Towards a Dynamic Negotiation Mechanism for QoS-Aware Service Markets

Claudia Di Napoli, Paolo Pisa, and Silvia Rossi

Abstract. The market value of commercial Service-Based Applications (SBAs) will depend not only on their functionality, but also on the value of QoS parameters referring to its not functional properties. These parameters are not static properties since they may vary according to the provision strategies of providers as well as the demand of users having their own preferences on the application's QoS values. In this paper we propose a negotiation-based mechanism among service providers and a user requesting a QoS-aware SBA to select services with suitable QoS values, i.e. values that once aggregated satisfy the user's requirements. The proposed mechanism simulates a market-based provision mechanism that allows to take into account the variability of service QoS attribute values typical of the future market of services, as well as to dynamically set the length of the negotiation process that is usually very time consuming.

Keywords: Market-based negotiation, Quality of Service, Service Selection.

1 Introduction

Service Based Applications (SBAs) are composed of autonomous and independent services each one provided with Quality of Service (QoS) attributes that take account of its non-functional properties (NFPs) such as cost, execution time, reliability, reputation, and so on [7]. In the future market of services QoS-aware SBAs

Claudia Di Napoli
Istituto di Cibernetica "E. Caianiello" - C.N.R., Via Campi Flegrei 34,
80078 Pozzuoli, Naples, Italy
e-mail: c.dinapoli@cib.na.cnr.it

Paolo Pisa · Silvia Rossi
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione,
University of Naples "Federico II" , Napoli, Italy
e-mail: silvia.rossi@unina.it

J.B. Pérez et al. (Eds.): *Trends in Prac. Appl. of Agents & Multiagent Syst.*, AISC 221, pp. 9–16.
DOI: 10.1007/978-3-319-00563-8_2

will be required by users that have their own preferences over the values of these attributes that may change in time according to dynamic circumstances. In order for QoS-aware SBAs to be delivered, the attribute values of their component services, once aggregated, have to meet the user requirements. In this context, it becomes crucial to provide service-oriented infrastructures with mechanisms enabling the selection of services with suitable QoS attribute values allowing to manage the dynamic nature of both QoS values, and QoS requirements.

In this paper we propose a negotiation-based mechanism among service providers and a service consumer to select the suitable services to compose QoS-aware SBAs through a market-based provision mechanism. The use of a negotiation-based mechanism allows to take into account the variability of service QoS attribute values typical of the future market of services since service providers may change these values during the negotiation according to their own provision strategies. For mandatory (i.e. not negotiable) QoS values the selection is obtained by solving a constraint satisfaction problem. Since negotiation can be computationally expensive, a set of experimental results were carried out to determine the parameters affecting the negotiation process to extract useful information to drive service consumers decisions about whether to proceed with the negotiation, under specific conditions, or not.

2 Selecting Services through Automated Negotiation

In this work it is assumed that a request for an SBA is expressed by a directed acyclic graph, called an *Abstract Workflow* (AW), specifying the functionality of each service component, and their functional dependence constraints, together with a quality attribute value representing the *QoS* required by the user for the application. AW nodes represent the required functionalities, referred to as *Abstract Service*s (ASs), and AW arcs represent control and data dependencies among nodes. For each AS, a set of *Concrete Services* (CS) may be available on the market, each one provided by a specific *Service Provider* (SP) with QoS attributes whose values are set by the corresponding SP dynamically according to its market provision strategies. The user request is managed by a *Service Compositor* (SC), responsible for the selection of the CSs whose attribute values, once aggregated, satisfy the QoS required by the user. Both SPs and SC are represented by software agents able to negotiate.

The selection process is modelled as a negotiation process over the service quality attributes occurring among the SC and the SPs, available to provide the required services, that populate the multi-agent system. SPs issue their offers to the SC by specifying a reference to the CS together with the value of the QoS attribute they can provide the service with at that time. If the negotiation is successful, then the AW can be instantiated with the CSs having the suitable QoS value, and the *Instantiated Workflow* (IW) represents the requested application ready to be executed.

