A user driven policy selection model

Mariagrazia Fugini, Pierluigi Plebani, and Filippo Ramoni

Politecnico di Milano, Dipartimento di Elettronica e Informazione
[fugini|plebani|ramoni]@elet.polimi.it

Abstract. This paper introduces a model for expressing quality according to both applications and human users perspectives. Such a model, compliant with the WS-Policy framework, not only mediates between the application and human user perspectives, but is also capable of considering the different importance that the user can assign to a quality dimension. In addition, the paper introduces a policy selection model based on the adopted quality model. So a human user expresses its requirements according to a high level language and such requirements are matched against a lower level service quality specification.

1 Introduction

Web service selection plays a crucial role in Service Oriented Computing, since it is responsible for identifying which is the best Web service among a set of available Web services with respect to the user needs. It is worth noting that service selection can be performed by two kind of users: applications and human being with different perspectives. For example, technical parameters such as throughput, bandwidth, latency, framerate could be comprehensible by applications and developers but not by final users: the latter have not skills to define the quality by technical parameter but they need a set of higher level dimensions such as video quality and audio quality defined according to discrete scales such as, for instance, *good*, *average*, and *worst*. The aim of this paper is twofold. On one hand, it introduces a model for expressing the quality of service according to both applications and human users perspectives, also considering the different importance that the user can assign to a given quality dimension. On the other hand, the paper introduces a policy selection model based on the adopted quality model. According to this selection model, a human user can express its requirements according to a high level language and such requirements are matched against to the Web service quality specification expressed through a more technical language. Both models are based on AHP (Analytical Hierarchical Process) developed by T.L. Saaty [6].

2 Quality dimension model

In the literature [1] several quality models have been proposed. In our opinion they are able to express the non-functional properties of a service, but they do not

[1] see http://www.cs.uni-magdeburg.de/∼ rud/wsqos-links.html

deal with the difference between user and service perspective. In the same way, not all the quality dimensions have the same importance and such importance depends on both the application domain and the service users.

The quality of Web service model we propose in this work aims at dealing with this aspect and it is based on three main elements: a quality dimension model, a quality of Web service definition model, and a quality evaluation model.

A *quality dimension*, also known as *quality parameter*, is a non-functional aspect related to an entity that we are describing. Thus, the quality dimension is close to the application domain we are taking into account and it can be directly measured or estimated starting from other dimensions. We identify two classes of quality dimensions, namely, *primitive* and *derived*. A *primitive quality dimensions* (*pqd* hereafter) is a directly measurable quality dimension and it is defined as follows:

$$pqd =< name, values > \tag{1}$$

- *name*: uniquely identifies the quality dimension (e.g., framerate, bandwidth).
- *values*: defines the domain of values of the dimension. The domain can be either continue (e.g., 0..100) or discrete (e.g., high, medium, low).

On the other hand, a *derived quality dimension* (*dqd* hereafter) is not directly measurable but it depends on other quality dimensions:

$$dqd =< name, f(pqd_i, dqd_j) > \quad i = 0..n,\ j = 0..m \tag{2}$$

- *name*: uniquely identifies the quality dimension.
- $f(pqd_i, dqd_j)$: the *dependency function* stating the influence of other quality sub-dimension (both *pqd* or *dqd*). The nature of the function may vary from a simple expression to a composite function.

2.1 Quality of Web Service definition model

Quality of Web service can be defined as the set of quality dimensions which express the non-functional aspects of a Web service. Due to the strong dependency of a quality dimension of the considered application domain, in our quality model we include an actor called *quality designer* that is in charge of collecting and organizing the relevant quality dimensions. Since the quality designer is a domain expert, he is also able to state if a quality dimension is primitive or derived. As stated above a *dqd* depends on both *pqd*s and *dqd*s, thus the work of the quality designer results in a tree named *quality tree* (QT) (see Figure 1):

$$\begin{aligned} QT &= < dqd_{QoS}, tree\_node_k > \quad k = 1..p \\ tree\_node &= [< pqd > | < dqd, v(pqd_i, dqd_j), w(pqd_i, dqd_j) >] \\ &\quad i \leq n, j \leq m, domain(f_{dqd}) \supseteq domain(g) = domain(w) \end{aligned} \tag{3}$$

