Evaluation Framework for Quality of Service in Web Services: implementation in a pervasive environment

Master Thesis

Master Research in Informatics

Specialized in Technologies of Information & Web

Enrique Lafuente Hernandez
Followed by:
M.C. Frédérique LAFOREST
Claudia Catalina GUTIERREZ RODRIGUEZ
# Index

List of Figures  ........................................................................................................................................iv
List of Tables ...........................................................................................................................................iv
Abstract ..................................................................................................................................................v
Résumé ....................................................................................................................................................vi

1. Introduction & Objectives ...................................................................................................................1
   1.1. Background ......................................................................................................................................1
   1.2. Problem description .....................................................................................................................1
   1.3. Objectives ......................................................................................................................................1

2. State of the art ....................................................................................................................................2
   2.1. Services and Web Services ............................................................................................................3
       2.1.1. Non-functional properties (NFP) ..........................................................................................3
       2.1.2. Non-functional properties classifications .........................................................................4
   2.2. Aggregation of non-functional properties ..................................................................................6
       2.2.1. Aggregation of non-functional properties in a Single Service ...........................................6
       2.2.2. Aggregation of non-functional properties in a Service Composition ...............................6
   2.3. Quality of Service in web services ..............................................................................................7
       2.3.1. Model for Quality of Service ..............................................................................................7
   2.4. Service-Oriented Architecture and QoS .....................................................................................8
       2.4.1. QoS Measures ....................................................................................................................8
       2.4.2. QoS Publication ..................................................................................................................9
       2.4.3. Service Selection according to QoS ..................................................................................9
   2.5. Service Level Agreement in Web Services ...............................................................................10
   2.6. Conclusion ....................................................................................................................................10

3. Quality of Service Framework for Web Service ..............................................................................11
   3.1. Preliminary: Service selection by functional properties ............................................................12
   3.2. Non-functional properties of Web Services and Quality of Service ........................................12
       3.3. Non-functional Properties .......................................................................................................13
           3.3.1. Non-functional properties classification ..........................................................................13
           3.3.2. Web Service Host ..........................................................................................................14
           3.3.3. Getting Non-Functional Properties Values ...................................................................14
           3.3.4. Normalization function ..................................................................................................15
       3.4. Quality of Service Items ..........................................................................................................16
   3.5. Quality of service grading ..........................................................................................................18
   3.6. Synthesis of Quality of Service Calculation ..............................................................................18
   3.7. Towards Quality of Service evaluation of composition services .............................................19
       3.7.1. Composition of Non-functional Properties Values .............................................................20
       3.7.2. Composition of Quality of Service Model .........................................................................20
   3.8. Architecture for the evaluation of Quality of Service .................................................................21
       3.8.1. Non-functional properties measurements .........................................................................21
       3.8.2. Non-functional properties publication ..............................................................................22
       3.8.3. Quality of Service calculation ............................................................................................22
4. OPTIMACS Specifications ................................................................. 23
   4.1. Non-functional properties identified in OPTIMACS ......................... 23
   4.2. Architecture of the OPTIMACS Testbed and implementation of QoS in this Architecture 23
5. Evaluation & Results ........................................................................ 26
6. Conclusion ......................................................................................... 29
7. Annexes ............................................................................................. 30
   7.1. Normalization Function .................................................................. 30
   7.2. Quality Model in Web Service Composition .................................... 33
   7.3. Sequence diagrams implemented in the architecture ....................... 34
      7.3.1. Publish a service ................................................................. 34
      7.3.2. Measure of non-functional Properties .................................... 34
      7.3.3. Publish non-functional Properties ......................................... 35
      7.3.4. Calculate the Best Service .................................................... 36
   7.4. Histograms of the implementation tests ......................................... 37
      7.4.1. Histograms of the response time depending on the number of non-functional properties 37
      7.4.2. Histograms of the response time depending on the number of services .......... 37
8. References .......................................................................................... 39
   8.1. Biography .................................................................................... 39
List of Figures

Fig. 1: Possible classifications of non-functional properties ........................................ 4
Fig. 2: A Semantic Link between two services (11) .................................................... 6
Fig. 3: Example of composition chart for QoS compositions (10) .................................. 6
Fig. 4: Architecture of QoS measurement, publication & calculation ............................ 8
Fig. 5: Invasive VS non-invasive measurements of QoS ............................................ 8
Fig. 6: General View of all the steps to select the best service .................................... 11
Fig. 7: Representation of the belonging of the non-functional properties to a quality item ... 16
Fig. 8: QoS Model including all the Quality Items concerned in a service evaluation ........ 17
Fig. 9: Web Service Composition ............................................................................. 19
Fig. 10: Generic Architecture for evaluating Quality of Service ................................. 21
Fig. 11: Current Architecture in the Project OPTIMACS ........................................... 24
Fig. 12: QoS-aware OPTIMACS architecture .......................................................... 24
Fig. 13: Statistical values in the response time depending on the number of NFP .......... 27
Fig. 14: Statistical values in the response time depending on the number of services ....... 28
Fig. 15: Normalization result of non-functional properties distribution ..................... 30
Fig. 16: Normalization problems consequence of extreme values and non-normal distributions 31
Fig. 17: Normalization function for negative non-functional properties ..................... 31
Fig. 18: Normalization function for positive non-functional properties ..................... 32
Fig. 19: Registration by the Provider in a push mode .............................................. 34
Fig. 20: Service Discovery in a pull mode done by the Web Directory ......................... 34
Fig. 21: Invocation of a Service function for a client ............................................. 35
Fig. 22: Self-Test of the Services from the Provider ............................................... 35
Fig. 23: Test done by the Web Directory, more precisely by the QoS Monitoring .......... 35
Fig. 24: Update Service registration after a modification or because of a timer ........... 36
Fig. 25: QoS Monitoring requesting a Service Update ............................................. 36
Fig. 26: Complete sequence of retrieving the best functional service ....................... 36
Fig. 27: Histogram of the response time depending on the number of NFP ................. 37
Fig. 28: Histogram of the response time depending on the number of services .......... 38

List of Tables

Table 1: Non-functional properties in the literature ..................................................... 3
Table 2: Definition set of relevant Non-functional properties ..................................... 4
Table 3: Measures Depending on the Service System .............................................. 15
Table 4: Aggregation of non-functional values in a composition .............................. 20
Table 5: Update frequency depending on service host and the service demand .......... 22
Table 6: Non-functional Properties in the OPTIMACS specifications ....................... 23
Abstract

Nowadays in the new technology era, the services that people and companies need cross the Internet network. For example, reserving a fly, organizing a trip or even shopping is more and more done through a web browser. Companies also expand their market by offering their products or services through the Internet. To better organize this huge market, we need standards that allow both clients and providers to communicate and that help the client retrieving the service it searches for.

Different protocols and mechanisms have been defined to access services available on the web. In consequence these services are nowadays accessible by the final clients. In the last few years, more and more Web Services providing the same functionalities have been made available. In this context, one question arises: how to select the best service for a client request when several services with the required functionality are available? In order to answer this question and to satisfy client requirements, a detailed model for Quality of Web Services (QoS) has to be defined. The main objectives of this model are to discover and to propose the web service that is the most adapted for the specific needs of the current client.

In this Master thesis, we have designed a framework for the evaluation of the Quality of Service of Web Services. This framework is accompanied by an implementation in a pervasive environment.

In this document, we first of all better define the problem of QoS evaluation for web services and synthesize the existing works on this subject. We also leverage the importance of Quality evaluation and the lack of a general model exploitable in a heterogeneous environment. We have studied the architectures used nowadays to evaluate this QoS, extracting all the positive and negative aspects of each one.

Ensuring Quality of Service is the capability to respond to the requirements (constraints) of a client and to fulfil these needs with the best criteria (preferences) established by the client. It is calculated based on the non-functional properties of the services.

Then we have defined a QoS model to grade the services and then to make a ranking of functionally-equivalent services. This model contains all the possible aspects of a service evaluation. These aspects, called Quality Items, are calculated as an aggregation of the non functional properties values (NFP values) of the service. Non functional properties depend on the context, in order to adapt the evaluation to the service context. The whole process of grading the service is explained.

The main contributions of this work are to be able to evaluate the quality of service of a service in the following situations:

- using an evaluation procedure that is independent of the context (wide range of services)
- supporting lack of information (some non-functional properties without a value)
- taking care of the client needs so as to be the most adapted to the client. Three points of customization are provided to the client.
Résumé

Actuellement dans la nouvelle ère technologique, les services dont les gens et les entreprises ont besoin sont reparties dans le réseau d’internet. Par exemple, réserver un avion, organiser un séjour ou même faire les achats se font de plus en plus à travers d’un navigateur web. Les entreprises sont aussi en train d’élargir leur marché en offrant leurs produits à travers d’internet. Pour une cohérence dans cette marche en expansion, nous avons besoin des standards qui permettent la communication entre les entreprises et les clients, en plus il doit aider au client à trouver le service recherché.

Plusieurs protocoles et mécanisme ont été définis pour accéder aux services disponibles sur le net. En conséquence ces services sont aujourd’hui accessibles pour un client. Dans les dernières années, chaque fois plus de services web proposent la même fonctionnalité. Dans ce contexte une nouvelle question se pose: comment sélectionner le meilleur service pour une demande du client, quand plusieurs services proposant cette fonctionnalité sont disponibles ? Pour répondre à cette question et satisfaire les demandes du client, un modèle de Qualité de Service (QdS) doit être établi. Les objectifs principaux de ces modèles sont découvrir et proposer le service web qui est le mieux adapté aux besoins spécifiques du client.