In the proposed negotiation mechanism only SPs formulate new offers, and only the SC evaluates them. The rationale of this choice is twofold: on one hand it makes it possible to simulate what happens in a real market of services where an SC does not have enough information on the SPs strategies to formulate counteroffers; on

the other hand it takes into account that the offers for a single functionality cannot be evaluated independently from the ones received for the other functionalities, i.e., negotiating over the attributes of the single AS cannot be done independently from each other. So, the negotiation mechanism should allow to negotiate with the SPs, and at the same time to evaluate the aggregated QoS value of the received offers for all the required functionality in the AW during the negotiation.

In order to meet these requirements, an iterative negotiation protocol [3], based on a Contract Net Iterated Protocol, is adopted. The negotiation occurs between the SC and the SPs available for each AS of the AW, and it may be iterated for a variable number of times until a *deadline* is reached or the negotiation is successful. Each iteration is referred to as a negotiation *round*, and the deadline is the number of allowed rounds. According to the protocol, at each negotiation round the SC sends $m * n$ call for proposals, where m is the number of ASs in the AW, and n is the number of SPs available to take part in the negotiation for each AS, and after waiting for the time set to receive offers (know as the *expiration time*), it checks first that there are offers for each AS; if not the SC declares a failure since it is not possible to instantiate the AW. Otherwise, it evaluates the received offers, and, according to the result of the evaluation (see Sec. 2.1), it starts another negotiation round, or it selects the best offers in terms of its own utility.

2.1 The Negotiation Evaluation

The SC evaluates the offers received at each negotiation round to check whether the global constraints specified by the user are met by using a solver of a Linear Programming problem so formulated. There are nm decision variables $x_{i,j}$ where i identifies one of the m ASs and j identifies one of the n SPs compatible with the i_{th} AS. The variable $x_{i,j}$ is equal to 1 if the j_{th} SP is selected for the i_{th} AS, 0 otherwise. Since only one SP has to be selected for each AS, then $\sum_{j=1}^{n} x_{i,j} = 1$ for all ASs.

In the general case of a multidimensional QoS $(Q_1, \ldots, Q_r)$, the n-tuple $(q_{i,j}^1, \ldots, q_{i,j}^r)$ of offered values is associated to each corresponding SP identified by the decision variable $x_{i,j}$. To check whether each QoS constraint is satisfied, taking into account all the ASs in the workflow, the aggregated values of the parameters $q_{i,j}^k$ offered by each selected SP must not exceed the user upper bound Q_k, i.e.:

$$aggrFunc_{i,k}(\sum_{j=1}^{n} x_{i,j} q_{i,j}^k) \leq Q_k, \forall k = 1, \ldots, r \tag{1}$$

where *aggrFunc* depends on the type of the considered parameter. Typically, in the literature additive (e.g., price and execution time) and multiplicative (e.g., reliability and availability) parameters are studied [8], so *aggrFunc* is either a summation or a multiplication over the number of ASs. Once solutions that satisfy the constraints are found, the SC evaluates their utility with the formula [1]:

$$U_{i,j}(SC) = \sum_{k=1}^{r} \frac{Q_{max(i,k)} - q_{i,j}^k}{Q_{max'(k)} - Q_{min'(k)}} \tag{2}$$

where $Q_{max(i,k)} = max(q^k_{i,j})$, $Q_{max'(k)} = aggrFunc_k(Q_{max(i,k)})$ aggregating the local maxima of the offers received for each AS, and $Q_{min'(k)} = aggrFunc_k(Q_{min(i,k)})$, aggregating the corresponding local minima.

Eq. 2 evaluates the SC's utility of an offered QoS value w.r.t. both the ones offered by the other SPs of the same service (local evaluation), and the QoS value of a possible instantiated workflow (global evaluation). In fact, $Q_{max(i,k)} - q^k_{i,j}$ gives an indication of how good the value of each QoS parameter is with respect to the QoS values offered by other SPs of the same AS (local evaluation) by taking as a reference the maximum offered value for that parameter. Local evaluation compared to $Q_{max'(k)} - Q_{min'(k)}$ gives an indication of how good the value of each parameter is with respect to the possible aggregated values of the same parameter for all the ASs (global evaluation).