A QT refers to a given application domain and it includes and organizes all the relevant quality dimensions identified by the quality designer. Given a class

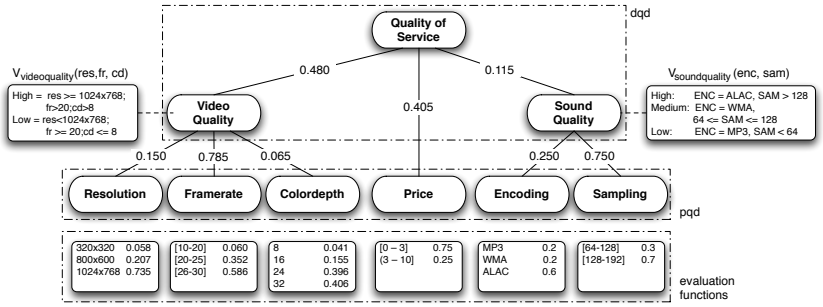

Fig. 1. Quality tree for video-on-demand Web services

of Web services (e.g.: video-on-demand, flight booking), both service providers and users will rely on the related QT tree to describe the offered and desired quality. So, the tree offers a common knowledge for reasoning about quality.

About the structure of a QT, the root is a *dqd* named *QoS*, leaves refers to *pqd*s, and internal nodes are *dqd*s. The function $v(pqd_i, dqd_j)$ derives from $f(pqd_i, dqd_j)$ and returns the value of a *dqd* with respect to the quality dimensions which the *dqd* depends on. The domain of f might contain the domain of v since the quality designer can decide to include in the quality tree only some of the dependent quality dimensions which usually define a given *dqd*.

In addition to the function v, a tree node is also specified by a weigth function w expressing the importance of the quality dimensions which the *dqd* depends on: the higher the weight value, the higher the importance of the quality dimension. The weight assignment is a quite critical activity and we decide to adopt the *AHP (Analytic Hierarchy Process)* approach, developed by T.L. Saaty [6], to perform such an activity. This is a decision-making technique that assigns to each sub-dimension a score that represents the overall performance with respect to the different parameters. AHP is suitable for hierarchical structure as QT and proposes to user some pairwise comparisons between sub-dimensions.

According to this approach, given a *dqd* the quality designer should fill tables like the one shown in Table 1. The first column and the first row are populated with the name of the sub-dimensions influencing the given *dqd*. For each cell, the quality designer assigns a number in $[\frac{1}{9}..9]$ range according to the meaning defined in Table 2 which is the usually adopted one in AHP. About our example, the eigenvector of the matrix in Table 1 is $\{0.150; 0.785; 0.065\}$. This motivates the values reported in QT of Figure 1.

2.2 Quality evaluation model

In some case, e.g., bandwidth, lower value means lower quality; in some other cases, e.g., latency, higher value means lower quality. For this reason, nearby the QT, the quality designer also defines, for each quality dimensions in QT, an

	Res	FR	CD
Resolution	1	$\frac{1}{7}$	3
Framerate	7	1	9
ColorDepth	$\frac{1}{3}$	$\frac{1}{9}$	1

Table 1. Comparison Matrix for VideoQuality dimension

a_{ij}	Definition
1	Equal importance
3	Moderate importance
5	Essential or strong importance
7	Demonstrated importance
9	Extreme importance
2, 4, 6, 8	Intermediate values (compromise)

Table 2. The Saaty Pairwise Combination Scale

evaluation function which captures the quality trend with respect to the quality dimension value.

The evaluation function has different forms with respect to the kind of quality dimension. In case of *pqd*, the evaluation function – $e_{pqd}(values)$ – is a punctual function required to state how a quality value is close or far to the best quality value. Such values can be obtained exploiting the AHP approach.