Dans ce thèse de Master, j’ai propose un Framework pour l’évaluation de la Qualité de Service dans les Services Web. Ce Framework est accompagné par une implémentation dans un environnement pervasif.

Dans ce document, j’ai d’abord définie la problématique de l’évaluation de la QdS dans les services web et synthétisé les travaux existant dans ce domaine. J’ai aussi remarqué l’importance de l’évaluation de la QdS est la manque de travaux générales exploitable dans de domaines hétérogènes. J’ai, en suite, étudié les architectures utilisés actuellement pour l’évaluation de cette QdS, en prenant tous les aspects positifs et négatifs pour chaque une.

Assurer une Qualité de Service est la capacité d’un service de répondre aux besoins du client (contraintes) et de réaliser sa fonctionnalité avec les propriétés le plus adapté au client (préférences). La valeur de QdS est calculée en base des propriétés non-fonctionnelles du service.


La contribution la plus importante de ce travail est la capacité d’évaluer la qualité de service dans les situations suivantes :

- utilisation d’une procédure d’évaluation indépendante du contexte (service hôtes hétérogènes).
- supporter le manque d’information (des propriétés non-fonctionnelles pas renseignées).
- tenir en compte les besoins est les préférences du client pour trouver le service le plus adapté au client. Trois points de customisation sont proposés au client.

Quality of Services in pervasive environments
Master thesis
1. Introduction & Objectives

Nowadays in the new technology era, the services that people and companies need cross the Internet network. For example, reserving a flight, organizing a trip or even shopping is more and more done through a web browser. Companies also expand their market by offering their products or services through the Internet. To better organize this huge market, we need standards that allow both clients and providers to communicate and that help the client retrieving the service it searches for.

But more concretely we experiment the reaction of these communication methods in a pervasive environment. A pervasive environment (also called ubiquitous environment) is a post-desktop model of human-computer interaction. The characteristics of this environment are:

- Integrated everyday objects and activities in the computing.
- Engages many transparent computational devices.
- High dynamic user contexts, constantly changing network, not possible central control.
- Complex interaction between people, intelligent objects and computers.

1.1. Background

This research is done in the context of the national research project called OPTIMCAS. OPTIMCAS combines hybrid query processing and services composition, addressing services composition and query processing including adaptive hybrid query optimization according to QoS criteria. Service based querying on dynamic environments is validated through a testbed proposed in the project.

OPTIMACS addresses novel challenges on data/services querying that go beyond existing results for efficiently exploiting data stemming from many different sources in dynamic and multi-scale environments. Coupling together services, data and stream with query processing considering dynamic environments and QoS issues is an important challenge in the database community that is partially addressed by some groups.

1.2. Problem description

In this context of web services and even more in pervasive environment, we need to establish a way to evaluate the services. This evaluation is called Quality of Services. More concretely it helps to rank the services in order to select the one which responds the best to the client needs among all the services responding to the client functional demands. But the solution that we propose will focus in establishing this quality in such a general way that it will be valid to evaluate a pervasive environment and even compare it with other services that are not in the environment context.

The evaluation of QoS is one of the main points to retrieve the best performance and client satisfaction in services execution. Nowadays the concept of Quality of Service remains flue and the definitions depend strongly on the research context and are not adapted to other contexts. Then these solutions are not usable in a heterogeneous context where we need to compare services coming from different context. Each service is evaluated depending on its context but then we need a way to unify the different evaluations and rank the different services.

1.3. Objectives

The objectives of this project are:

- Study the research works to better understand the needs and the work already done.
- Define Quality of Service for web services.
- Describe a QoS Model to evaluate the quality of web services.
- Propose an architecture to measure, publish and calculate QoS.
2. State of the art

Web Services are software functionalities publish and accessible through the internet. Different protocols and web mechanism have been defined to access these Services. In consequence these services are nowadays accessible to the final clients. In the last few years, more and more Web Services providing the same functionalities are available in the environment.

Now, in this context, one question arises: how to select the best service for a client request when several services with the same functionality are available? In order to answer this question and to satisfy client requirements a detailed QoS model has to be defined. The main objectives of this model are to discover and propose the client the most adapted web service for the specifics needs if each client.

In order to discover a service, we need a full description of a service, containing the functional properties of the service but also other information typically called non-functional properties. These non-functional properties help to adapt the system by selecting the most appropriate service regarding QoS preference of the client.

In this section we review related research and evaluate their support for the selection of the best service among functionally equivalent services. This section is organized as follows: In section 2.1 we present a brief definition of a service and the characteristics of web services. Section 2.1.1 describes the main concepts of quality and quality model. Next, we introduce the non-functional properties and we explain how they are used to calculate the quality of a service. We propose a classification of these properties in Section 2.1.2. In Section 2.2, we describe some principles about services aggregation and composition. Section 2.6 is oriented to describe the typical architectures proposed to implement Quality of Service assessment and publication. We present in section 2.5 a view of QoS in a more continuous aspect with the Service Level Agreements. We conclude in section 2.6.
2.1. Services and Web Services

Nowadays, many definitions of the concept of service are already proposed in different application domains as Internet, business or telecommunications (1)(2)(3). However, in a generic way, a service could be regarded as “a function performed on your behalf at a cost”, where the cost is not only monetary, it concerns a whole collection of limitations (4). In fact, a service represents more than a function; it can be regarded as an interaction between a client and a provider, or as a service contained within other services such an aggregation or a composition. An aggregation combines multiple services and provides access to them in a single location, while a composition is an integration of services that results in values not present within the individual services.

As literature explains in this context, a service is described by its functional and non-functional properties (4). The functional properties describe what a service does (e.g. move a person between point X and point Y, calculate A plus B…), and the non-functional properties describe how the service executes some functionality (e.g. time, price…). As (4) states, a service is not completely described if it does not provide both descriptions. Actually, an accurate description of a service benefits in services discovery, substitution, composition and management.

More specifically, Web Services are considerable important as an approach for the integration of distributed, heterogeneous and autonomous applications (5). A web service can be considered as an application accessible from the network and it is executed on a remote system hosting the service. The power of such services is the possibility to create new services from other services. The web services can be atomic (one functionality) or composite (aggregated). These compositions can be of atomic services or other compositions [12]. Web Services are also defined by their functional and non-functional properties (6). In this case, the functional properties are defined using for example a Web Services Description Language (WSDL) in the Universal Description, Discovery and Integration (UDDI) (4). For the non-functional properties different definitions and classifications exist. We describe them in the following.

2.1.1. Non-functional properties (NFP)

According to the literature, an essential aspect of service representation is the capturing of the non-functional properties of services (4)(6). This includes methods, requests and provision of services, constraints of availability, service quality, security, trust and the rights attached to a service (7). These properties are not only comprehensive descriptions for useful service discovery; they are also essential for service management, enabling service negotiation, composition and substitution (4).

As (4) proposes, the non-functional properties in general services could be: Availability, Channel, Charging Styles, Settlement, Payment Obligations, Service Quality, Security and Trust, Ownership and Rights. These properties should be adapted to each particular case. For example, the spatial availability represents a really important property for a taxi cab service, opposite to a plane booking service. Generally, in web services, the location where the service is executed is not important as soon as it is executed under the client’ constraints (time, accuracy …).

One can find different points of view and different definitions for these non-functional properties. Table 1 gives an extensive list of the non-functional properties described in the recent literature (4) (6) (8) (9). These properties allow the description of the characteristics of web services.

<table>
<thead>
<tr>
<th>Non-functional properties in web services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
</tr>
<tr>
<td>Accessibilty</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>Audittrail</td>
</tr>
<tr>
<td>Authentication</td>
</tr>
<tr>
<td>Authorization</td>
</tr>
<tr>
<td>Best Practices</td>
</tr>
<tr>
<td>Capacity</td>
</tr>
</tbody>
</table>

Table 1: Non-functional properties in the literature
In table 2, we introduce a set of non-functional properties that are relevant to our research.