The SC objective function is a maximization of the sum of the utilities for each $m-tuple$ of selected SPs that satisfies the QoS global constraints given by Eq.3:

$$max(\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j}U_{i,j}(SC)) \tag{3}$$

2.2 The Negotiation Strategy

SPs strategies are modeled as a set of functions that are both time and resource dependent [4] taking into account both the *computational load* of the provider, and the *cost* of the provided service. The computational load accounts for the provider workload, i.e., the amount of service implementations it will deliver, while the cost of the service is directly proportional to its complexity.

For each SP the negotiation strategy is modeled by a Gaussian distribution that represents the probability distribution of the offers in terms of the provider's utility. As shown in Fig. 1, the mean value of the Gaussian $maxU$ represents the best offer the SP may propose in terms of its own utility with the highest probability to be selected; while the standard deviation σ represents the attitude of the SP to concede during negotiation and it is given by $\sigma_{i,j} = maxU_{i,j} - maxU_{i,j}percent_{i,j}$, where $percent \in [0,1]$ represents the concession percentage of the SP with respect to its own utility. The parameter σ varies from SP to SP providing the same AS, so that the lower its computational load (in terms of available resources) is, the more it is available to concede in utility and the lower its reservation value is. The negotiation set for the SP is $[maxU - \sigma; maxU]$, where $maxU - \sigma$ is the reservation value.

In Fig. 1 the functions associated to two different SPs for the same AS are reported. The best offer is the same for both SPs (i.e. $maxU_1 = maxU_2$) since it is assumed that services providing the same functionality have the same utility value, while their concession strategies are different according to their workload when the negotiation takes place. In fact, σ_1 is greater than σ_2 meaning that SP_1 has a lower computational load than SP_2, so it concedes more in utility than SP_2.

At each negotiation round, the SP generates a new utility value corresponding to a new offer according to its Gaussian distribution (for values generated in the set

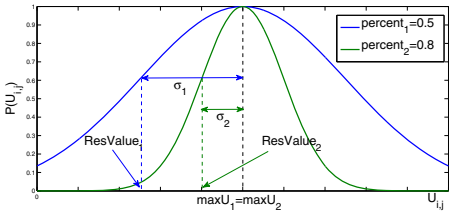


Fig. 1 An example of probability functions to compute new offers

$[maxU; maxU + \sigma]$, the values within the negotiation set $[maxU - \sigma; maxU]$ with the same probability to be selected, are considered): if this value is lower than the one offered in the previous round and within the negotiation set, then the SP proposes the new value; if this value is higher than the one offered in the previous round, or it is outside the negotiation set, the SP proposes the same value offered in the previous round. This strategy allows to simulate SPs that prefer not having a consistent loss in utility, even though by increasing the number of negotiation rounds the probability for the SP to move towards its reservation value increases.

3 A Cost-Based Testbed

A preliminary set of experiments was carried out in order to determine the main factors affecting the negotiation success/failure, and to evaluate the impact of the SPs negotiation strategies on the negotiation progress. The experiments were aimed at verifying the possibility to extract information that can be used by the SC to decide whether to iterate or to stop the negotiation according to the current situation.

In these experiments, the QoS attribute is the service price, so the QoS value of the requested application is additive in the number of ASs, and it does not depend on the structure of the AW. The utility value for the SP represents the price for the service it offers, so *maxU* is the highest price (*bestPrice*) offered by the SP, and it is the same for all SPs of the same AS. An SP offer is $Price_{i,j}$, and the reservation price for the SP is $bestPrice - \sigma$. It is assumed that the more complex a service functionality is, the higher its "market price" is, i.e. the variability in prices for different ASs is proportional only to their complexity. To simulate this variability, a parameter k is used: the more complex the functionality provided by a service is the higher the k value is. In particular, for SPs providing less complex services $k \leq 1$, while for SPs providing more complex services $k > 1$. The k parameter is equal for all SPs of the same AS, meaning that services providing the same functionality have the same market price. In fact, k determines the mean value of the Gaussian distribution, and so the *bestPrice* for an ASi is:

$$bestPrice_i = \frac{globalPrice \;\; k_i}{m} \quad i \in [1, \ldots, m] \tag{4}$$

where, m is the number of ASs in the AW. So, Eq. 4 takes into account both the computational cost of the offered service, and also the assumption that the requested

price *globalPrice* is not "unreasonable" compared to the market price of the required ASs in the AW. A *feasible* solution exists if the QoS constraint is satisfied, i.e. $\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j}Price_{i,j} \leq globalPrice$.