In case of *dqd*, the evaluation function – $e_{pqd}(QT, pqd_i, dqd_j)$ – is a linear combination of the quality dimensions which influence such a *dqd* according to the considered QT (i.e., $domain(e) = domain(v_{pqd})$). In particular, since the influencing quality dimensions can be both primitive or derived, the evaluation function of a *dqd* will be:

$$e_{dqd}(QT, pqd_i, dqd_j) = \sum_{i=0..n} e_{pqd}(pqd_i.values) * w(pqd_i) + \\ + \sum_{j=0..m} e_{dqd}(QT, domain(g_{dqd_i})) * w(dqd_i) \quad (4)$$

2.3 Policy model

A policy is a document stating the requirements or the offering of a Web service. Following the quality dimension model, a policy document collects a set of relevant quality dimensions included in a QT and defines the admissible values for each of them. According to WS-Policy specification and the model introduced in [5], a policy P can be defined by a set of mutually exclusive alternatives A:

$$P(QT) = \bigoplus_{k=1..l} A_k(QT) \quad (5)$$

where an alternative A is defined as a set of assertions a:

$$A_k(QT) = \bigwedge_{pqd_i \in QT} a(pqd_i) \quad (6)$$

An assertion $a(pqd)$ is a specialization of a quality dimension with a restricted admissible value set, i.e., $a(pqd) =< pqd.name, values \subseteq pqd.values >$.

At the provider side, we have a service policy document $SP(QT)$ (SP hereafter) which specifies the quality of service with respect to several configurations, i.e. alternatives. At user side we have both a user policy document $UP(QT)$ and the user quality tree (UQT), a version of QT customized by the user.

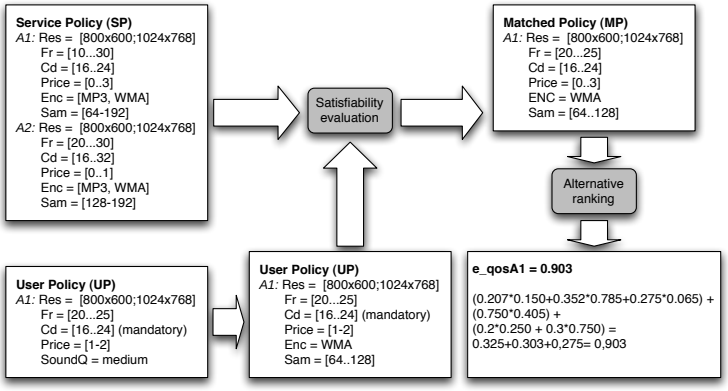

Fig. 2. Example of SP, UP and quality evaluation

3 Policy selection model

Given a user request and a set of Web service policies, the policy selection is in charge of figuring out the best Web service policy with respect to the user preferences. The policy selection considers Web services of the same type so, the related quality tree (QT) can be obtained by the quality designer for the given application domain.

The selection process starts when an input of type SP, UP and UQT [2] is received. The process output is a revised policy (RP) where the alternatives included in SP are sorted from the alternative which best matches the user requirements to the worst one.

The selection process is composed by two main steps: *Satisfiability evaluation* and *Alternative ranking*. Figure 2 exemplifies the process considering the video-on-demand scenario.

3.1 Satisfiability evaluation

Given a $A_i \in SP$, the satisfiability evaluation aims at stating if

$$\forall u_j \in UP, \exists s_i \in A_i \mid s_i \; satisfies \; u_j \tag{7}$$

The operator $satisfies$ considers both the name and the values of a quality dimensions. About the former:

$$s_i \quad satisfies \quad u_j \Rightarrow s_i.name = u_j.name \tag{8}$$

[2] For the sake of simplicity, we only describe the matching between a UP and a single SP where the latter expresses several configurations. In addition, we assume that the user does not modify the QT so $UQT = QT$. The general scenario where a set of SPs are considered and $UQT \neq QT$ can be simply derived.

Roughly speaking, during this activity, the selection process verifies that, for all the service requests expressed in UP, there exists at least one of the service offering assertions which satisfies such a request. This means that all the quality dimensions included in UP must be included in SP as well. If at least one of the quality dimensions in UP is not satisfied, then the process considers the alternative A_i not compliant with respect to the user request.

At the opposite, it might happen that a quality dimension included in the SP could not be considered in the UP. In this case, the process continues since the user might be unaware about a quality dimension that the Web service offers.

