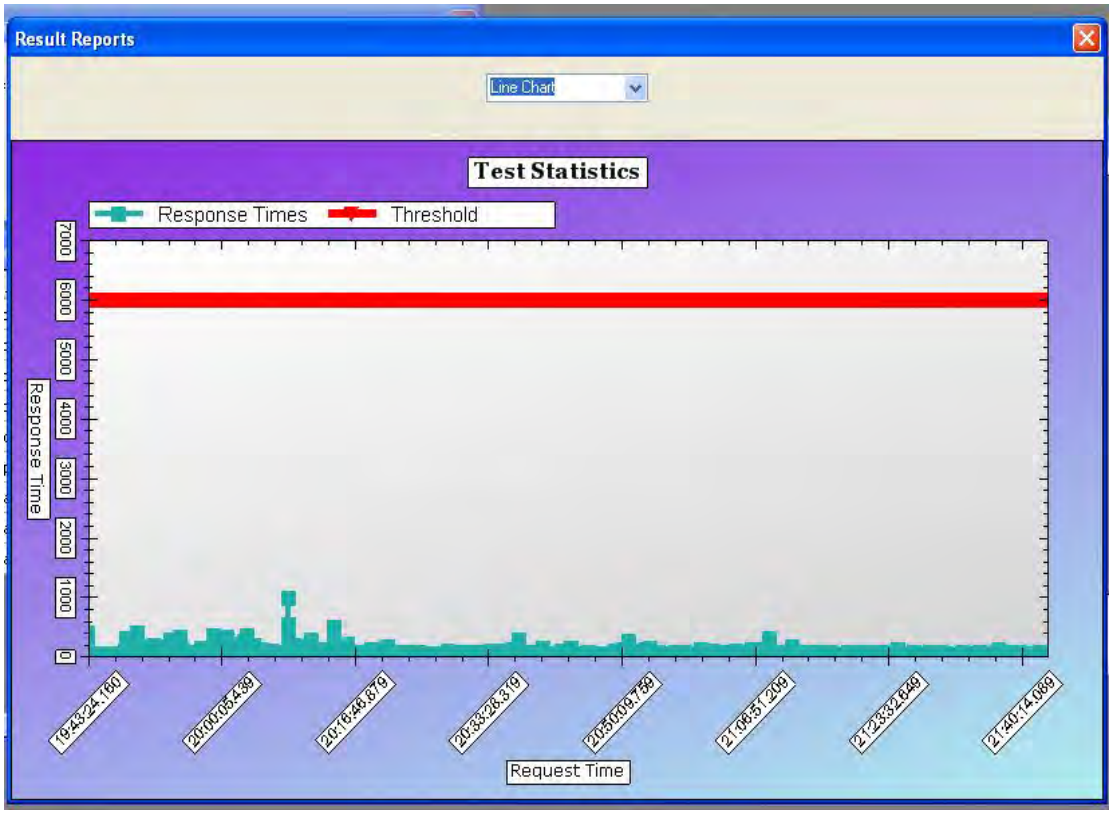
All the available tests of our sample service were then run for study. Interestingly enough, the overhead delay we implemented in the PerformAddition method caused an increased delay in the responses and had as a result the Load test failing quite often, even though the minimum response time set as requirement is as high as 6 seconds. That is 1 second higher than the maximum enforced delay from the artificial error and 3 seconds higher than the average expected delay.



(Figure 22) line graph and pie chart of load test

The Spike test is always successful, with a success statistic (percentage of handled requests), despite the extremely high number of the request load (2.500 requests per minute, i.e. more than 40 requests per second!). Interestingly enough it showed that the implemented service while able to run normally showed some increased delays towards the end of the test. This can be interpreted that in maintained extreme load the service performance degrades.

Another interesting point is the density of the line graph for the Soak test, which is due to the significant number of requests sent and handled in the given time period (2 hours). Graphical representations of the test results are shown below.



(Figure 23) Load test Line graph

2.5 Further research

In the future goals for the proposed tool is to enable it to work with other services that are not WCF schema compatible. To make it as such a dynamic WSDL recognition mechanism should be developed to incorporate any type of schema that it might encounter.

