Evaluating Negotiation Cost for QoS-aware Service Composition

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

Dario Di Nocera*
Dipartimento di Matematica e Applicazioni,
Università degli Studi di Napoli
"Federico II", via Cintia MSA,
80126 Napoli, Italy
dario.dinocera@unina.it

Silvia Rossi
Dipartimento di Ingegneria Elettrica
e Tecnologie dell'Informazione,
Università degli Studi di Napoli
"Federico II", via Claudio 21,
80125 Napoli, Italy
silvia.rossi@unina.it

Abstract—**The value of commercial Service-Based Applications (SBAs) will depend not only on their functionality, but also on the value of their non-functional properties, known as QoS attributes, that are not tied to a specific functionality, but rather to its delivery features. QoS values may vary according to the provision strategies of providers as well as users' requirements expressed as global constraints on the SBA QoS. Automatic negotiation is a viable approach to drive QoS-aware selection of services for SBAs, but its adoption may result computationally expensive due to the communication overhead among the involved negotiators, so limiting its application to real service-based scenarios. In this paper, an empirical evaluation of the impact of negotiation communication costs occurred when composing services to deliver a QoS-aware SBA is carried out, in order to estimate the advantages and disadvantages of negotiation in a market of services, and to identify negotiation parameters settings for which communication costs can be compensated by an increased probability for the negotiation to succeed.**

I. Introduction

The increased popularity of the Service-Oriented-Computing (SOC) paradigm [1] is enhancing the development of Service Based Applications (SBAs) that are distributed applications obtained by combining existing services in a loosely coupled manner that collectively fulfill a requested task. Services are independent and autonomous entities provided by different service providers to be consumed by users requesting a given application. Users do not need to be aware of the actual composition as long as the functional and non-functional requirements are satisfied [2]. The consumption of these services is commonly governed by an agreement among providers and consumers, known as Service Level Agreement (SLA), which regulates terms and conditions of service provision [3]. Agreements on service provisioning may include not only the provider's commitment to execute a given task (coming from functional requirements), but they also include terms about performance levels (or quality levels) of services [4] (coming from non-functional requirements). Service non-functional requirements refer to service attributes known as Quality of Service (QoS) attributes that play an important role in service selection.

In a previous paper [5] we proposed a market-based negotiation mechanism among service providers and a service consumer to select the suitable services to compose QoS-aware SBAs. The approach allows for the selection of services according to the values of their quality attributes that, once aggregated, have to meet end-to-end user QoS constraints/preferences. The 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, and market trends.

The negotiation protocol is designed as a *one-to-many-to-many* iterative protocol since the service consumer negotiates at the same time with the different providers of each functionality required in the SBA, as well as with the providers of the different functionalities required in the SBA in a coordinated way. In fact, when dealing with end-to-end QoS requirements typical of SBAs, the QoS values of each functionality are not independent from one another, but it is necessary to find a set of interrelated QoS values. So, the negotiation process may require several iterations to successfully end, becoming computationally expensive in terms of the involved communication costs.

In this paper we propose an experimental evaluation of the proposed negotiation mechanism in terms of its computational costs due to the communication overhead coming from the possibility to negotiate with all the available providers during each iteration of the negotiation. The aim of the evaluation is to compare the cost of negotiation with respect to the potential benefit of having a success at the end of the process, and to evaluate the pros and cons in negotiating with all available providers until the negotiation ends.

The paper is organized as follows. In Section II some works related to service composition are reported; Section III describes the main features of the negotiation mechanism adopted in the present work; in Section IV the rationale of the experiments set to evaluate the cost of the negotiation protocol is reported, together with the decision making mechanisms adopted by the service compositor and the service providers during the negotiation. Then the experiments carried out, and the evaluation of the obtained results are described and discussed in Subsections IV-B, and IV-C. Finally, Section V reports some conclusions and planned future works.

* Ph.D. scholarship funded by Media Motive S.r.l, POR Campania FSE 2007-2013.