<table>
<thead>
<tr>
<th>Non-functional property</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>Quality of a service to be in a temporal and spatial context ready to deliver the demanded service. It can be calculated as $\frac{(1 - \text{Down Time})}{\text{Measurement Time}}$. The Measurement Time can be adapted to the application.</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Error rate that produces a service. It is calculated as the number of errors divided by the total number of executions.</td>
</tr>
<tr>
<td>Execution time</td>
<td>Measures precisely the time the provider takes to execute a request.</td>
</tr>
<tr>
<td>Penalty</td>
<td>Economical (or other) retributions that the provider has to give to the client in case of not providing the requirements.</td>
</tr>
<tr>
<td>Price</td>
<td>Amount of money the client has to pay for executing the request of the service provider. The provider can propose a fixed price or a price depending on values of the non-functional properties proposed by the service.</td>
</tr>
<tr>
<td>Reliability</td>
<td>Probability with which the provider correctly answers a request within a maximum expected time. It is measured as the number of success request divided by the number of request.</td>
</tr>
<tr>
<td>Reputation</td>
<td>Measure from the client or virtual clients of the experience they had using the service. The value of the reputation is the average of reputation of this service shared by all the clients, i.e., sum of the reputations divided by the number of clients. Normally reputations are defined within a range of values.</td>
</tr>
<tr>
<td>Robustness</td>
<td>Establishes the capacity of the system to give an accurate answer in presence of invalid, incomplete or conflicting inputs. These can be estimated at the provider side or tested and making an average from the last experiences.</td>
</tr>
<tr>
<td>Round Trip Time</td>
<td>Measures the complete time spent between a request is send and the response is received, it can be measured with a clock or calculated like Time of transport (depending strongly on the network) plus Time of Execution (containing the wrapping/de-wrapping time). Providers can publish their execution time. The RTT has to be calculated by an external component or client. The final value is the mean of all the past calculations.</td>
</tr>
<tr>
<td>Security</td>
<td>Comprises other non-functional properties (i.e. data encryption, authentication…). It evaluates the security in the transaction who assures the ACID properties (Atomicity, Consistency, Isolation and Durability). This value can be presented by the provider separately or as an aggregation of the different qualities.</td>
</tr>
<tr>
<td>Throughput</td>
<td>Measures the maximum network capacity the client and the provider can get to communicate. It is time dependent, so if a quality reservation is not possible on the network, the throughput would be either the maximum throughput registered or the average.</td>
</tr>
</tbody>
</table>

Table 2 : Definition set of relevant Non-functional properties

As we distinguish in the literature, it is possible to classify these non-functional properties according to different points of view. Next, we propose a way to classify them, in order to better explain how these non-functional properties are used in web services discovery, composition, negotiation and substitution.

**2.1.2. Non-functional properties classifications**

We observe that in the literature there are different ways to classify non-functional properties (6) (8) (9) (10). As an abstraction, we propose to classify them according to their reaction in service aggregation, composition, architecture, etc., and according to their semantic description.
As figure 1 shows, the first classification concerns five axes helping to understand how the non-functional properties react in aggregation, in the architecture, etc.

- **Domain-dependency** (9) (10): aspects that can be applied for any web service, or in a specified application domain
- **Negotiability** (9): the provider proposes the same service with different non-functional properties.
- **Type of value** (9): ordinal values normally come from measures, categorical values come from a taxonomy.
- **Direction** (10): in order to have better quality, some non-functional properties have to be maximized (positive), other minimized (negative).
- **Place of measurement**: non-functional properties cannot be measured in all the points of the architecture. There are mainly three places: client, provider and network. When some properties can be measured at different points, they do not measure exactly the same thing.

The second classification is a semantic classification that is intended to help the designer to organize his work. This classification can be described by different levels (8) or by categories (11). Concerning different levels, we can cite:

- **Business View**: These values look at the service becoming a business so it takes more into account the possible clients. Example: price, reputation…
- **Service Level View**: These properties evaluate the quantities that can be evaluated at the service level, i.e. measurement that can be taken during the service execution. Example: Response Time, Throughput…
- **System Level View**: These values quantify the operations of the Web Service and the platforms where it is executed. Example: introspection, authorization…

And an example concerning a classification into categories (11):

- **Performance**: The measurements evaluate the performances of the services in terms of execution. Example: Execution Time, Latency …
- **Dependability**: It evaluates the service not in a single execution but in a more general continuous work. Example: Availability, robustness …
- **Security & Trust**: It tries to classify the services depending on the security issues. Examples: reputation, encryption …
- **Cost & Payment**: It defines the factor that interacts establishing the non-functional properties constraints. Example: price, penalty …

After the description of the non-functional properties and how they could be classified, we introduce in the next section how these properties could be aggregated or composed in order to provide better or different functionalities.
2.2. Aggregation of non-functional properties

We describe in this section the way non-functional properties can be aggregated in a single service or in a composition of services. This aggregation is used to calculate the final value of quality and then to be able to rank services.

2.2.1. Aggregation of non-functional properties in a Single Service

The idea is to obtain a figure showing how good the service is. In the literature we do not find any aggregation system. They use algorithm to maximize all the non-functional properties in parallel (11).

2.2.2. Aggregation of non-functional properties in a Service Composition

For a composition, the idea is to define how the different non-functional properties aggregate in the composition. We have observed that concerning an evaluation of a service composition, two big families exist:

1. Lecue proposes in (11), the definition of a semantic link between the output of a service and the input of the next one (Fig. 2). Using this semantic link, we can define the similarity between the inputs and the outputs of the web services (12). This similarity can either concern the type of information (string, integer, double...) or their meaning (semantic matching). The semantic matching requires an ontology to clearly define the relations between the concepts. This similarity has also been used to identify services that can be equal or exchangeable (13).

   ![Fig. 2 A Semantic Link between two services (11)](image)

   This method seems to be an adequate and detailed method to obtain the best QoS in a service composition. It cannot evaluate only the best services, but also the best composition. Because in a composition the join of the best two services does not give strictly the best possible result, possible mismatching in type or meaning (price in Euros or in Dollars) between the output of the first service and the input of the second can happen.

2. The second view is based on the composition of the QoS of each service (10). However, this method does not consist in a simple addition or multiplication of the non-functional properties, they treat each property differently. The definition of the composition of the properties depends on the conjunctions showed in the composition chart (10). In Fig. 3, we can see an example of composition that depends on the conjunction function given in the state chart.

   ![Fig. 3 Example of composition chart for QoS compositions (10)](image)

   We conclude, that in order to get the best performance and quality we should use a combination of these two methods. The first one gives us a method to evaluate the matching between services in a semantic way and the second one explains how to compose the quality of the services.
2.3. Quality of Service in web services

In order to select the best service adapted to client’s requests, we need some method capable to evaluate and compare different services providing the same functionalities. In this context, Quality of service can be defined as offering service differentiation based on the requirements of clients and applications (14). This implies the evaluation of the non-functional properties of the service described in a quality model. In order to define how quality of service could be evaluated in web services, we present some important concepts related to our research.

2.3.1. Model for Quality of Service

To the best of our knowledge, there is no generic quality model proposed in the literature to evaluate quality of services over web services. Nowadays, most of the quality model proposals are focused in ad hoc applications. As a result we consider that ad hoc quality models used in such applications are basically composed by:

- **Properties**: values exposed by the service or captured on it
- **Classification of properties**: allows to categorize properties and to define the ways of inclusion in the Quality Model.
- **Aggregation formula**: It allows defining a Quality figure for a quality for a given service at a given moment. It specifies also the aggregation of qualities in a service composition.

We detail here after, how quality of service calculation is integrated in a service-oriented architecture. It presents which components should be implemented in order to supply the basic functionalities to retrieve the non-functional properties of the services.
2.4. Service-Oriented Architecture and QoS

Different architectures can be implemented to measure the QoS. They differ according to the location where measurements are executed (9) (15). The components are represented in Fig. 4.

![Diagram of QoS measurement, publication & calculation](image)

In this figure we can identify all the components used in the literature to evaluate the QoS. They are all the possible points where interesting information regarding the service can be measured. In conclusion, after the result of the research, we will be able to propose a general architecture answering the maximum of user cases. But first of all, we will detail each component regarding its functionality in the QoS framework.

2.4.1. QoS Measures

Different components in the architecture can take measures (9) (15) (16):

1. **Provider-side**: the service provider declares its properties. The measures can be done in an invasive way, if the measures are taken through the code of the service, or in a non-invasive way, if the measures are taken by a monitoring framework that encapsulates the service (16).

![Invasive VS non-invasive measurements of QoS](image)

The strengths of this monitoring are the specific measures for the client, and the independence from the network. But it must have the capabilities & willingness of taking the right measures, and the client must trust the provider.

2. **QoS Proxy**: it is a component introduced between the client and the provider during the service invocation. The “QoS Proxies” are intermediary parties that route the traffic and measure the NFP Values. They contain real clients’ measures, real response to real scenario. But, it is a high-run overhead, and it can quickly become a bottleneck. The results depend on the location of the proxy.

3. **Sniffing**: independent service monitors the traffic on the network and produces the NFP Values. It has a lower overhead and the measures can be realistic and consumer-specific. The problem is to find the packet transmitted between the client and the provider and the routes they take, it can imply a security problem and the results strongly depend on the network location.
4. **Server Probing**: the service functions are invoked regularly by an independent component in order to calculate the QoS of the Web Service. This approach takes real measures through the network. It is interesting in order to keep a record about the services and to know if the non-functional properties are always given and to make studies in a long term. But the non-functional properties can change because of the dependence from the network and it is not easy to scale up while measuring a great number of the services.

5. **Client Probing**: the client sends a probe request to the service to evaluate. This request can be generated by own code or generated automatically with the stubs, received from the UDDI, after parsing the WSDL files. It can also be used after the execution of the service. The client can send the quality values of the execution. This approach takes advantage of real measures that the client receives when the service is used. But the measures are not always adapted to other clients because they are too specific to the client requesting the service. This implies that every client who wants to execute a service has to take measures before doing it, and we face big scaling-up problems. On the contrary it could be a good source of information for example for reputation information.

### 2.4.2. QoS Publication

Services QoS can be published in different ways(9)(17):

1. **Expanding WSDL to registry in UDDI**: during the registration in UDDI, the information about the service is extended to include the different QoS Properties. The QoS must be updated by the Web Service leading to possible outdated values or extinction of the service.

2. **QoS Registry**: another component manages the registration of the QoS of several Services. It requests the services for new values and also controls the life of services. It can be extended to have the functionalities of a “broker”. This option is more adapted to guarantee the values freshness. But it requests to add a new component in the architecture and may induce additional delays in services requests.

