A MULTI-AGENT SELECTION OF MULTIPLE COMPOSITE WEB SERVICES DRIVEN BY QOS

Fatma Siala and Khaled Ghedira
University of Tunis – SOIE, Tunis, Tunisia

Keywords: QoS, Web service, Multi-agent system, Composition, Contract-net protocol, Combinations, Execution paths.

Abstract: As the number of functionally similar Web Services from providers with different Quality of Service's (QoS) scores is increasing, a selection needs to be made to determine which services are to participate in a given composite service. Moreover, QoS becomes one of the most important factors for Web Service selection. Indeed, one of the main assets of service-orientation is a composition to develop higher level services, so-called composite services, by re-using existing services. However, for a composition, we can have different combinations and execution paths. Particularly, a composite service can generate different schemes that give various QoS scores. We propose, in this paper, two frameworks. The first framework deals with the selection of composite Web services on the base of Multi-Agents negotiation. The objective of these agents is to find out the best Composite QoS (CQoS) based on Web services availability. The second framework improves the first one by supporting different combinations and execution paths.
The proposed Multi-Agents frameworks are compared to an existing approach in terms of execution time and QoS's score. Experiments have demonstrated that our frameworks provide reliable results in comparison with the existing approach.

1 INTRODUCTION

Web service technology is greatly contributing to change the landscape of today's software engineering. One of the most promising advantages of the service-oriented paradigm is related to service run-time discoveries and late binding mechanisms (Canfora et al., 2008).

Web services are the basis of service-oriented software, which is based on network, including simple standards such as the Web Services Description Language (WSDL), the Universal Description, Discovery and Integration (UDDI) and the Simple Object Access Protocol (SOAP). These standards can help service publication, service location and use services through Internet. WSDL is the standard language for describing a Web service. UDDI is used to register and search Web services. SOAP, as a transport layer, is used to send messages between service users and service providers[1]. XML is a language to support these standards.

Indeed, one of the main assets of service's orientation is composition to develop higher level services. A composite Web service is a collection of single Web services that form a new complex service. Based on a control flow description of the composition, suitable task candidates must be discovered and from these candidates, the most suitable must be selected. The flow description with an assignment of concrete services can be used to parametrise an execution engine to perform the composition (Jaeger and Ladner, 2006).

As the most requests concern composite services, the QoS-based Composition (CQoS) selection becomes very important. It is related to the quality of any elementary service. Notably, a system composed of services that ensures the individual QoS that should achieve optimal CQoS is always a problem during the search. Issues related to the guaranteed QoS compositions are as important as the issues related to the functional services. The Web services QoS focus on non-functional attributes identified by the W3C[2]. The elements of quality include price, availability, reliability, reputation and so on.

Moreover, a composite service can be represented by a statechart which has multiple execution paths

[1] http://www.w3c.org/TR/wsdl

[2] http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

Siala F. and Ghédira K..
A MULTI-AGENT SELECTION OF MULTIPLE COMPOSITE WEB SERVICES DRIVEN BY QOS.
DOI: 10.5220/0003555306750684
In *Proceedings of the 1st International Conference on Cloud Computing and Services Science* (MAS-2011), pages 675-684
ISBN: 978-989-8425-52-2

when containing conditional branchings. Each execution path represents a sequence of tasks to complete a composite service execution. Furthermore, for a composite Web service, we notice that we can have different possible combinations.

Our work aims at advancing the current state of the art in technologies for Web service composition by first, taking into account the user's preferences, second, by using agents to negotiate the QoS value and dynamically select providers for various services in the composition (based on availability). Finally, by proposing a framework that supports different execution paths and combinations. Our contributions are revealed when negotiating only with available Web services providers that give a better Central Processing Unit (CPU) time and by supporting different execution paths and combinations for a composition. The combination concept has never been addressed by any approach.

The structure of this paper is as follows: The coming section briefly introduces the main concepts used in our work. Section 3 presents the major related works in the area of Web service composition based on QoS. Section 4 proposes two frameworks prototypes, an initial framework and its amelioration for taking into account different execution paths and combinations in a composition. Section 5 explains the implementation of these frameworks using a case study. Accordingly, section 6 presents the experimentation results. After that, the approach is discussed in section 7 by presenting existing approach limitations and detailing our contributions and the difference between the two proposed frameworks. Finally, we conclude the whole paper.

2 MAIN CONCEPTS

In this section, we define three concepts used in the remainder of the paper: execution path, composite Web service and Web service combination.