II. Related Works and Background

Service composition allows to aggregate autonomous and independently developed services in order to build added value applications (SBAs). With the pervasive growth of available services, it is widely recognized that multiple services providing the same functionality may be available, but they may differ in their non-functional features, usually known as Quality of Service, such as cost, execution time, and so on. In this context, users will require SBAs specifying their own preferences/constraints on the global QoS values of the application, so it becomes crucial to select the appropriate component services, i.e., services whose QoS attribute values, once aggregated, meet users' preferences/constraints.

Several research efforts addressed this challenge proposing approaches that mainly apply static methods to find the set of services whose QoS values meet the global constraints set by the users. Some works propose algorithms to select service implementations relying on the optimization of a weighted sum of global QoS parameters as in [6] by using integer linear programming methods. In [7] local constraints are included in the linear programming model used to satisfy global QoS constraints. In [8] Mixed Integer Programming is used to find the optimal decomposition of global QoS constraints into local constraints so that the best services satisfying the local constraints can be found. Typically, these works rely on static approaches assuming that QoS values of each service are predefined by providers and do not change during the selection process.

In dynamic markets where service provision is regulated by demand and supply mechanisms, it is likely that different users may have different QoS requirements for the same (from a functional point of view) application, as well as QoS attribute values for the same service may change in time according to dynamic circumstances affecting service provision strategies. In this context, it becomes crucial to provide service-oriented infrastructures with mechanisms enabling the selection of services with suitable QoS attribute values so that QoS requirements can be satisfied when forming new added-value applications through service composition. Such mechanisms should allow to manage the dynamic nature of both QoS values, and QoS requirements. Negotiation has gained more and more attention in SOC applications as a viable approach to drive the selection of suitable component services. It allows to address the dynamic nature of both the provided and required QoS since the offered QoS value of single services may change as soon as new offers and counteroffers are exchanged.

Practical negotiation mechanisms for B2B applications must be computationally efficient [9]. This implies that the interaction rules have to guarantee the quick end of the process and that agents behaviors and negotiation strategies should be developed based on the assumption of bounded rather than perfect rationality [10]. One of the common requirements for a negotiation protocol is the monotonicity of the utilities of the offers as in [11]. This allows to guarantee the end of the process without a deadline: either an agreement is reached (sooner or later), or a conflict is reached in the case all agents stop to concede in utility.

Most approaches, that use negotiation mechanisms to select services according to their QoS values, usually apply negotiation for each required service independently from the others relying on bilateral one-to-one negotiation mechanisms [4], [12]. Attempts to propose a coordinated negotiation with all the providers of the different required services in a composition have been proposed as in [2], but they introduce a Negotiation Coordinator that instructs the negotiation of the single component services by decomposing end-to-end QoS into local QoS requirements, so making the negotiation process computationally heavier from the point of view both of the involved negotiators, and of the necessary decision making mechanisms.

III. One-to-Many-to-Many Negotiation Protocol

In this work we adopt the iterated negotiation mechanism proposed in [5], starting from the assumption that SLAs for QoS-aware SBAs have to be set by coordinating the single agreements of each component service.

In the proposed approach a *Service Compositor* (SC), acting on behalf of a service consumer, issues an SBA request represented by a Directed Acyclic Graph, referred to as an *Abstract Workflow* (AW), specifying the functionality of each service component (AW nodes referred to as *Abstract Service*s ASs), and their functional dependence constraints (AW arcs), together with the value(s) of the end-to-end QoS requirements the user wants the application to provide. It is assumed that for each AS a set of *Concrete Service*s (CSs) are 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. The protocol allows only the SPs to formulate new offers, and only the SC to evaluate 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. So, it is necessary to design a negotiation mechanism that allows both to negotiate with the SPs providing services for each required functionality in the AW, but 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. Indeed, the SC is not able to make single counter-proposals with respect to each received offer, because the change of a value of a particular QoS can impact the others QoS attributes of the same service, as well as the constraints to be fulfilled by the QoSs of the other services. In other words, negotiating over the attributes of the single AS cannot be done independently from each other.