If for each AS $k > 1$, then the QoS constraint is never satisfied at the first round.

3.1 Experimental Results

The impact of the σ parameter is evaluated considering a configuration with k and the number of AS (i.e. provider agents) fixed. The default price is assigned to each *ASi* according to Eq.4, with $m = 5$ and $k_i = 2.4$, 2.0, 1.3, 1.0 and 0.8 (average value of $k = 1.5$), approximately corresponding to 32%, 27%, 17%, 13% and 11% of the *globalPrice*. The corresponding SPs send as initial offer a price in the neighborhood of $bestPrice_i$ $[bestPrice_i - 5\%, bestPrice_i]$. The *percent* value randomly varies for each SP in the range [0.5, 1.0], so including the possibility to have SPs with the maximum computational load, not willing to concede (i.e., *percent* = 1). The maximum number of negotiation rounds is 100. In the case the SPs of AS3, AS4 and AS5 are not willing to negotiate, the negotiation always fails even with the minimum *percent* value for AS1 and AS2. Tables 1a, 1b, 1c, 1d report the average of the minimum *percent* values for each AS with respectively 2, 4, 8, 16 SPs.

Table 1 Minimum *percent* with respectively a) 2 SPs, b) 4 SPs, c) 8 SPs, d) 16 SPs for each AS

a)

Succ/Fail	%	AS1	AS2	AS3	AS4	AS5
successes	61	0.61	0.63	0.64	0.65	0.65
failures	39	0.74	0.74	0.74	0.70	0.69
successes	8	0.56	0.56	0.57	1	1
failures	92	0.67	0.68	0.68	1	1

b)

Succ/Fail	%	AS1	AS2	AS3	AS4	AS5
successes	95	0.59	0.60	0.61	0.61	0.60
failures	5	0.72	0.65	0.70	0.72	0.61
successes	35	0.54	0.56	0.57	1	1
failures	65	0.65	0.62	0.61	1	1

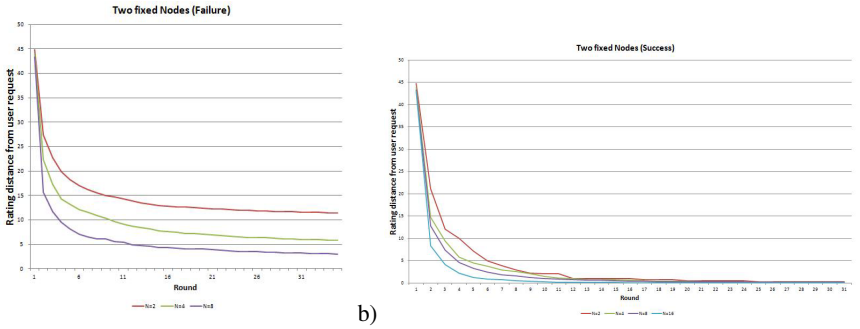
c)

Succ/Fail	%	AS1	AS2	AS3	AS4	AS5
successes	100	0.55	0.55	0.54	0.56	0.55
failures	0	-	-	-	-	-
successes	77	0.54	0.55	0.55	1	1
failures	23	0.60	0.61	0.59	1	1

d)

Succ/Fail	%	AS1	AS2	AS3	AS4	AS5
successes	100	0.53	0.53	0.52	0.52	0.52
failures	0	-	-	-	-	-
successes	100	0.52	0.53	0.53	1	1
failures	0	-	-	-	-	-

As expected, with SPs for AS4 and AS5 not willing to negotiate, the minimum *percent* values for the providers of the remaining ASs have to be lower than the ones obtained for configurations where all the SPs are available to make concessions. In fact, for successful negotiations, the average minimum value of *percent* is 0.55, while with an average minimum value of *percent* = 0.63 negotiation failures are obtained (third and fourth rows in tables 1a, 1b and 1c). The only exception is table 1d) where, increasing the number of SPs to 16 for each AS, a 100% rate of successes is obtained with an average minimum value of *percent* equal to 0.53. Moreover, in the case of the providers for AS4 and AS5 not willing to negotiate, there are more failures than successes in tables 1a and 1b, while more successes



a) b)