To simplify the discussion, we initially assume that all the statecharts that we deal with are acyclic. If a statechart contains cycles, a technique for unfolding it into an acyclic statechart needs to be applied beforehand detailed by (Zeng et al., 2004). We present in Fig. 1 an example of a statechart with AND and OR states.

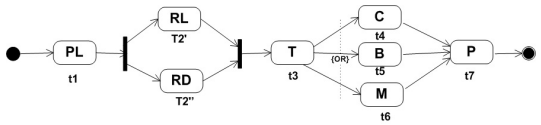

Figure 1: Example of a statechart.

2.1 Execution Path

If a statechart contains conditional branchings, it has multiple execution paths. Each execution path represents a sequence of tasks to complete a composite service execution. Fig. 2 gives an example of a statechart's execution paths. In this example, since there is one conditional branching after service S_T, there are three paths, called EP1, EP2 and EP3 respectively. In the execution path EP1, service S_C is executed after service S_T while in the execution path EP2, service S_B is executed after service S_T and in the execution path EP3, service S_M is executed after service S_T.

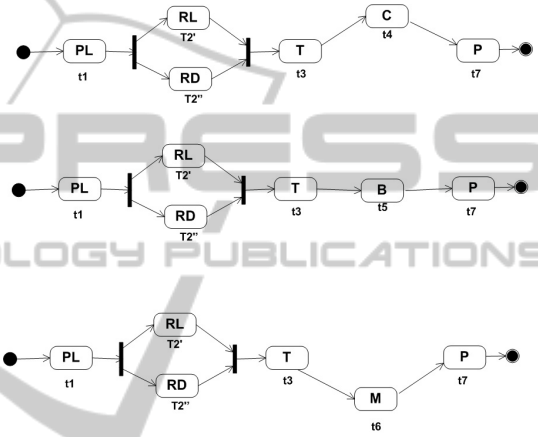

Figure 2: Execution paths representation of Fig 1 statechart.

2.2 Composite Web Service

The services composition can be defined as "the process of combining existing services to form new services". These assembled Web services will certainly play a greater role for communication among applications. In particular, if non of the services are able to respond to a request from a user, it will be possible to combine existing Web services to perform this task.

As shown in Fig. 3, a service composition consists normally of some services linked together, each of them may be provided by more than one supplier. For a particular service, many providers may offer the same functionality but offer different QoS. We assume in Fig.1 that a composite service is composed on n services and each service S_i may be provided from p providers S_{i1} to S_{ip}. Let n = 3 and p = 5 (as an example), to supply the composite service, we need services S_1, S_2 and S_3 in the composition. In addition, S_1 may be afforded by service providers $S_{1.1}$, $S_{1.2}$, $S_{1.3}$, $S_{1.4}$ or $S_{1.5}$.

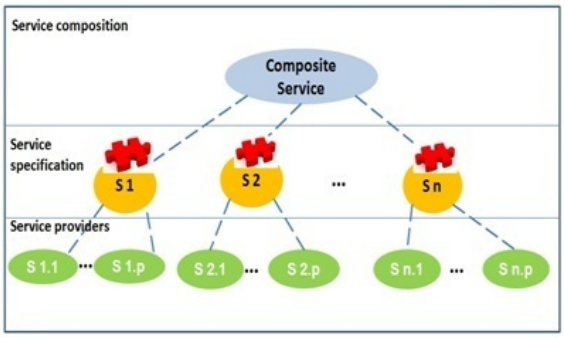

Figure 3: Web service composition.

2.3 Web Service Combination

For a composite Web service, we notice that we can have different possible combinations. As an example, when a user requests a composite Web service, the same result service can be provided from different web services combinations (fig. 1 and fig. 4). The user will not be aware if the Web service S_T is executed after or before Web services S_{RL} and S_{RD}. The two combinations give the same result but not the same QoS.

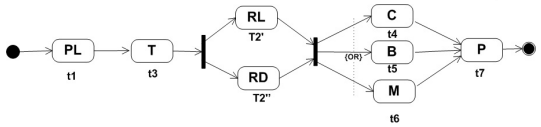

Figure 4: Example of a combination.

3 RELATED WORKS

Several studies have been made to the services composition taking into account QoS requirements (Brandic et al., 2008) (Mukhija et al., 2007) (Kona et al., 2009). In this section, we review selected works based on their relevance for our approach.

(Zeng et al., 2004) have presented a solution for the composition problem by analyzing multiple execution paths of a composite service which are specified using UML (Unified Modeling Language) statecharts. They have modeled the composition problem using different approaches, including a local optimization approach and global planning approach using linear programming.