Since SC does not provide counteroffers the negotiation could be model as simply an auction mechanism as in [13]. However, in order to model a real market of services, it cannot be assumed that all providers, providing different functionalities, follow the same rules when bidding (such as Vickrey, English, and so on), as it happens in auctions mechanisms. In fact, rules may vary according to the type of provided service (i.e., its functionality), and above all according to the trends of the market that may vary quickly, and not in the same way for all the QoS attributes. With auction mechanisms, each bidder may have its own strategy, but once the type of auction is decided, then all bidders know the rules and they have to stick to them until the auction ends. Moreover, a simple auction mechanism cannot be used in our setting because of the interdependence among the QoS attributes of the component services. In fact, it would not be possible to award an auction winner without evaluating the offers for a given AS with respect to the ones received for the other ASs in the AW. We argue that these solutions do not model what happens in real markets of services where predefined bidding mechanisms cannot be assumed and fixed for all the service types and for all the considered QoS attributes. Therefore, traditional methods with the protocols and strategies hard-coded in the agents would not work in real market of services that are open systems.

This is why an hybrid negotiation approach was used, where an auction-like protocol models the bidding of single component services, but without relying on a specific auction mechanism to allow SPs to adopt their own private strategies when bidding, and also to change them if required by the market trends.

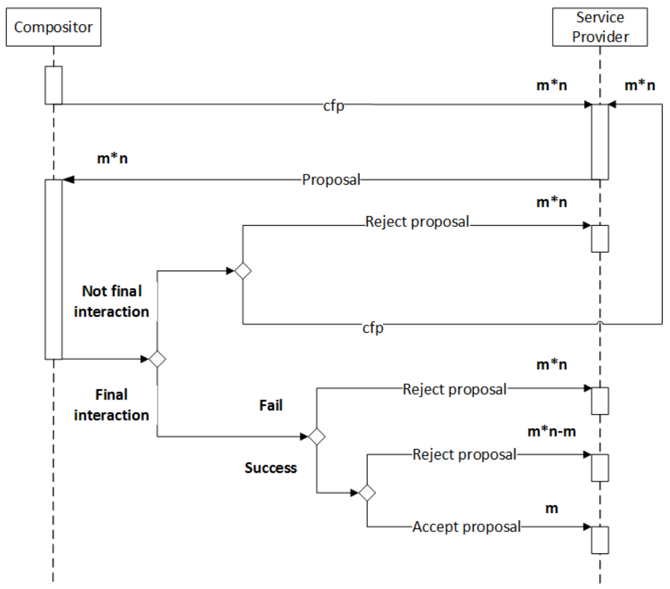

Figure 1. The negotiation protocol.

In [5], we presented a *one-to-many-to-many* protocol (OMM) for the dynamic selection of services, based on the FIPA Contract Net Iterated Protocol [14], [15] (ICNET), one of the most used protocols for negotiating SLAs [4]. As described in Figure 1, the negotiation occurs between the SC, that is the *initiator* of the negotiation, 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. The deadline is the number of allowed rounds. The SC prepares m call for proposals (`cfps`), one for each AS in the AW and, assuming that there are n SPs for each of the m AS, it sends $m * n$ `cfps` at each negotiation round. After waiting for the time set to receive offers, if there are not offers for each AS in the AW, the SC rejects the received proposals (`reject proposal`) since it is not possible to find a CS corresponding to each AS. Otherwise, it evaluates the received offers, and, if the QoS value obtained by aggregating the received offers does not meet the user request, it starts another negotiation round sending $m * n$ `reject proposals`, and, at the same time, new $m*n$ `cfps`. If the QoS values of the received offers, once aggregated meet the user request, it accepts the offers sent by the corresponding SPs (sending m `accept proposals` and $m*n-m$ `reject proposals`). If the deadline is reached without a success, the negotiation ends.