Another issue to be addressed is to have the test declarations outside the service code to enable greater portability and ease of incorporating the tool to a service system.

The adoption of some formal ontological language for representing the results of the QoS measurements as well as the SLA violation is something that can be expanded upon also.

References

[1] I. ISO, "ISO 9126/ISO, IEC (Hrsg.): International Standard ISO/IEC 9126: Information Technology-Software Product Evaluation," 1991.

[2] ISO/IEC 9126 Software Engineering - Product Quality - Parts 1-4, ISO, Geneva, 2001-2004.

[3] ISO/IEC 25000:2005 Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE, ISO, Geneva, 2005.

[4] ISO. ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models, by ISO/IEC, 2011.

[5] OASIS, "OASIS Web Services Quality Model TC Public Documents: http://www.oasisopen.org/committees/documents.php?wg_abbrev=wsqm," 2008.

[6] "Web Services Quality Factors Version 1.0". 31 October 2012, Candidate OASIS Standard 01, http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/cos01/WS-Quality-Factors-v1.0-cos01.html.

[7] IEEE Software 13(1), "Software quality: The elusive Target", Kitchenham S., Pfleeger, 1996

[8] ISO 9001:1994, International Standardization Organization, 1994

[9] H. Lausen, A. Polleres, D. Roman, "Web Service Modeling Ontology (WSMO)", W3C Member Submission 3, June 2005

[10] [MOF] The Object Management Group: Meta-Object Facility, version 1.4, 2002. Available at http://www.omg.org/technology/documents/formal/mof.htm.

[11] Jos de Bruijn, H. Lausen, "Web Service Modeling Language (WSML)", W3C Member Submission 3, June 2005

[12] "A formal model of the Semantic Web Service Ontology (WSMO)", Hai H. Wang , Nick Gibbins, Terry R. Payne, Domenico Redavid, Elsevier, Information Systems 37 (2012) 33–60

[13] R. Duke, G. Rose, Formal Object Oriented Specification using Object-Z, Cornerstones of Computing, Macmillan, 2000

[14] X.Wang, T. Vitvar, M. Kerrigan, I. Toma, "A QoS-aware selection model for semantic web services", Proceedings of the 4th international conference on Service-Oriented Computing, December 04-07, 2006, Chicago, IL [doi>10.1007/11948148_32]

[15] Mark H. Burstein , Jerry R. Hobbs , Ora Lassila , David Martin , Drew V. McDermott , Sheila A. McIlraith , Srini Narayanan , Massimo Paolucci , Terry R. Payne , Katia P. Sycara, "DAML-S: Web Service Description for the Semantic Web", Proceedings of the First International Semantic Web Conference on The Semantic Web, p.348-363, June 09-12, 2002

[16] C. Zhou, L.T. Chia, B.S. Lee, "DAML-QoS Ontology for Web Services", In: ICWS '04, Proceedings of the IEEE International Conference on Web Services 2004 (ICWS '04), Washington, DC, USA, IEEE Computer Society (2004) p472

[17] "OWL-S: Semantic Markup for Web Services", W3C Member Submission 22, November 2004

[18] "OWL Web Ontology Language Reference", W3C Recommendation 10, February 2004

[19] Kyriakos Kritikos, Dimitris Plexousakis, "OWL-Q for Semantic QoS-based Web Service Description and Discovery", SMRR, volume 243 of CEUR Workshop Proceedings, CEUR-WS.org, 2007

[20] G. Fortes Tondello , F. Siqueira, The QoS-MO ontology for semantic QoS modeling, Proceedings of the 2008 ACM symposium on Applied computing, March 16-20, 2008, Fortaleza, Ceara, Brazil

[21] Samer Hanna, Ali Alawneh, "An Approach of Web Service Quality Attributes Specification", IBIMA Publishing, Communications of the IBIMA http://www.ibimapublishing.com/journals/CIBIMA/cibima.html Vol. 2010 (2010), Article ID 552843, 13 pages DOI: 10.5171/2010.552843

[22] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[23] C. Lin, K. Kavi and S. Adepu. "A description language for QoS properties and a framework for service composition using QoS properties", Proceedings of the Seventh International Conference on Software Engineering Advances (ICSEA 2012), pp 90-97, Lisbon, Portugal, Nov. 18-23, 2012

[24] YunHee Kang, "Extended Model Design for Quality Factor Based Web Service Management," fgcn, vol. 2, pp.484-487, Future Generation Communication and Networking (FGCN 2007) - Volume 1, 2007.