The operator $satisfies$ also considers the values describing an assertion. About this analysis, we first need to distinguish between mandatory and non-mandatory value ranges. If a value range is mandatory then the following definition holds:

$$s_i \quad satisfies \quad u_j \Rightarrow s_i.value \supseteq u_j.value \tag{9}$$

Instead, if a user defines a non-mandatory value range then:

$$s_i \quad satisfies \quad u_j \Rightarrow s_i.value \cap u_j.value \neq \emptyset \tag{10}$$

The satisfiability evaluation results in a MP (matched policy), a revised version of SP where the structure remains the same of SP and the value ranges are redefined according to the user request:

$$\begin{aligned} MP = \bigoplus A_m \mid & (\forall A_m, \exists A_k \in SP \mid \\ & (\forall s_i \in A_k,\ \exists m_i \in A_m \mid \\ & (m_i.name = s_i.name = u_j.name \wedge \\ & m_i.values = s_i.values \cap u_j.values))) \end{aligned} \tag{11}$$

3.2 Alternative ranking

The second step of the selection process has to sort the alternatives included in MP taking into account the importance assigned by the user to the quality dimensions. So, the inputs of the alternative ranking phase are both MP and UQT whereas the output is the final policy RP (Ranked Policy).

Similarly to what done during the satisfiability evaluation, RP has the same structure of MP and is defined as follows:

$$\begin{aligned} RP = \bigoplus A_r \mid & (\forall A_r, \exists A_m \in MP | A_r = A_m) \wedge \\ & (\forall A_i, A_j \in RP, i < j \Rightarrow \\ & e_{qosA_i}(UQT, domain(g_{qosA_i})) \geq e_{qosA_j}(UQT, domain(g_{qosA_j})) \end{aligned} \tag{12}$$

The quality of an alternative that we need to rank is calculating using the evaluation functions of the assertions composing the alternatives. An alternative in MP, in fact, is expressed in terms of assertions related to *pqd* which, in turns, are the leaves of the quality tree associated to the alternative as well. Actually,

during the quality calculation we do not consider the original quality tree but UQT, i.e., the version that the user customizes.

$$quality(A, UQT) = e_{qos}(pqd_i) \;\; pqd_i.name \in UQT = a_k.name \in A \qquad (13)$$

4 Concluding remarks

In this work we have proposed an approach for selecting Web services by analyzing the offered quality. A quality definition and evaluation model are introduced to allow both Web service providers and users to specify, namely, the offered and desired quality. Such models also deal with the different levels of details in expressing quality by these two actors.

Based on this model, the selection process we propose is capable of automatically matching user and provider policies and ranking several quality alternatives to identify the best Web service. A prototype implementing our approach is under development.

Comparable approaches are given by WSOL [7] and WSLA [2], which provide some description model which our work can use to express *pqd*. Focusing on the selection process, in [4] the dynamics selection of the services is discussed proposing a solution based on agents, using the Web Services Agent Framework (WSAF), that includes an ontology for the QoS and a *ad-hoc* language to specify quality. The proposed approach only evaluate services with feedback assigned from the user that have already used the service, and does not consider the actual users' needs. In [3], the proposed utility theory uses utility functions to estimate every quality parameter without any focus on the dynamic creation and personalization of the dimensions. In our work we have adopted *WS-Policy* as policy language; other languages, such as *Features and Properties (F&P)*, are also available. Different work, like [1], show the substantially equivalence between the two languages, finding the differences at the syntax level.

References

1. G. Daniels. Comparing Features & Properties and WS-Policy. *W3C Workshop on Constraints and Capabilities for Web Services*, 2004.
2. A. Keller and H. Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Technical Report RC22456(W0205-171), IBM Research Division, T.J. Watson Research Center, May 2002.
3. S. Lamparter and S. Agarwal. Specification of Policies for Web Service Negotiations. *Semantic Web and Policy Workshop, Galway*, November 2005.
4. E. M. Maximilien and M. P. Singh. A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, September-October 2004.
5. T. Mikalsen, N. K. Mukhi, P. Plebani, and I. Silva-Lepe. Supporting Policy-driven behaviors in Web services: Experiences and Issues. *ICSOC-04*, 2004.
6. T. L. Saaty. *The Analytic Hierarchy Process*. Mc Graw Hill, New York, 1980.
7. V. Tosic, K. Patel, and B. Pagurek. WSOL - Web Service Offerings Language. In *Web Services, E-Business and the Semantic Web, CAiSE 2002 International Workshop (WES 2002)*, Toronto, Canada, May 2002.