IV. Evaluating the Negotiation Cost

We evaluate experimentally the efficiency of the negotiation mechanism with respect to two performance measures: negotiation outcome (i.e., utility of the solution and/or the negotiation success rate), and communication complexity (i.e., the number of exchanged messages), by varying the parameters affecting the OMM protocol that are the number of allowed negotiation rounds, and the number of SPs involved in the interaction.

A. Compositor and Providers Utility Functions and Strategies

Here, we briefly describe the decision making algorithm for the SC evaluation of proposals, and the strategies adopted by SPs to generate an offer, as proposed in [5], considering the case of a single additive QoS parameter (the parameter considered here is the "price").

Following the approach formulated in [8], the SC first evaluates the utility of the offer provided by the jth SP for the ith AS, with respect to both the other offers for the same AS (local evaluation), and to the entire workflow (global evaluation):

$$U_{SC}(o_{i,j}) = \frac{max_k(price_{i,k}) - price_{i,j}}{\sum_{i=1}^{m}(max_k(price_{i,k}) - min_k(price_{i,k}))} \quad (1)$$

where i identifies one of the m ASs and the j identifies one of the n SPs. Once the most promising offer for each AS is selected according to Eq. 1, we modeled the global requirements satisfaction problem in terms of SC's global utility. This utility is related to the distance between the QoS preferences, expressed at the time the request is issued, and the aggregated QoS values obtained by combining the the best selected offers. It is normalized so that it is 1 in case the requirements are met, and in $[0, 1]$ otherwise. The SC's utility is expressed as follows:

$$U_{SC} = \begin{cases} 1 & if \quad \sum_{i=1}^{m} price_{i,s} < reqPrice \\ 1 - \frac{\sum_{i=1}^{m} price_{i,s} - reqPrice}{reqPrice} & otherwise \end{cases} \quad (2)$$

where, $price_{i,s}$ is the price offered for the ith AS by the selected sth SP, $\sum_{i=1}^{m} price_{i,s}$ is the aggregated value for the price, and $reqPrice$ is the user requested price.

On the contrary, SPs apply their own strategies to formulate their offers. These strategies are modeled as a set of functions that are both time and resource dependent [16], and they take into account both the *computational load* of the provider, and the *computational cost* of the provided service. The computational load of the provider accounts for the number of requests it agreed to fulfill, i.e., the amount of service implementations it will deliver, while the computational cost of the service represents a measure of the complexity of the provided service, i.e., the more complex the service is the higher its expected cost is. SPs strategies to concede in utility are modeled as Gaussian distributions [5]. The mean value of the distribution $maxU$ is the best offer the SP may propose in terms of its own utility and as such it has the highest probability to be selected. The standard deviation σ represents the attitude of the SP to concede during negotiation and it varies from SP to SP providing the same AS, so that the lower the SP's computational load is, the more it is available to concede in utility and the lower its reservation value is. The best offer $maxU$ is the same for all SPs of the same AS. This assumption models a scenario where services providing the same functionality have the same "market price" corresponding to the maximum utility for the SP providing that service. At each negotiation round, the SP generates, following its provision strategy, a new value of utility corresponding to a new offer to be sent to the SC by comparing the new offer with the previously generated one to decide whether to submit it or not. This is a variation of the standard negotiation mechanisms where the utility of a new generated offer is compared with the last one generated by the other negotiation partner.

B. *Experimental Settings*

The configurations considered for the experiments set five different deadlines at 1, 10, 20, 30 and 40 rounds, and for each deadline the number of SPs at 2, 4, 8, 16. Let us highlight that for a deadline of 1 round, the protocol is based on m concurrent ContractNet Protocols (CNET). We also considered a configuration where the negotiation occurs only with one SP for each AS, that is the best SP, in terms of $U_{SC}(o_x)$, according to the offers received at round 1. In this case the deadline varies from 10 to 40 rounds.