Fig. 2 Distance in case of failures (a) and successes (b)

than failures in table 1c. So, by increasing the number of SPs, the chances to succeed in the negotiation increase. In the case all SPs are willing to negotiate (first two rows in the tables) more successes than failures are obtained with smaller medium minimum values of *percent*. Finally, the higher the computational cost of the AS is, the smaller the corresponding average minimum value of *percent* is to have a success. This shows that the SC should negotiate essentially with SPs providing higher computational cost services since they impact more the negotiation success/failure.

In Fig. 2 the distance of the price obtained at each negotiation round from *globalPrice* is reported varying the number of available SPs given by:

$$\frac{((\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} Price_{i,j}) - globalPrice)100}{globalPrice} \tag{5}$$

In Fig. 2a the distance for the failure cases is plotted showing that the curve trend is the same varying the number of SPs. After the 25th round, the distance from the required price varies very little (0.1%) and there is a failure at the 100th round. This information can be used at runtime to dynamically set the negotiation deadline (e.g. in our experiments it can be 30 rounds). In Fig. 2b the distance for the success cases is plotted, and it shows that negotiation always ends before the 30th round.

4 Conclusions

Several efforts have been carried out in the areas of QoS-based service selection for SBAs. Some works propose algorithms to select service implementations relying on the optimization of a weighted sum of global QoS parameters as in [8] by using integer linear programming methods. In [2] local constraints are included in the linear programming model used to satisfy global QoS constraints. In [1] Mixed Integer Programming is adopted to find the optimal decomposition of global QoS constraints into local constraints so the best services satisfying the local constraints can be found. Typically, these works rely on static approaches assuming that QoS parameters of each service do not change during the selection process. Other

approaches rely on negotiation mechanisms to select services according the QoS values [5, 6], but usually negotiation is carried out for each required service independently from the others. Attempts to propose a coordinated negotiation with all the providers of the different service in a composition have been proposed [3], but not empirical evaluation is provided to show that if a solution exists it is found.

This work proposes an approach for QoS-based service selection that takes into account the variability of service providers provision strategy. The use of an iterative negotiation mechanism allows to address the limitations of assuming static QoS values that is not realistic in market-based service scenarios, since providers might change dynamically their provision strategies according to market trends during service selection to become more competitive in the market during negotiation. For this reason, the service compositor negotiates with all available providers so to not discharge providers that may become more competitive during negotiation. The proposed mechanism allows to evaluate the progress of the negotiation, so it can be stopped if the SC utility is not improving. This feature is useful in service-based application settings since negotiation is computationally expensive. Furthermore, aggregated offers are evaluated at each negotiation round since the selection of one service cannot be done independently from the other services. This is even more crucial in case of multidimensional QoSs.

References

1. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, pp. 881–890. ACM, New York (2009), http://doi.acm.org/10.1145/1526709.1526828, doi:10.1145/1526709.1526828
2. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. on Software Eng. 33(6), 369–384 (2007)
3. Di Napoli, C.: Using software agent negotiation for service selection. In: Mele, F., Ramella, G., Santillo, S., Ventriglia, F. (eds.) BVAI 2007. LNCS, vol. 4729, pp. 480–489. Springer, Heidelberg (2007)
4. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation Decision Functions for Autonomous Agents. International Journal of Robotics and Autonomous Systems 24, 3–4 (1998)
5. Paurobally, S., Tamma, S., Wooldridge, M.: A framework for web service negotiation. ACM Trans. on Autonomous and Adaptive Systems 2(4) (2007)
6. Siala, F., Ghedira, K.: A multi-agent selection of web service providers driven by composite qos. In: IEEE Symposium on Computers and Communications, pp. 55 –60 (2011)
7. Strunk, A.: Qos-aware service composition: A survey. In: Proceedings of IEEE 8th European Conference on Web Services, pp. 67–74. IEEE Computer Society (2010)
8. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. on Soft. Eng. 30(5), 311–327 (2004)