### 2.4.3. Service Selection according to QoS

As several approaches shows, the selection of a service among candidates can be computed in different devices:

1. **Client-Side**: The client downloads for each service, which responds to the functional needs, all its non-functional properties. Then it computes the selection of the best suitable services.

   This solution is the most adapted to a clients scaling-up philosophy where we cannot have a unique supercomputer to calculate all for us. However, it depends strongly on the device, so the algorithm has to be adapted. Moreover a strong overhead is necessary to transmit all the services with their QoS properties.

2. **Broker**: Here a new component calculates the best Services responding to the client needs based on the preference of the client. It can contain the NFP Values of the services or it can request the QoS from an UDDI or QoS Registry. The second solution keeps the idea of being able to use this component in a distributed UDDI.

   The broker architecture can guarantee the QoS and manage the capabilities of the services, by for example, taking demands to another service in order not to block the best one. It could also implement a negotiation between the services and the broker, in order to choose between the possible compositions, or on the contrary on the constraint that should be relaxed in order to obtain services responding to the demands.

   The main different algorithms to resolve this multi-criteria optimization are: **Integer Programming(18)**, **Genetic Algorithm(11)** (19), or **Constraints Programming(20)**. The results from (19), demonstrate that Gas Algorithm behave better with non-linear aggregation of rules, and they keep this performing in a scaling-up process.

   In the next section, we present a description of how to evaluate non-functional properties in order to select the best web service and also to guarantee continuously the qualities established, due to SLA Agreements.
2.5. Service Level Agreement in Web Services

The quality of service can be used not only to select the best services in order to produce the desired results, but also to establish an *agreement* that ensures the quality level we get from the provider.

For that reason different efforts have been done (9) (21) (22)(23). In such propositions, the main idea is that when the client has chosen a service, a Service Level Agreement (SLA) is established in order to guarantee some qualities. In this “contract”, the penalties which the provider should respect if the specific qualities are not provided are also defined.

In this context, the architecture is adapted to monitor the values of the agreement. This monitoring must be independent. Even though, it should be as close as possible to the client, because the network could interfere with the quality of the provided service.

2.6. Conclusion

In this section we have introduced different aspects concerning the Quality of Service, the Monitoring and the Management of Web Services. To the best of our knowledge, no standard definition and no standard model for the Quality of Service (QoS) exists in the Web Services domain. However, some researches try to apply the QoS definitions existing in software or network domains; others try to redefine models and adapt them to web services.

Existing QoS models are adapted to the particular system they try to evaluate. But, when different systems are combined, for example in the case of a web service, models are not adapted to a generic evaluation. In this context, we conclude that a generic QoS model is necessary in order evaluate the quality of a service.
3. Quality of Service Framework for Web Service

Our review of the literature indicates that all the QoS models presented are not based on a general standard; so, they respond just to a specific scenario. In order to tackle this aspect, we propose a generic framework to evaluate quality of service in web services. Our proposition reuses some existing non-functional properties and classifications.

A standard QoS model has to include the following features:

- It should take into account the heterogeneity of sources,
- It should offer the possibility to evaluate sources in a large number of contexts.
- It should be robust against lack of information (unavailable non-functional properties values).
- It should face scaling-up problems.

Existing works lack a model that defines the constraints and compares the quality of the services. Then, we need to explicit:

- The measures of the non-functional properties (non-functional properties values) that a service must satisfy (constraints).
- The mathematical model of Quality of Service for the comparison on non-functional properties values among all the possible services.

We propose to combine the different service-oriented architectures in a framework in order to provide the best composition for a service selection. This generic framework allows to:

- Define Quality of Service for web services.
- Describe a QoS Model to evaluate the quality of web services.
- Propose an architecture to measure, publish and calculate QoS.

The final objective is to obtain a value of “Quality” in order to be able to rank all the services responding to functional needs. The next figure (Fig. 6) shows in a general way all the steps of our proposition to find a quality value and rank the possible services:

![Fig. 6 General View of all the steps to select the best service](image)

In Fig. 6, we represent all the steps to select the best services. We can subdivide the selection in two big steps. The first one is based on finding the candidates service to the client requirements. First of all, we select the service that gives the functionality desired by the client. Then, we discard the services that do not respond to the non-functional properties values constraints. The second step is to evaluate, for each candidate service, the QoS of the service in order to select the best one. This evaluation is done with the normalized non-functional properties values. Then, we group semantically these non-functional properties. Finally, we calculate a weighted sum that produces the Quality grade, used to rank the services.
In order to grade and select the most suitable service, first we describe how to select the services responding to the functional client demands (Section 3.1). Then, we introduce the difference between the concepts of non-functional properties and Quality of Service (Section 3.2). Next, we define non-functional properties that we can have in Web Services and how we classify, measure and normalize them (section 3.3). Then, we expose our QoS model that helps us to evaluate different services (section 3.4). We give final grading of Quality of Service in section 3.5. We synthesize the quality evaluation in section 3.6. In section 3.7, we describe how the quality can be calculated in a service composition. Finally, we present the architecture to retrieve, publish and calculate the proposal (Section 3.8).

3.1. Preliminary: Service selection by functional properties

Services provide functionalities characterized by a triplet of \(<\text{Input, function, output}>\). The inputs represent the parameters needed to execute the functionality, the method describes what the service does, and the outputs are the results of the function. The inputs and the outputs are objects, lists of objects or also null.

In order to find the best service responding to client requirements we have to find all the candidate services that respond to the desired functions. In push mode, the service registers itself in a web directory with its functional and non-functional properties. Next the client asks for the services responding to a desired functionality (Ex: convert the format of an image, get the map from a GPS position...). Finally, web directory searches and selects among its registered services all the services responding to the functional triplet introduced by the client.

\[
\begin{align*}
  f: & \text{functional property} \\
  select_{f,c}(\omega) , & q: \text{request} \\
  c: & \text{client}
\end{align*}
\]

For each request \((q)\), we select \((select)\) the web services \((\omega)\) base on the functional properties \((f)\) established by the client \((c)\).

3.2. Non-functional properties of Web Services and Quality of Service

In order to describe our proposition, a distinction needs to be done between non-functional properties and Quality of Service. As we state before, a service has functional and non-functional properties. Functional properties define what the service takes as input, what it does and what it gives as the output. On the other hand, non-functional properties define other characteristics or specifications that define how the service reacts. These non-functional properties do not change the behavior of the service.

Quality of Service is an abstract idea that could not be exactly defined, because it changes depending on the domain and depending on the client. A service has a quality if a client says which non-functional properties are important for him. For example: we have a service with high availability and no codification system. For the client, availability is not important and he wants a codification system. The quality of this service is poor because it does not respond to the client demands, even if it has a high availability which is generally a quality item.

Then, we define:

**Non-functional property (NFP):** they are all the properties defining the service and the services, including its performances, that characterize how and in which conditions (context) the functionality of the service is executed.

*Quality of Services in pervasive environments*

*Master thesis*
**Quality of Service**: is the capability to respond to the requirements (constraints) of a client and to fulfill these needs with the best criteria (preferences) established by the client. It is calculated based on the non-functional properties of the service.

In fact, some non-functional properties can be seen as functional properties. For example, we can have the same functionality published in three different services like: `loginWith128DES()`, `loginWith256DES`, `loginWithRSA()`. Then the codification becomes a functional property. We can see that even if the service publishes these three functionalities, they all belong to the same functionality of `login()` with a specific non-functional properties called encryption. This example shows that the line between functional and non-functional properties is sometimes fuzzy. Even though, we encourage describing functional properties only what the service does and giving the maximum of all the other information in the non-functional properties. This, helps the system compare the different functionalities of the service. The usage of non-functional properties helps the services publish the same functionality with a range of possible non-functional properties values.

In the following, we extend the state of art by developing a classification of non-functional properties.

### 3.3. Non-functional Properties

As we have described in section 3.2, non-functional properties can be measured to establish the Quality of a service. We define:

**Non-functional Property Value (NFP Value):** is the value issue from measuring a non-functional property.

Constraints are defined on the non-functional properties, and with these constraints we are able to select the services responding to client requirements. For example, the client can request for a Round Trip Time to be less than a certain value or for a maximum availability. The constrains selection is represented as:

\[ \text{select}(f, k, p, q, c) (\text{NFP}) \]

We select (select) the web services (NFP) based on the functional properties (f), as seen in the chapter before, and on the constraints (k) defined for each property (p). This parameters are defined by the client (c) for each request (q).

In this section we propose a classification of non-functional properties adapted to our context. Next, we describe the different possible web service hosts (sensors, servers and mobile softwares), and we continue by explaining how to measure non-functional properties in each context. We conclude with the normalization function that will help us to combine the NFP Values.

### 3.3.1. Non-functional properties classification

In order to calculate the non-functional properties, we need to organize these properties in a functional classification. We base our classification on those explained in section 2.1.2:

- Domain-dependent vs. domain-independent
- Negotiable vs. Non-negotiable
- Ordinal vs. Categorical
- Positive vs. Negative
As we have seen in section 2.1, a service can be an atomic service (single service) or a composition of other services. We would like to compute the aggregation of non-functional properties in the case of a composition. Then we propose to extend non-functional properties classification taking into consideration how non-functional properties behave in a composition (details for calculation see 3.7.1):

- **Additive**: these properties values are added to the other service based on the composition graph. Example: duration, energy usage.
- **Multiplicative**: properties are multiplied to the other service values based on the composition graph. Example: availability, reliability. They include intrinsic properties like security, accessibility, connection.
- **Extreme**: properties depend on the maximum or minimum value among all the services in the composition. Example: availability, reliability. They include intrinsic properties like security, accessibility, connection.
- **Other**: properties have a specific way of reacting in composition. Normally they mix the other reactions. Example: cost.