In the configurations there are 5 ASs, AS_1, AS_2, AS_3, AS_4, AS_5, with a computational costs decreasing from AS_1 to AS_5. The user requested price is 1500$ ($reqPrice$). For each AS a default price $bestPrice_i$ is set, and the corresponding SPs will send as initial offer a price randomly extracted in the neighborhood of $bestPrice_i$ $[bestPrice_i - 5\% * bestPrice_i, bestPrice_i]$. This is because, even though the market price of services corresponding to the same AS is the same, to model a real market of services the variability of the first offered price is introduced. In the experiments, the market prices for the ASs are: $bestPrice_1 = 540\$, bestPrice_2 = 468\$, bestPrice_3 = 351\$, bestPrice_4 = 270\$, bestPrice_5 = 216\$$. The σ value randomly varies for each SP in the range [0.0, 0.5], so including the possibility that SPs with the maximum computational load are not willing to concede.

C. *Evaluation of Results*

In all the experiments 100 tests were performed for each of the described configurations.

Table I. SUCCESS RATE VARYING THE NUMBER OF ROUNDS WITH ONLY THE "BEST" SP FOR EACH AS.

Rounds	10	20	30	40
1SP for AS	1%	6%	12%	12%

In Figure 2 the percentage of negotiation successes varying the number of rounds and the number of SPs (from 2 to 16) for each AS is plotted. In Table I the same percentage is plotted, varying the number of rounds, but in the case of negotiation with only the "best" SP for each AS. As expected, for a deadline of 1 round (simple CNET protocol) we have 100% of failures since the specific settings of the tests require a negotiation phase to find a solution. In Table I the results show that selecting the best SP at round 1 does not reduce the negotiation cost. In fact, only the 12% of success rate is obtained with 30 or 40 rounds of negotiation allowed, so the computational cost of the negotiation is not compensated by an high rate of success. A better trend is obtained by adding another provider for each AS (as shown in Figure 2). In fact, in this case, with 30 or 40 rounds of negotiation allowed, 50% of successes are obtained. Scaling up the number of SPs from 4 to 16 the success rate increases from 90% to 100% just after 10 rounds. These results support the choice to negotiate with all the available SPs, as proposed by the OMM protocol, since the cost of negotiation is partially compensated by an increase in the negotiation success rate.

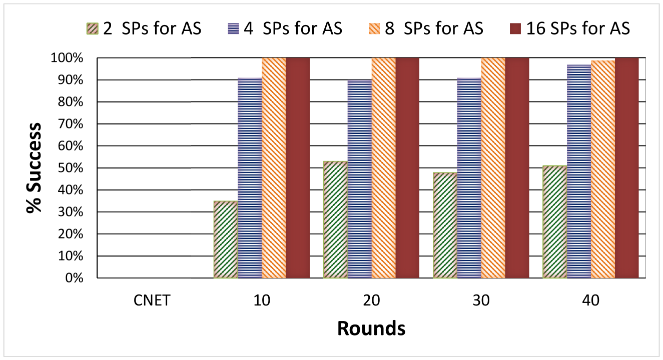

Figure 2. Success rate varying the number of rounds and the number of SPs for each AS.

Table II. NUMBER OF MESSAGES VARYING THE NUMBER OF SPS AND THE DEADLINE.

		# SPs				
		1	2	4	8	16
# rounds	1	15	30	60	120	240
	10	150	300	600	1200	2400
	20	300	600	1200	2400	4800
	30	450	900	1800	3600	7200
	40	600	1200	2400	4800	9600

In order to evaluate the computational cost of the OMM protocol, also the number of exchanged messages are considered. At each negotiation round the SC sends $m * n$ cfps, receives at the most $m * n$ possible offers, and it sends back at most $m * n$ accept and/or reject messages. This means that for each round the cost of communication in terms of exchanged messages is $3 * n * m$. The numbers of messages are reported in Table II for all the experimental configurations.