[25] X. Liu, W. Gui, H. Wang, Y. Liu, J. Long, "A New Dynamic QoS Management Model Base on DQ-WSDL for Web Services", Journal of Information & Computational Science 10: 4 (2013) 1159-1167, Binary Information Press

[26] Jacek Kopecký , Tomas Vitvar , Carine Bournez , Joel Farrell, SAWSDL: Semantic Annotations for WSDL and XML Schema, IEEE Internet Computing, v.11 n.6, p.60-67, November 2007

[27] Joel Farrell, Holger Lausen, "Semantic Annotations for WSDL and XML Schema", W3C Recommendation 28 August 2007, http://www.w3.org/TR/sawsdl/

[28] Hobold, G.C. , Siqueira, F. , "Discovery of Semantic Web Services Compositions Based on SAWSDL Annotations", Web Services (ICWS), 2012 IEEE 19th International Conference 24-29 June 2012 Honolulu, HI, pages 280 – 287, ISBN: 978-1-4673-2131-0, IEEE

[29] M. Klusch, P. Kapahnke, "Semantic Web Service Selection with SAWSDL-MX", Proceedings of the 2nd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, Karlsruhe, Germany, CEUR-WS.org, 2008

[30] M. Klusch, P. Kapahnke and I. Zinnikus., "Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer.", The Semantic Web: Research and Applications:550-564, 2009.

[31] B. WU and X. WU, "A QoS-aware Method for Web Services Discovery," Journal of Geographic Information System, Vol. 2 No. 1, 2010, pp. 40-44

[31] Ravi Shankar Pandey, "A PLATFORM FOR SERVICE PROVIDER TO ADVERTISE QUALITY OF SERVICE USING WSDL", International Journal of Information Acquisition Vol 8. No 4 (2011) 291-302, World Scientific Publishing Company

[32] A. D'Ambrogio, "A Model-driven WSDL Extension for Describing the QoS of Web Services" In IEEE International Conference on Web Services (ICWS'06) (2006), pp. 789-796

[33] Z. Min, T. Xiao-yan, Li Jian-xin, C. Xiao-li, "Research on Service Discovery Based on QoS in the Composition of Web Services", I.J. Engineering and Manufacturing, 2012,2, 58-64 http://www.mecs-press.net

[34] N, Parimala, and A, Saini, "Web Service with Criteria: Extending WSDL, " in Proc, of the 6th Int, Conf, on Digital Information Management, 2011, pp, 205-210

[35] OASIS, "Web Services Quality Description Language v1.0 Working Draft", 10 October 2008

[36] OASIS, WSQDL schema https://www.oasis-open.org/committees/download.php/47780/WSQDL.xsd

[37] T.Rajendran, P.Balasubramanie, R. Cherian, "An Efficient WS-QoS Broker Based Architecture for Web Services Selection", 2010 International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 9

[38] Blum, Representing Web Services Management Information in UDDI CTO, Systinet Corporation & Fred Carter, Sr. Architect, AmberPoint, Inc. January 2004

[39] OASIS, "Representing Web Services Quality of Service Information in UDDI", https://www.oasis-open.org/committees/download.php/6227/uddi-spec-tc-tn-QoS-metrics-20040224.doc, 2004

[40] O. Martin-Diaz, A. Ruiz-Cortez, R. Corchuelo, M.Toro, "A Framework for Classifying and Comparing Web Services Procurement Platforms", Procedings of 1st International Web Services Quality Workshop, Italy, pp. 37-46, 2003

[41] N. Parimala, Anu Saini, "X-WSDL: An Extension to WSDL and its mapping to X-UDDI", International Journal of Information Studies, Volume 3 Number 3, July 2011, 115-127

[42] N. Parimala, Anu Saini, "Decision Support Web Service", ICDCIT 2011, LNCS 6536, p. 221-231, Springer

[43] A. Eleyan and L. Zhao, "Extending WSDL and UDDI with Quality Service Selection Criteria" Proceedings of the 3rd International Symposium on Web Services: 3rd International Symposium on Web Services; 21 Apr 2010-22 Apr 2010; Dubai. 2010.