### 3.3.2. Web Service Host

A Web Service is an interface created by a local service in a host. We can differentiate three kinds of hosts. For each host type we can weigh the different properties so as to calculate the best quality of service, based on the host type. The hosts are described as follows:

- **Sensors**: They are hardware that take measurements at intervals of time. Most of the time, they remain in a sleep mode, then, they turn on, make the measurements, send the data before going to sleep mode again. The main properties of these components are for example: availability, response time, battery level, error rate.
- **Servers**: These hosts give information directly from a database or after a software treatment in an internal function. Properties of these components are for example: availability or response time.
- **Mobile software**: These hosts are between sensors and servers because they can compute information but they have mainly the same kind of restrictions as sensors. Their main properties are for example: availability, response time, connectivity, battery level.

### 3.3.3. Getting Non-Functional Properties Values

A non-functional property is an element of a quality model that represents a characteristic of the service or the host giving the desired functionality. Non-functional properties values should be numeric to be composable into a grade. We propose to represent the numerical value of this element by two main mechanisms: measurement (ordinal NFP, cf 2.1.2) or taxonomy (categorical NFP, cf 2.1.2).

#### 3.3.3.1. Non-Functional Properties Measurements

Metrics allow to assign a value for all the ordinal properties presented in the previous sections. We need a process to evaluate a property giving an exact value. We propose three processes:

- **Direct measure**: the value retained for the non-functional properties is the one issue from a direct measure at the evaluation moment.
- **Calculation**: we can use theoretical calculation, based on statistics evaluating an estimation of the possible real value. Example:
  \[ \text{Execution Time of a service function} = 0.15 \text{ ms} \times 25 \text{ Memory Access} + 2.0 \text{ ms} \times 25 \text{ Access Disk} \]
- **Measure**: we can also use external ways to measure the behavior of past values and then expect that the service behaves in the same way. They represent real performance of the service. Example: Executions time is measured as the difference time measured with an external clock
  \[ \text{Execution Time} = \text{Mean (Execution time)} \]
In Table 3 we show how these measurements are different depending on the type of host:

<table>
<thead>
<tr>
<th>Type of Source Property</th>
<th>Sensor</th>
<th>Server</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>0 Non accessible, 1 Accessible</td>
<td>1 - Downtime Measurement time</td>
<td>Non applicable</td>
</tr>
<tr>
<td>Battery Level</td>
<td>Measured in percentage</td>
<td>Non applicable</td>
<td>Measured in percentage</td>
</tr>
<tr>
<td>Average Speed</td>
<td>LastPosition - Position / LastTime - Time</td>
<td>Non Applicable</td>
<td>LastPosition - Position / LastTime - Time</td>
</tr>
<tr>
<td>Position</td>
<td>GPS Position</td>
<td>Static Position</td>
<td>GPS Position</td>
</tr>
</tbody>
</table>

Table 3 Measures Depending on the Service System

In this table we can see how the properties are measured and the dependence to the service host. Some non-functional properties may not be measured for some host. It means that they are not relevant to the quality evaluation for this kind of service.

3.3.3.2. Non-Functional Properties Taxonomy

For the categorical non-functional properties we need a taxonomy that will attribute a numerical value to each categorical value. We need taxonomy or some other kind of definition accepted and used by all the services to guarantee a coherent system.

More concretely, in our proposition the taxonomy helps to:

1. Express the same concept in a non-functional property. We make sure that all the service measure the same thing and that they measure it using the same units.
2. Evaluate categorical properties with the same values. The usage of a common taxonomy ensures that the same non-functional categorical value is evaluated with the same result.

3.3.4. Normalization function

We need a method to aggregate the different non-functional properties. The goal is to aggregate them in order to obtain a final Quality grade that allow to rank the services. The final result of the normalization function will be a value between 0 and 1: $N: (NFP Value) \rightarrow [0,1]$. This normalization is done for all the non-functional properties (categorical and ordinal).

This function is essential, because it qualifies and gives a grade to non-functional properties. It also, allows to qualify the non-functional properties, comparing among them.

In this proposal we suggest to use the normal distribution normalization calculated as follow:

$$N: x \rightarrow \frac{x-\mu}{\sigma}, \mu: mean \ value \ of \ the \ variable \ x, \sigma: standard \ deviation \ of \ the \ variable \ x$$

This normalization is represented as follow:

$$norm(\mu, \sigma)_{p,q} \left( select(f_k, \rho)_{q,c}(\omega) \right), \mu: mean \ value \ of \ the \ variable \ x, \sigma: standard \ deviation \ of \ the \ variable \ x$$
The normalization function \( (\text{norm}) \) is calculated for all the non-functional properties among the services previously selected \( (\text{select}(f,k,p)_{q,c}(\text{as})) \). This normalization depends on the statistical values: mean value of the variable \( (\mu) \) and standard deviation of the variable \( (\sigma) \). This parameter depends on the non-functional property \( (p) \) and is calculated for each request \( (q) \).

In annex 7.1, we evaluate the possible normalization functions and propose a different normalization function which does not depend on the statistics, but on the client preferences.

### 3.4. Quality of Service Items

Once we have the measurements of the non-functional properties of the service, we need to compare the services and to establish the quality of a service. To this end we propose a generic Quality of Service Model. This model is based on the quality view of a service.

We define:

**Quality Item**: it is an evaluation aspect of a service. It represents a semantic grouping of non-functional properties representing the same evaluation concept of a service.

**Quality Model**: it is a quality items classification based on the service level evaluated by the quality item. The quality model covers all the possible aspects of a service evaluation.

In our proposition, the Quality Items does not depend on the type of host, while non-functional properties do. The belonging of the non-functional properties is pre-defined. If new properties appear, they need to be related to a quality item in order to take them into account in the quality of service evaluation.

![Fig. 7: Representation of the belonging of the non-functional properties to a quality item](image)

In Fig. 7, we see the matching between non-functional properties and quality items. In our proposal we consider that a non-functional property belong to only a single quality item. At the same time a non-functional property is referenced in the taxonomy, it has to define to which quality item it belongs.
In our proposition we have built the following QoS Model, extended from the state of art propositions given in section 2.1.2:

**Fig. 8**: QoS Model including all the Quality Items concerned in a service evaluation

In Fig. 8 we represent the QoS model used in this proposition. It subdivides the quality Items in three level of service evaluation. First, the quality related to the service host. Secondly, we group the quality of the service itself. Finally, we group the quality of the business and trust aspects of the service. All QoS Items values in this model are between 0 and 1 and all are positive.

The QoS Items values are calculated as the mean of all the non-functional properties responding to this concept (semantic grouping), these properties are already weighted in the normalization function:

\[
\text{Quality Item}_i = \frac{\sum_{j=0}^{n} p'_{ij}}{n}, \quad \forall \ p \neq 0 \in \text{Quality Item}_i
\]

The semantic grouping realized in the quality evaluation helps to introduce different non-functional properties that different services have in the same concept. If a non-functional property does not exist, it is simply not used. It resolves the problem of non-specified properties.

If all the non-functional properties of a quality item are not evaluated, the final value of the quality item is 0. In this solution, it penalizes the services not giving non-functional properties and it ensures that the final service proposed responds perfectly to the client needs.

The semantic group function of non-functional properties values is represented in the next expression:

\[
group_{qi} \left( \norm_{(\mu,\sigma)_{pq}} \left( \text{select}_{(f,k)_{p,q,c}}(\text{ws}) \right) \right), q_i: \text{Quality Item}
\]

We group \((\text{group})\) the normalized non-functional properties of the previous selected services \(\norm_{(\mu,\sigma)_{pq}} \left( \text{select}_{(f,k)_{p,q,c}}(\text{ws}) \right)\). This grouping depends to the belonging of the property to a Quality Item \((q_i)\).
3.5. Quality of service grading

The final Quality of Services value is calculated as the weighted sum of the different quality items values (weighting parameters introduced by the client):

\[ Q_{WS} = \sum_{i=0}^{n} \alpha_i q_i \]

This model lets us compare two services by making an average of all the quality items values. The end client would be able to decide the most important qualities using quality items weighting parameters or more precisely using the non-functional properties normalization function.

The final ranking is expressed as follows:

\[ \text{rank}_{(\alpha)_{q,c}} \left( \text{group}_{q_i} \left( \text{norm}_{(\mu,\sigma)_{p,q}} \left( \text{select}_{(f,k)_{q,c}} (\omega_s) \right) \right) \right), \alpha: \text{weight parameters} \]

The final ranking function \((\text{rank})\) is calculated to all the quality items calculated previously in the semantic grouping \((\text{group}_{q_i} \left( \text{norm}_{(\mu,\sigma)_{p,q}} \left( \text{select}_{(f,k)_{q,c}} (\omega_s) \right) \right)\)). This ranking depends on the weight parameters \((\alpha)\) introduced by the client \((c)\) for each request \((q)\).