A comparative evaluation of Table II and Figure 2 is shown in Figure 3 that provides information on the trade-off between communication costs and success rate. In particular from configurations with 2400 exchanged messages to configurations with 9600 no variation in success rate is obtained (that is stable at 100%). This means that from 2400 onward there is only a communication overhead without any gain in the success rate. A first conclusion of this evaluation is that negotiating with 16 SPs for each AS with respect to

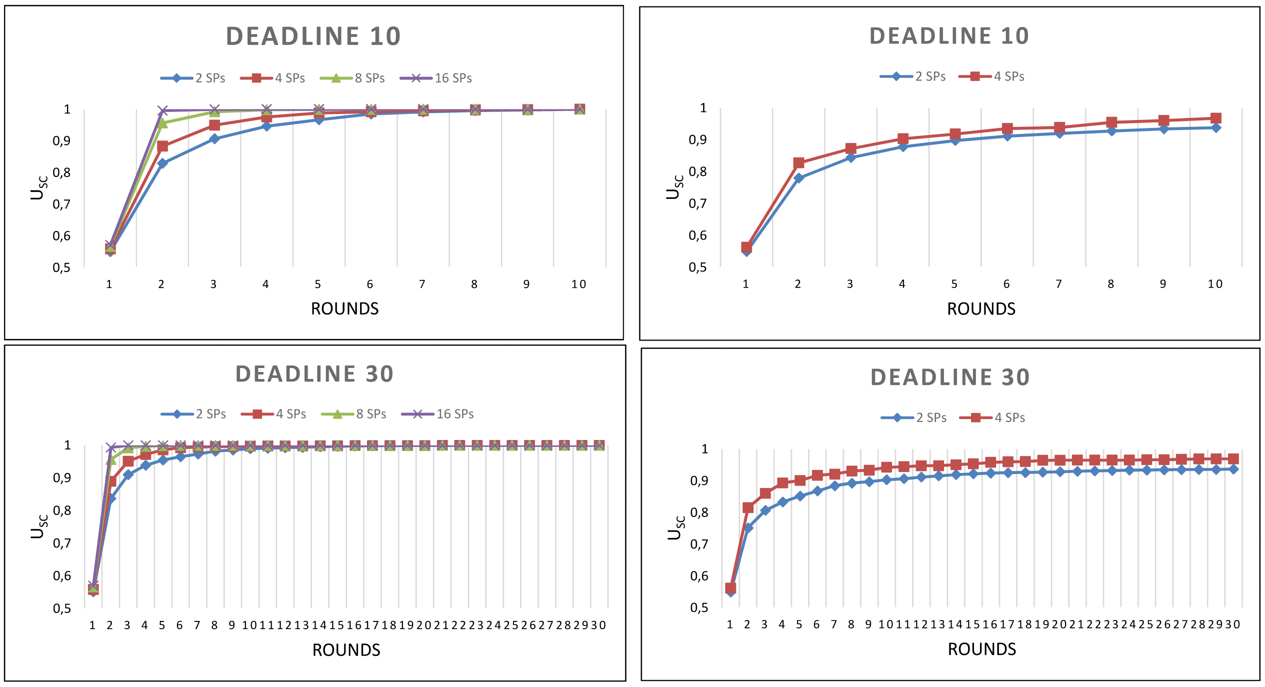

Figure 4. The SC's utility for different configurations in case of tests that led to a success (on the left) or to failure (on the right).