(Guan et al., 2006) are the first who propose a framework for QoS-guided service compositions which uses constraint hierarchies as a formalism for specifying QoS. They use a branch and bound algorithm that is only capable of solving sequential compositions. The authors do not present any empirical evaluation to demonstrate the optimization performance of their approach.

(Rosenberg et al., 2009) have used constraint programming and integer programming approach for optimizing QoS by leveraging constraints hierarchies as a formalism to represent user constraints (specified with a Domain-Specific Language) of different importances.

(Canfora et al., 2008) have proposed an approach based on genetic algorithms. To determine the optimal set of concretizations, the approach needs to estimate the composite service QoS. This is done using some aggregation formulas.

(Hong and Hu, 2009) have used an ordinary utility function as a numerical scale of ordering local services and a multi-dimension QoS based local service selection model is proposed to provide important grounds to choose a superior service and shift an inferior one. Secondly, subjective weight mode, objective weight mode, and subject-objective weight mode are constructed to determine the weight coefficient of each QoS criterion, and to show the users' partiality and the service quality's objectivity.

(Alrifai and Risse, 2009) have employed Mixed Integer Programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints. They have used distributed local selection to find the best web services that satisfy these local constraints.

(Yan et al., 2007) have presented a framework in which the service consumer is represented by a set of agents who negotiate QoS constraints specified using SLA (Service Level Agreement), with the service providers for various services in the composition applying the Contract-Net protocol. Their idea of using Multi-Agents System and the Contract-Net protocol is in line with the work presented in this paper. However, the authors do not deal with the the Web dynamism (the availability concept) so their approach takes a large CPU time. Moreover, their framework does not support the different execution paths and combinations.

The majority of these approaches does not take into account that for a composite service we can have an execution plan that generates different execution paths. These approaches deal only with the optimization problem itself (finding the best CQoS) without giving prominence for this aspect. Some approaches deal with this aspect but repeating each time the generation of all the elementary Web services. Moreover, all these approaches do not take into consideration that for a composite Web service we can have different combinations. This concept has never been addressed by any approach. By using Multi-Agents

System, we generate all the execution paths and combinations in parallel and select the best execution path or combination, so we gain in terms of CPU time.

We also take into consideration the importance of the Web services' availability since the Web is a dynamic environment. By considering only available Web services we also improve the CPU time. Our approach allows to find the best CQoS by considering different execution paths and combinations and by also taking into account the dynamical aspect of the Web (the web service availability changes) which directly influences the CPU time.

4 THE PROPOSAL FRAMEWORKS

The first step towards autonomously establishing QoS value for a service composition is to have a supporting framework. This framework should be able to address the special requirements for establishing QoS value for a service composition. For the composition process, we use (Lajmi et al., 2006) technique.

Our goal is to propose an approach to the Web services composition that guarantees non functional properties (QoS). This approach must also support different execution paths and combinations. We present, in the following, two frameworks. The first framework allows to select the best elementary Web services in terms of QoS based on Web services availability. The second framework offers all first framework's functionalities but also supports different execution paths and combinations. Several motivations lead to use Multi-Agent Systems (MAS) (Barbuceanu and Fox, 1996). In fact, a Web service is passive until it is invoked, have only a knowledge of himself and is not adaptable and is not able to benefit of new capabilities of the environment (Charif et al., 2010).

We propose to represent each service by an agent. In the MAS field, negotiation is a core component of agents interactions. Indeed, since agents are autonomous, there is no imposed solution in advance, but agents must reach solutions dynamically while solving problems. The basic assumption of this approach is that each elementary Web service has an agent responsible to it. The objective of these agents is to find out the best CQoS. They aware available providers also represented with agents.

In this context, we propose two global frameworks. The first one optimizes the QoS criteria for a composite service provision. The second framework improves the first one by considering different execution paths and combinations for a given statechart.

4.1 The First Framework

To better explain our approach, we present in Fig 5 the first framework associated to Fig 3. This framework consists of an Interface Agent (IA) and a set of Negotiator Agents (NAs) that negotiate with a set of Web Service Agents (WSAs).

The IA associates an NA for each elementary service in the composition. These NAs negotiate via the Contract-Net (CNET) protocol with WSAs (the providers) to find the best elementary Web service and send the response to the IA which evaluates the results and either confirms the acceptance or repeat the negotiation.