3.6. Synthesis of Quality of Service Calculation

The entire process of Quality evaluation can be expressed as follows:

\[ \text{rank}_{(\alpha)_{q,c}} \left( \text{group}_{q_i} \left( \text{norm}_{(\mu,\sigma)_{p,q}} \left( \text{select}_{(f,k)_{q,c}} (\omega_s) \right) \right) \right) \]

with:

- \(\omega_s\): web service
- \(\text{select}\): selection function
- \(\text{norm}\): normalization function
- \(\text{group}\): semantic group function
- \(\text{rank}\): ranking function
- \(\mu\): mean value of the non-functional properties values
- \(\sigma\): standard deviation of the non-functional properties values
- \(q_i\): quality item

and the parameters added by the client:
- \(c\): client
- \(q\): request
- \(f\): functional properties
- \(p\): non-functional properties
- \(k\): constraint on the non-functional properties
- \(\alpha\): weight parameters
In this process we rank all the functionally candidates services depending on their quality grade. This QoS is calculated by a weighted sum of the semantic grouping of the normalized non-functional properties values.

The QoS grade of a service is calculated as:

\[ QoS: NFP \rightarrow R \]

\[ (NFP_{(1,n)} + \mu_{(1,n)})^T \times \sigma_{(n,n)} \times Gr_{(n,m)} \times \alpha_{(m,1)} = q_{(1,1)} \]

With:
- NFP: non-functional properties values
- \( \mu \): mean value of the non-functional properties values
- \( \sigma \): inverse standard deviation of the non-functional properties values
- Gr: semantic grouping, establish the belongings on NFP to a quality item
- \( \alpha \): quality items weighting parameters
- n: total number of not-null non-functional properties for a service
- m: number of quality items in the quality model

This formula allows to calculate the QoS grade by the normalization of the non-functional properties, the semantic group and the final weighted sum.

3.7. Towards Quality of Service evaluation of composition services

A composition of Web Services can be seen as a state diagram or a workflow. The client can provide global constraints for composition or local constraints on a component/service in the composition. In our context we need to establish the final quality of composition knowing the single non-functional properties of each selected service in the composition.

*Fig. 9 Web Service Composition*

In Fig. 9, we represent a standard composition graph containing the three main component/services links. The sequence link is between two services that the second one waits for the results of the first one. An AND link establish that the next service after the first one is two (or more) services instead of one, and all services needs to be executed. An OR link, on the contrary, only one of the possible services will be executed.
We suppose an existing taxonomy (not an ontology) defining the different functional and non-functional properties that are used by all the services. In our context we cannot use the semantic link concept due to a lack of ontology. Actually the semantic link can be represented as additional non-functional properties.

Our Quality of Service evaluation allows to:

- Compare already established web services compositions.
- Find the best suitable service, considering local constraints of a component of a composition.

If global constraints are specified, this solution requires a preliminary selection of all the candidates’ composition responding to these constraints. In annex 7.2 we propose a method using the Quality of Service evaluation to improve the performances finding the most suitable services for a composition.

As for the individual service, we distinguish two steps in the evaluation. First we calculate the composition non-functional properties as an aggregation of the individual components non-functional properties. These composition non-functional properties help the client to establish constraints in the composition performance. Next, is the quality evaluation of the composition based on the non-functional properties of the composition (as if they were from a virtual service) to obtain a grade and rank the composition.

3.7.1. Composition of Non-functional Properties Values

In order to distinguish how non-functional properties aggregates in a composition we need to use the different non-functional classifications (section 3.3.1). We propose the aggregation of non-functional values in a composition as shown in Table 4:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Sequential</th>
<th>AND</th>
<th>OR</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additive</td>
<td>ad</td>
<td>( \sum_{i=1}^{n} ad_i )</td>
<td>max (ad_i)</td>
<td>max (ad_i)</td>
</tr>
<tr>
<td>Multiplicative</td>
<td>mu</td>
<td>( \prod_{i=1}^{n} mu_i )</td>
<td>( \prod_{i=1}^{n} mu_i )</td>
<td>min (mu_i)</td>
</tr>
<tr>
<td>Extreme</td>
<td>ex</td>
<td>min (ex_i)</td>
<td>min (ex_i)</td>
<td>Throughput, reliability</td>
</tr>
<tr>
<td>Other</td>
<td>ot</td>
<td>( \sum_{i=1}^{n} ot_i )</td>
<td>( \sum_{i=1}^{n} ot_i )</td>
<td>max (ot_i)</td>
</tr>
</tbody>
</table>

Table 4: Aggregation of non-functional values in a composition

This table shows the specified formula to aggregate the non-functional property values depending on their classification. To calculate the non-functional property values of a composition we aggregate all the non-functional properties of each component/service based on the composition graph (ex: Fig. 9) and on the non-functional property classification (Table 4).

3.7.2. Composition of Quality of Service Model

With the values for the composition, we can compare two compositions as the comparison of two services. In consequence, we use the same normalization function described in section 3.3.4 and calculate the final quality grade as shown in section 3.5.
3.8. Architecture for the evaluation of Quality of Service

In our context the required architecture has to:

- Take the measures of the non-functional properties where they need to be measured
- Publish this information in an efficient way.
- Calculate the QoS grade

Some constraint must also be followed:

- Being prepared for an scaling-up of services and of clients
- Being ready for all different kind of environments (pervasive, static …)

In this proposition we try to develop a general architecture for evaluating the QoS in Web Services, and demonstrate how the pervasive environments are taken into account in this solution.

Our architecture (Fig. 10) is composed by:

Fig. 10 : Generic Architecture for evaluating Quality of Service

1. **Provider**: is the component that provides the service.
2. **Web Directory**: it is the component where all the services are published with their functional and non-functional properties.
3. **QoS Monitoring**: it is a new component that helps to integrate the QoS evaluation needs in the Web Service Architecture.
4. **Client**: is the end user or computer who requests a service from the UDDI and after get the service through the provider.

In this architecture we distinguish the three main functionalities for a QoS environment: measurements, publication and calculate the quality. We describe these functionalities as follow:

3.8.1. Non-functional properties measurements

Measurements are done in three points:

1. **Provider**: The provider takes its own measures. In preference the measurement should be taken in a non-invasive way. The idea is to use a common measurement system in all the providers that ensures the values are coherent and reliable.
2. **Client**: for the client would be similar in the code is generated with the stubs after parsing the WSDL so we are sure of coherent measurements.
3. **QoS Monitoring**: its function is not to retrieve the QoS properties of the host, but to make sure these qualities are the real ones. This component distinguishes between the source type and the frequencies the service are demanded. Depending on this classification the QoS Monitoring makes request to: be sure the sources are alive, retrieve the more updated QoS and test the QoS given are correct. We are based on the next table:
Table 5: Update frequency depending on service host and the service demand

<table>
<thead>
<tr>
<th></th>
<th>Alive</th>
<th>Update</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High Demand</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>Low Demand</td>
<td>0.9</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>Software Device</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High Demand</td>
<td>0.3</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>Low Demand</td>
<td>0.9</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High Demand</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>Low Demand</td>
<td>0.1</td>
<td>0.6</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The values are relative, so they represent how many times we send these specifics request for a single unit of time. The concrete numbers can be modified but the most important is to maintain the relation between the values.

3.8.2. Non-functional properties publication

For publication we use:

1. Web Directory: structured like a classical web directory, it also contains the non-functional properties values gathered at publication. When a service announces itself, it includes its functional description and non-functional properties. The QoS Monitoring components are in charge of the non-functional properties updating.
2. Service Provider: each service provider can publish its non-functional properties values. A client can contact directly the provider to get fresher values.

3.8.3. Quality of Service calculation

The calculation of quality of service can be done at 2 different places:

1. QoS Monitoring: this module is in first-line as it knows the non-functional properties values. So calculating quality of service there does not introduce a throughput for the transmission of the non-functional properties values.
2. Client: in adhoc networks or in the case the QoS Monitoring fails, the client could retrieve the non-functional value from the web directory, or directly from the providers and make the quality of service calculation.
4. OPTIMACS Specifications

OPTIMACS is a national project (24) for optimizing query where sources of information are services. The query optimization step includes the selection of a service among all the functionally compatible ones. This selection will be based on non-functional properties and Quality of Service plus the description of services mobility.

4.1. Non-functional properties identified in OPTIMACS

A working group of the Optimacs project has built a first list of non-functional properties that will be measured in the project. We have classified these non-functional properties in our classification presented in section 3.3.1. The result is given in Table 6.

<table>
<thead>
<tr>
<th>Static</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device type</td>
<td>Total storage capacity</td>
</tr>
<tr>
<td>Battery Level</td>
<td>Available storage capacity</td>
</tr>
<tr>
<td>Total Memory</td>
<td>Framework</td>
</tr>
<tr>
<td>Available Memory</td>
<td>Deployment model</td>
</tr>
<tr>
<td>Supported Types</td>
<td>IP Address</td>
</tr>
<tr>
<td>Location Type</td>
<td>Mobile Profile</td>
</tr>
<tr>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>Bandwidth</td>
</tr>
<tr>
<td>Position</td>
<td>Average Speed</td>
</tr>
</tbody>
</table>

|                         |                             |
|                         | Availability                |
|                         | Reponses time               |

Table 6: Non-functional Properties in the OPTIMACS specifications

Some properties are directly related to services themselves, but most are related to the host supporting the service execution or the end user device. Moreover, some items are dynamic, but most of them are static information. This implies that the OPTIMACS implementation of our proposal can be a simplification of our proposition. Static information can be captured once and stored at the QoS calculation location. Dynamic information can be captured and exchanged, and as they are sparse, no big efforts are required for the optimization of the protocol and exchanged messages.