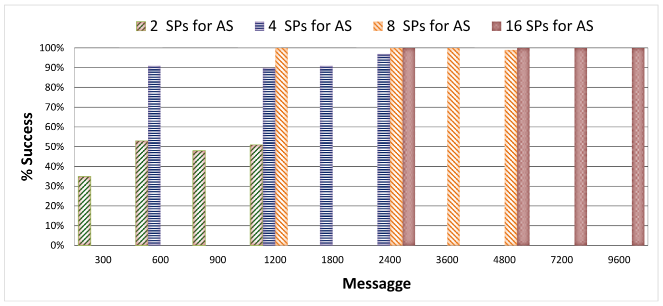

Figure 3. Success rate evaluated with respect to the number of messages.

negotiating with 8 SPs will not change the success rate, but it will only require more exchanged messages. So, in this case, selecting a subset of available providers reduces the cost of communication without affecting the success rate. Moreover, as already showed in Figure 2.b, such selection cannot be made only evaluating current offers because promising providers may change their concession strategy during the interaction. However, a bigger number of SPs, as shown in the following figures will provide a quicker achievement of the agreement, so reducing the negotiation length. Such evaluation of the number of exchanged messages with respect to the success rate shows that, in the case of a relevant number of available providers, long negotiation mechanisms are not necessary. So, in the following experimets only negotiation deadlines smaller than 40 rounds will be considered.

In Figure 4 the variation of the SC's global utility is reported at different negotiation rounds varying the number of available SPs and the deadline of the negotiation. In particular, on the left cases leading to success are considered varying the number of SPs for each AS from 2 to 16, and the deadline from 10 to 30 rounds. On the right cases leading to a failure are considered with the same deadlines as before, but varying the number of SPs for each AS from 2 to 4 since by increasing the number of SPs only successes are obtained from the round 10th onward. Let us note that the success is obtained as soon as the SC's utility becomes equal to 1, and it is reached in a less number of rounds by increasing the number of SPs for each AS. In case of failure, the SC's utility varies very little by increasing the number of the negotiation rounds, so making proceeding with negotiation expensive without any benefit.

Table III. SC'S UTILITY VARIATION IN CASE OF FAILURES.

		Round Range		
		10/20	20/30	30/40
#SPs	2	0,0223	0,0082	0,0042
	4	0,0221	0,0043	0,0083
Average		0,0222	0,0063	0,0062
STD		0,0002	0,0028	0,0029

In Table III the SC's utility variation in case of failure is reported in order to allow the SC to dynamically stop the negotiation according to its trend, i.e., according to whether and in which measure its utility is varying. The variation is calculated as a difference between the value of the SC's utility respectively at rounds 10 and 20 (for the first column), at rounds 20 and 30 (for the second column), at rounds 30 and 40 (for the third column). The variation is evaluated varying the number of SPs (2 and 4). The average SC's utility variation, and the corresponding standard deviation are also reported. As shown in Table III, in the configurations with the number of SPs equal to 2 and 4, by increasing the number of rounds the SC's utility variation is less than 1% after 20 rounds, so indicating that keeping on negotiating is not likely to lead to a success.

V. Conclusions and Future Works

Software agent negotiation is considered a promising approach for modeling the interactions between a service consumer and service providers when composing QoS-aware Service Based Applications. It is well recognized that the provision of such applications will be regulated by an agreement between the application consumer and the providers of the component services stating agreed terms and conditions related to both functional and non-functional application features. In particular, the negotiation mechanism is suitable to model the interactions occurring among consumers and providers when services are provided according to market-based mechanisms and when dynamic features, like QoS ones, have to be considered.

Nevertheless, automated negotiation did not succeed in real service-oriented market scenarios since it is computationally expensive in terms of the involved decision making mechanisms, and the communication overhead deriving from the complexity of the communication patterns.

In this work, an evaluation of the cost of the negotiation mechanism proposed in [5] is carried out with the aim to extract useful information to limit the length of the negotiation and also its communication overhead.