[44] V. Diamadopoulou, C. Makris, Y. Panagis, E. Sakkopoulos, "Techniques to support Web Service selection and consumption with QoS characteristics", Journal of Network and Computer Applications 31 (2008) 108–130, Elsevier

[45] Rathore, M. and U. Suman, "A quality of service broker based process model for dynamic web service composition", J. Comput. Sci., 7: 1267-1274. 2011

[46] Tao Yu, Kwei-Jay Lin, "Service Selection Algorithms for Web Services with End-to-End QoS Constraints", Proceeding CEC '04 Proceedings of the IEEE International Conference on E-Commerce Technology, Pages 129-136

[47] H. Chen, P. Mohapatra, "Using Service Brokers for Accessing Backend Servers for Web Applications ", Proceeding ICDCSW '03 Proceedings of the 23rd International Conference on Distributed Computing Systems, Page 928

[48] G. Vadivelou, E. IIavarasan, R. Manoharan, P. Praveen, "A QoS Based Web Service Selection Through Delegation", International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011, ISSN 2229-5518

[49] M. Alrifai, T. Risse, P. Dolog, W. Nejdl, "A Scalable Approach for QoS-Based Web Service Selection", Service-Oriented Computing --- ICSOC 2008 Workshops, Pages 190 – 199 Springer-Verlag Berlin, Heidelberg ©2009

[50] Serhani, M. Adel. & Benharref, A, "Enforcing quality of service within web services communities", Journal of Software, vol. 6, no. 4, pp. 554-563, 2011

[51] M.F. Arlitt, N. Carlsson, C. Williamson, J. Rolia, "Passive crowd-based monitoring of World Wide Web infrastructure and its performance", Proceedings of ICC. 2012, 2689-2694.

[52] Y. Fuy, L. Cherkasovaz, W. Tangz, A. Vahdat , "EtE: Passive End-to-End Internet Service Performance Monitoring", ATEC '02 Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference, Pages 115 – 130

[53] R. Jurca, B. Faltings, W. Binder, "Reliable QoS monitoring based on client feedback," Proceedings of the 16th International Conference on World Wide Web, 2007, 1003-1012.

[54] C. Zeginis, D. Plexousakis, "Monitoring the QoS of Web Services using SLAs", Technical Report 404, ICS-FORTH, May 2010

[55] D. Chappel, "Enterprise Service Bus", O'Reilly Media, Inc. ©2004 ISBN:0596006756

[56] A. Michlmayr, F. Rosenberg, P. Leitner, S. Dustdar, "Comprehensive QoS Monitoring of Web Services and Event-Based SLA Violation Detection", MW4SOC '09, November 30, 2009, Urbana Champaign, Illinois, USA, Copyright 2009 ACM

[57] L.Zeng, H. Lei, H. Chang, "Monitoring the QoS for Web Services" Lecture notes in Computer Science, vol 4749, 132, 2007

[58] S. Bragaglia, F. Chesani, E. Fry, P. Mello, "Event condition expectation (ECE-) rules for monitoring observable systems", RuleML'11 Proceedings of the 5th international conference on Rule-based modeling and computing on the semantic web, Pages 267-281

[59] OASIS, "Web Services Dynamic Discovery (WS-Discovery) Version 1.1", OASIS Standard 1 July 2009 http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

[60] GT4.2.1:Information Services [DB/OL], http://toolkit.globus.org/toolkit/downloads/4.2.1/

[61] W. L. Lin, C. C. Lo, K. M. Chao, N. Godwin, "Multi-group QoS consensus for web services", Journal of Computer and System Sciences 77 (2011) 223–243

[62] http://zedgraph.sourceforge.net/index.html

[64] StackOverflow (2010). "How to retrieve web methods' names from a wsdl document?" http://stackoverflow.com/questions/3821262/linq-2-xml-how-to-retrieve-web-methods-names-from-a-wsdl-document

APPENDIX