4.2. Architecture of the OPTIMACS Testbed and implementation of QoS in this Architecture

The OPTIMACS solution is provided in a generic manner, independently from any implementation issue. But one of the main objectives of the project is to provide a testbed and to build experimental validation of the proposal. The LIRIS members of the project are, among all, in charge of the development of this testbed, and this has allowed us to implement, test and validate our QoS model, even if in a degraded mode.

The testbed is developed on top of the OSGI (25) middleware. It is a service-oriented component based platform. The existing architecture is presented in Fig. 11.
Fig. 11: Current Architecture in the Project OPTIMACS

This architecture is based on 3 main components:

1. **DataSource Containers (DS Containers)** publish services accessible from the network. Each DS Container has three main parts: the OSGI base, the Communication OPTIMACS in charge of the communication with the HTTP/REST protocol and an extensible number of instances of Services.

2. A **DataSource Manager (DS Manager)** registers all the services with their functional properties. The client requires this DS Manager to find the required services.

3. **Client**: it can be a human or software client that retrieves the information of the Registry to find the required Service and then interact with the selected services. When the client is a software system, it includes a query engine that captures complex end-users queries.

To include QoS in this architecture, we require elements that accomplish 3 tasks:

- Measure non-functional properties
- Publish non-functional properties
- Calculate best service

We have designed the architecture provided in Fig. 12:

Fig. 12: QoS-aware OPTIMACS architecture

*Quality of Services in pervasive environments
Master thesis*
In DS Containers, we have added two kinds of components:

- A QoS Monitoring Service that is in charge of evaluating the non-functional properties related to the service host. It is developed itself as a service in order to publish properties and provide some methods related to non-functional properties measures and update.
- Non-functional properties measuring components. Each NFP Measuring Component is dedicated to the measure of non-functional properties of a service offered in the DS Container.

The QoS Monitoring manages the Non-functional properties measuring component in order to ask the non-functional properties and also to ask a reevaluation of the service performances.

In the DS Manager we have added 2 components:

- A Global QoS Monitoring service that is in charge of refreshing the non-functional properties and monitoring the services themselves.
- A Calculate Best Service (CBS) service that gets the functional demand and the parameters of the client and ranks the services so as to provide the best service answering the client demand and responding the best to the client preferences.

The Global QoS Monitoring communicates with the QoS Monitoring Services hosted in DS Containers to update the values of non-functional properties or demand a reevaluation of the non-functional properties values (if the measures are not up-to-date).

We have also added a third component useful in a degraded mode of the use of OPTIMACS. This component is hosted on the client and has the objective of calculating the most suitable service after retrieving the non-functional properties values from the Registry.

The 3 tasks required for QoS Calculation are thus dispatched in the OPTIMACS architecture as follows:

1. Measure of Non-functional Properties Values

   In the OPTIMACS architecture the provider sends information of its own non-functional properties. Consequently the development is in the web service provider. We use a container that encapsulates the service as a non-invasive server monitoring or a proxy monitoring (cf. 2.4.1).

   For a detailed explanation, that provides sequence diagrams of the measures of non-functional properties in OPTIMACS see annex 7.3.2

2. Publish Non-functional Properties Values

   The non-functional properties are published by each provider as an extension of the functional properties. They are also accessible from the registry, because each service publishes its functional and non-functional properties. The QoS Monitoring is in charge of updating the values of the registry when values are outdated.

   For a detailed explanation presenting the sequence diagrams see annex 7.3.2

3. Calculate the most suitable service

   This functionality is done by the Calculate the Best Service (CBS) service in the web directory. But as the publication is executed in the service and in the registry, the calculation can be done at the client side in a degraded mode if the CBS module is not responding.

   For a detailed explanation presenting the sequence diagrams see annex 7.3.4.
5. Evaluation & Results

The evaluation of a quality model is not a simple task. Evaluating an evaluation procedure became impossible, comparing to other evaluation procedure is not possible. The only possible evaluation is to compare according to the performance of the evaluation itself. The solution is evaluated following three aspects:

- Service selection change depending on the client desire
- Performances in augmenting non-functional properties
- Performances in augmenting the number of services

1. Service selection change depending on the client desire

In the first evaluation we use 2 services and 2 properties characterized as follows:

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Service URL</th>
<th>Availability</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestService</td>
<td><a href="http://134.214.107.71:8080/ds/TestService">http://134.214.107.71:8080/ds/TestService</a></td>
<td>0.9594709413888658</td>
<td>1083.9457247285793</td>
</tr>
<tr>
<td>TestService_1</td>
<td><a href="http://134.214.107.71:8080/ds/TestService_1">http://134.214.107.71:8080/ds/TestService_1</a></td>
<td>0.6201286636637098</td>
<td>1034.0601136601756</td>
</tr>
</tbody>
</table>

We use a statistical normalization using the mean value and the standard deviation. We use the same 0.5 weighting coefficient for Performance (correspond to: Response time) and Accessibility (Correspond to: Availability). With these values we obtain the follow quality grades:

<table>
<thead>
<tr>
<th>Service</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestService</td>
<td>0.39655759683</td>
</tr>
<tr>
<td>TestService_1</td>
<td>0.6034424031616217</td>
</tr>
</tbody>
</table>

As we have seen the availability grade difference between the services is more important that makes the service TestService_1 obtain a better grade. Now, we change the weighting coefficient to 0.8 for Performance and 0.2 for Availability. Then the grades become:

<table>
<thead>
<tr>
<th>Service</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestService</td>
<td>0.634921549414053</td>
</tr>
<tr>
<td>TestService_1</td>
<td>0.36550784505859474</td>
</tr>
</tbody>
</table>

The client has selected the performances as his most preference, then the grade calculation take more in account the response time variation than the availability. Then, the service TestService obtain a better grade as I has a lower response time.

With this first evaluation we have proved the importance of normalization function and the power given to the client to define his preferences in the quality of service evaluation.

2. Performances in augmenting non-functional properties

Now we will experiment the performance of the algorithm augmenting the number of non-functional properties. We will use 2 service and we will test the response time 100 times for each number of non-functional properties. The histograms of each measure are in annex 7.4.1.

In this evaluation the services and the calculation are located in the same machine.

The results are represented in the next Fig. 13:
Fig. 13: Statistical values in the response time depending on the number of NFP

From Fig. 13, we conclude that the number of non-functional properties influences the response time. On the contrary, the values follow an augmenting tendency, but not really clear. That could be because a variation of the number of non-functional properties too little to retrieve tendency functions.

The minimum values show a better tendency of augmenting the response time as the same time of the number of non-functional properties.

Even thought, the results are satisfying because the response time remains little enough and we do not see a huge augmentation for response time while augmenting the number of non-functional properties.

If we would like to define a better relation between number of services and the response time, we should first deploy the services in a different machine and secondly increment the number of services enough to clearly depend on the calculation algorithm.

3. Performances in augmenting the number of services

Now we will experiment the performance of the algorithm augmenting the number of services. We will use 2 non-functional properties and we will test the response time 100 times for each number of services. The histograms of each measure are in annex 7.4.2.

In this evaluation the services and the calculation are located in the same machine.

The results are represented in the next Fig. 14: Statistical values in the response time depending on the number of servicesFig. 14:
Fig. 14 : Statistical values in the response time depending on the number of services

From Fig. 14, we conclude that the number of service influences the response time. On the contrary, the values do not follow a clear tendency. That could be because a strong relation depending on the performances of the machine when the calculation is done.

The clear tendency of the maximum values could be a consequence of having the deployment of the services in the same machine where the calculation is done. In consequence the most services we deploy the less resource we have left.

Even thought, the results are satisfying because the response time remains little enough and we do not see a huge augmentation for response time while augmenting the number of services.

If we would like to define a better relation between number of services and the response time, we should first deploy the services in a different machine and secondly increment the number of service enough to clearly depend on the calculation algorithm.
6. Conclusion

In this Master thesis, we are interested in evaluating the Quality of Service of Web Services with an implementation in pervasive environment. Nowadays more and more services are accessible through the network, and this tendency is just growing each time more and more. In this context we need a system to evaluate the services between them independently for the context.

First of all, we made a research in the state of art to be able to more precisely define the problems and the existing solutions. Then we realize that the importance of Quality evaluation and a lack of general model exploitable in a heterogeneous environment. We have studied the architecture used nowadays to evaluate this QoS, regarding all the positive and negative aspects of each one in order to be able to propose a general architecture taking advantage of the most efficient architectures.

We start defining what quality is in a heterogeneous context:

**Quality of Service**: is the capability to respond to the requirements (constraints) of a client and to fulfill these needs with the best criteria (preferences) established by the client. It is calculated based on the non-functional properties of the service.

Then we have defined a QoS model to grade the services in order to rank them. This model contains all the possible aspects of a service evaluation as showed in Fig. 8. These aspects, called Quality Items, are calculated as an aggregation of the non functional properties (NFP) of the service. These non functional properties depend on the context, in order to adapt the evaluation to the service context. The whole process of grading the service is explained in chapter 3. The main contributions of this work are to be able to evaluate a service in the following situation:

- Independent of the context (wide range of services host)
- Lack of information (non-functional properties without a value)
- Adapted the most to the client: 3 points of customization: constraints, normalization function and weighted sum (cf. Fig. 6).