The experiments were carried out for different configurations obtained by varying the two parameters affecting the cost of communication that are the number of involved negotiators, i.e., the number of SPs, and the number of negotiation rounds determining the length of the negotiation. The results showed that the increase in communication costs due to the possibility of negotiating with all the SPs instead of just one for each service type, is partially compensated by the fact that by increasing the number of SPs the success rate of the negotiation increases. In fact, in the case the best SP is selected to continue the negotiation, a 100% failure is obtained also by increasing the length of the negotiation. This is because in a market of services, that is dynamic by nature, it is not possible to assume that a promising provider will keep on sending promising offers, because a less promising provider may change its strategy in the meantime according to market trends and/or market strategies that are tied also to the specific service or quality attribute. So, the choice of negotiating with all the SPs is supported by the obtained results. Furthermore, in most configurations, the overhead due to the communication cost is partially compensated by a decrease in the negotiation length, i.e its overall computational cost.

In some cases the negotiation progress in terms of the distance between the requested QoS and the QoS obtained at each negotiation round shows that it is not worth to proceed with the negotiation after a certain number of rounds since no gain is obtained in the success rate. This means that in these cases the negotiation can be stopped without any loss in terms of consumer's utility, so limiting the cost of negotiation in terms of its length. Of course, these results are related to the considered configurations, and also to the specific strategies adopted for the SPs.

We plan to extend the proposed negotiation mechanism by including the possibility for the SPs to change their strategies on fly, and to carry out more experiments by considering different set of strategies to evaluate the negotiation cost in different experimental settings.

Acknowledgements

The research leading to these results has received funding from the EU FP7-ICT-2012-8 under the MIDAS Project (Model and Inference Driven - Automated testing of Services architectures), Grant Agreement no. 318786, and the Italian Ministry of University and Research and EU under the PON OR.C.HE.S.T.R.A. project (ORganization of Cultural HEritage for Smart Tourism and Real-time Accessibility).

References

[1] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *IEEE Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[2] J. Yan, R. Kowalczyk, J. Lin, M. B. Chhetri, S. K. Goh, and J. Zhang, "Autonomous service level agreement negotiation for service composition provision," *Future Generation Computer Systems*, vol. 23, no. 6, pp. 748 – 759, 2007.

[3] A. Keller and H. Ludwig, "The wsla framework: Specifying and monitoring service level agreements for web services," *J. Network System Management*, vol. 11, no. 1, pp. 57–81, 2003.

[4] S. Paurobally, V. Tamma, and M. Wooldrdige, "A framework for web service negotiation," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 4, Nov. 2007.

[5] C. Napoli, P. Pisa, and S. Rossi, "Towards a dynamic negotiation mechanism for qos-aware service markets," in *Trends in Practical Applications of Agents and Multiagent Systems*, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2013, vol. 221, pp. 9–16.

[6] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, may 2004.

[7] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. on Software Eng.*, vol. 33, no. 6, pp. 369–384, 2007.

[8] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th Int. Conf. on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 881–890.

[9] R. Y. K. Lau, "Towards a web services and intelligent agents-based negotiation system for b2b ecommerce," *Electronic Commerce Research and Applications*, vol. 6, no. 3, pp. 260–273, 2007.

[10] A. Lomuscio, M. Wooldridge, and N. Jennings, "A classification scheme for negotiation in electronic commerce," *Group Decision and Negotiation*, vol. 12, no. 1, pp. 31–56, 2003.

[11] G. Zlotkin and J. S. Rosenschein, "Mechanism design for automated negotiation, and its application to task oriented domains," *Artif. Intell.*, vol. 86, no. 2, pp. 195–244, 1996.

[12] F. Siala and K. Ghedira, "A multi-agent selection of web service providers driven by composite qos," in *Proc. of 2011 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2011, pp. 55–60.

[13] P. R. Wurman, M. P. Wellman, and W. E. Walsh, "The michigan internet auctionbot: a configurable auction server for human and software agents," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98. New York, NY, USA: ACM, 1998, pp. 301–308.

[14] R. G. Smith, "The contract net protocol: High–level communication and control in a distributed problem solver," *IEEE Trans. on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[15] "Fipa iterated contract net interaction protocol specification."

[16] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation Decision Functions for Autonomous Agents," *Robotics and Autonomous Systems*, vol. 24, pp. 3–4, 1998.