We have then established an architecture to evaluate this calculation. 3 main functions are given in the QoS framework: measure NFP, publish NFP values and calculate the best service.

Finally we have tested this Quality of Service evaluation in order to see the performance of the system in a real case. For that we have implemented an OSGi Architecture in the case of the OPTIMACS project as explained in chapter 4. The results are showed in chapter 5.

This project starts the work of a complete evaluation of Quality of Service evaluation. Four main research projects could be followed to better concretize the Quality evaluation process:

- The first aspect is the standardization if the non-functional properties measures. As we have seen, these measures are the base of the quality evaluation. So we need a way being sure the values we take are correct and can be guaranteed.
- The second aspect is the normalization function. Here we present a general approach in 3.3.4 and in annex 7.1, we see how this function can be more adapted to the client. This is another crucial point, NFP values come from different contexts and are evaluated in order to have a grade independent of the context. So an effort should be done to help the user express his preferences and to reflect them in the NFP Values normalization function.
- The third aspect is the composition. We explain the possible usages in chapter 3.7 and in annex 7.2 we introduce a possible first steps to improve the performances of a composition. The objectives would be to use this Quality Model to help finding the most suitable services in a composition by first computing the composition quality and then, checking whether it respects the constraints. A genetic algorithm would be a solution.
- The fourth one is to expand the solution to use ontology in order to use the concept of semantic link between services and to take into account the evaluation of the links between the services in a composition. That is really important in dynamic contexts or specific demands where we do not have an exact matching between our functional demands and the offered services. In this case we can afford loosing an exact matching by having better quality in other aspects.
7. Annexes

7.1. Normalization Function

As we can see the matching function is a crucial operation because it evaluates the non-functional properties and compares the non-functional properties between them.

The two main functions are:

1. Basic Normalization
   \[ x' = \begin{cases} \frac{x - \min}{\max - \min}, & \text{if } x \text{ a positive property} \\ \frac{\max - x}{\max - \min}, & \text{if } x \text{ a negative property} \end{cases} \]

2. Standard score (Z-score)
   \[ z = \frac{x - \mu}{\sigma}, \quad \mu = \text{mean}, \quad \sigma = \text{standard deviation} \]

The problems of this solution are:

- Dependence to statistics
- Do not give the power to the client to decide how much the property is important and the most important how much he evaluates this importance.

We suppose that the non-functional properties values are normalized distributed, then the normalization function react as shown in the Fig. 15:

Fig. 15: Normalization result of non-functional properties distribution

This normalization reacts well for large distribution values and only in a situation where we know all the range of possible values. On the contrary the first solution react badly with extreme values and the second on react badly in non normalized values.
This figure shows how sometimes the normalization functions give erroneous qualifications. In the first one the extreme values cause a bad final evaluation. In the second one a non-normalized distribution makes that the qualification gives erroneous values consequence of non adapted statistics calculation.

The ideal way is to let the client specify how the qualification/normalization should be done. The calculation is similar to Z-score but the user will specify the mean and the deviation to use:

\[
x' = \begin{cases} 
\frac{x - \delta}{\varepsilon} & \text{if } \frac{x - \delta}{\varepsilon} < 0 \\
0, & \text{or also calculates} \\
1, & \text{if } \frac{x - \delta}{\varepsilon} > 1 \\
\frac{x - \min}{\max - \min} & \text{max value who represents the } 1 \\
0, & \text{min value who represents the } 0 \\
1, & \frac{x - \min}{\max - \min} > 1
\end{cases}
\]

With this definition we establish a normalization function independent from statistics of the non-functional value and completely adapted to the client. The first way of pondering the properties is by adapting the normalization parameters.

The normalization curbs are represented in Fig. 17 and Fig. 18 depending if the non-functional properties are positive or negative:
As we can see in the graphs, the constraint values do not interfere in the matching function and some services not responding to the constraints can have a value but they are discarded in the first part of the evaluation.

The remaining services are evaluated between zero and one. After a threshold all the services are evaluated as 0 (bad quality) or 1 (good quality). That expresses the desire of the client that after a point any more improvement does not add any value.

This function can be even modifying to adapt more the desire of the client. But we consider that the most important is to make it client-friendly so we chose the first level normalization function.

Then the final Quality of Service note is calculated as:

\[
(NFP_{(1,n)} + \delta_{(1,n)})^T \times e_{(n,n)} \times Gr_{(n,m)} \times \alpha_{(m,1)} = q_{(1,1)}
\]

In conclusion, the normalization function should not depend on the statistics but on the client desire. Because is the way the client has to qualify the non-functional properties and compare this qualification among all the other non-functional properties.
7.2. Quality Model in Web Service Composition

If we use the same methodology as for evaluating a single service for a composition, we need first to composite the non-functional properties in order to remove the composition that does not response to our constraints. The problem is that we can calculate an exponential number of possible compositions.

Even thought once the composition is accepted, we use the same methodology to rank all the composition.

Here we propose to use the opposite solution. First we evaluate a rank all the services then we make the possible composition with the best services for each functionality. Then we test if the best composition response to the constraints.

We cannot suppose that the best services responding to each functional step will give the best composition but it will help as initial population for algorithm like genetic algorithm to converge much more quickly if an optimum solution exists. Because if a solution exists, it uses the best services for each functional step.
7.3 Sequence diagrams implemented in the architecture

In this part we present the sequence diagrams between the components in the architecture, for the 3 main tasks of the QoS implementation: measure, publish and calculate.

7.3.1. Publish a service

The first main important point is to register services to a Registry, so the clients can find the service when they ask for a desired functionality. This registration is done by the service to Registry the in a push mode, but also in a pull mode by the Registry requesting all the services accessible from the network.

![Sequence diagram]

Fig. 19: Registration by the Provider in a push mode

As we can see in Fig. 19, it is the Communication component that publishes the services, because this component is the one who know all the services contained in the Service Host.

![Sequence diagram]

Fig. 20: Service Discovery in a pull mode done by the Web Directory

In Fig. 20 we see how the registry can also broadcast the network to find the services published on the network. This loop functionality can be adapted to not overflow the network but also be aware of dynamics environments.

7.3.2. Measure of non-functional Properties

As explained in section 1 the non-functional properties are measured in the provider. Different protocols are established to have up to date values of the non-functional properties.

For dynamic non-functional properties, the measures are taken when services are called. This happens in three possible scenarios:

1. The Service is called by a client
2. The Service is called by the provider itself
3. The Service is called by the Registry, more precisely the Global QoS Monitoring

The corresponding sequence diagrams are shown in Fig. 21, Fig. 22 and Fig. 23:
Fig. 21: Invocation of a Service function for a client

In figure 18 the service functionality is monitored and the values are taken when the client access the service function.

Fig. 22: Self-Test of the Services from the Provider

Figure 19 shows the case of the services evaluate the performances of its own functions. This action is started by intervals, when the non-functional properties are not up-to-date.

Fig. 23: Test done by the Web Directory, more precisely by the QoS Monitoring

This figure represents the case when the Web directory finds that a services is not updating the non-functional properties values. Then the QoS Monitoring component starts the functional evaluation of a service.

7.3.3. Publish non-functional Properties

As explained in 2 the publication is done through the provider and the registry. The publication in the registry is done in three scenarios:

1. Register Service, at the same time the service registers its functional properties it also register the non-functional properties. (cf.7.3.1)
2. Update Service: when the service change its functional properties or due to a timer the service can update its values on the registry.
3. The QoS Monitoring reminds the service to update its values.

Fig. 19: Registration by the Provider in a push mode
Fig. 24: Update Service registration after a modification or because of a timer

In this figure 2, the service updates its own new non-functional properties values in the registry component of the Web Directory.

Fig. 25: QoS Monitoring requesting a Service Update

This figure shows the case when the service does not update its non-functional properties values. Then the QoS Monitoring component access the service function or updating the values that also implies the test the service existence.

7.3.4. Calculate the Best Service

The final objective is to find the best Service (or the best services in a composition) responding to a client demand. The client gives the functional properties (or the already calculated composition with the functional properties) in order to get a list of the services to contact in order to retrieve the information/functionalities.

Fig. 26: Complete sequence of retrieving the best functional service

In this Figure 23, the client sends the demands to the Calculate Best Service component. It calculates the best service depending on the client criteria and sends the best service to the client. Then, the client retrieves the desired functionality from the resultant selected service.
7.4. Histograms of the implementation tests

Here we will present all the histograms of the response times measured during the implementation test.

7.4.1. Histograms of the response time depending on the number of non-functional properties

Next, we present all the histograms of the results of the response time for the different number of non-functional properties taken into account in the QoS calculation.

Fig. 27: Histogram of the response time depending on the number of NFP

In the Fig. 27, we see that the response time no follow always a standard deviation. That can be a consequence of the punctual performances of the machine. That explains also the extremes really abroad from the mean value.

7.4.2. Histograms of the response time depending on the number of services

Next, we present all the histograms of the results of the response time for the different number of service taken into account in the QoS calculation.
Fig. 28 : Histogram of the response time depending on the number of services

In the Fig. 28, we see that the response time no follow always a standard deviation. That can be a consequence of the punctual performances of the machine. That explains also the extremes really abroad from the mean value.
8. References

8.1. Biography


15. Hung, Dr. Vladimir Tosi Prof. Patrick C.K. Contract-Based Quality of Service (QoS) Monitoring and Control of XML Web Services.