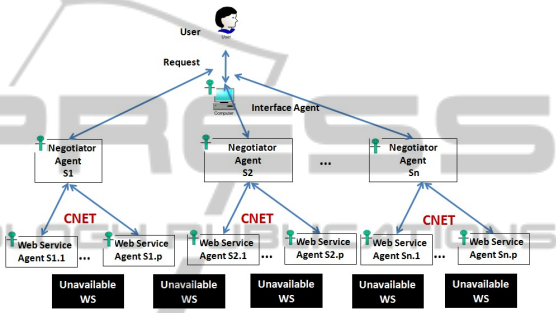

Figure 5: The First Framework.

4.1.1 The Interface Agent

The Interface Agent (IA) represents the interface that allows the user to access the framework for specify the desired service and QoS criteria. In fact, the user can specify each criterion with a weight since different users may have different requirements and preferences regarding QoS. For example, a user may require to minimize the execution time, while another user may give more importance to the price than to the execution time. The selection process is based on the weight assigned by the user to each criterion. The IA computes the CQoS score using a formula proposed by (Zeng et al., 2004). The IA will then, provide the expected service with the required QoS criteria.

We should note that the system can allow a user to define constraints expressed using a simple expression language. Examples of constraints that can be expressed include duration constraints and price constraints.

4.1.2 The Negotiator Agent

A Negotiator Agent (NA) is responsible of a Web services set offering the same functionality. This agent is in charge of negotiating with providers for a service from the composition in order to optimize the QoS. We use a variant of the CNET protocol, the directed

award where the critical information which must be aware by the NA is the availability. So, the NA acquaintances are the available Web service providers. For a negotiation, the NA must consult its list containing available Web services.

4.1.3 The Web Service Agent

A Service Level Agreements (SLA) defines the terms and conditions of service quality that a Web service delivers to service requesters. The major constituent of an SLA is the QoS information. There are a number of criteria (e.g., execution time, availability) that contribute to a QoS in an SLA. Web service providers publish QoS information in SLAs.

Each Web Service Agent (WSA) represents a Web service. This Web service belongs to a Web services class where an NA is responsible. For example, a travel service provider may specify that it supports the Trip-planning service and belongs to the service class FlightTicketBooking. Service class describes the capabilities of Web services and how to access them.

The NA has a list that he consult whenever he begins a negotiation. In this list there is the available WSAs. By this method, we also take into account the failure cases.

We note that in Fig 5, there is unavailable Web services represented by agents that will not negotiate with the NA.

4.2 The Negotiation

4.2.1 Negotiation Protocol

The negotiation protocol is the way and manner the negotiating parties interact and exchange information. It includes the way in which the offers and messages are sent to opponents. There are various negotiation protocols available in the research community. In this paper, we propose to use the CNET Protocol, designed by (Smith, 1980), and especially a variant named directed award where the manager must have a table of acquaintances that contains knowledge about other agents (eg, skills, knowledge, value judgments about these agents). In this protocol, one agent (the initiator) takes the role of manager which wishes to have a task performed by the other agents (the participants) and further wishes to optimize a function that characterizes this task. We use the function proposed by (Zeng et al., 2004). For a given task, any number of the participants may respond with a proposal, the rest must refuse.

4.2.2 Negotiation Coordination

The IA coordinates multiple negotiations for various services in the composition. The purpose of the coordination is to ensure that the results of these multiple negotiation can collectively fulfill the end-to-end QoS. The first task that the IA performs is to attribute a Web service with its QoS weights to the NAs. The second task that the IA performs is to confirm negotiation results. As the extended negotiation protocol suggests, when various NAs get the best deals, they consult the IA for confirmation. The IA evaluates the results and either confirms the acceptance or amends the reserve values to continue the negotiation. The QoS aggregation refers to the QoS model proposed in (Zeng et al., 2004).

The NA sends a CFP message only to the available Web Service Agents. When The proposals and counter proposals are then communicated iteratively between the NAs and the service providers (WSAs), following the standard FIPA protocol, until the best deal is reached (i.e. the proposals offered by one or more providers can satisfy the negotiation objectives) or the timeout occurs. The NA block the selected WSA. At this time, the best deal is sent to the IA. If the overall QoS requirements are satisfied, the IA confirms to each NA that the current deal is acceptable. Subsequently, the NA acknowledges the acceptance of the proposal to the selected providers (WSAs). When the user finish using Web services, the IA inform the NA to unlock the selected WSA.

In the case that the overall QoS requirements are not satisfied based on the current best deals, the user should modify the requirements and the IA amends the reserve values for the NA to re-start negotiation. Each corresponding NA then sends the modified CFP to WSA and begins a new negotiation process

This research refers to the QoS model presented in (Zeng et al., 2004) which consider five quality attributes but other quality dimensions can be used instead without any fundamental changes. The dimensions are numbered from 1 to 5, with 1 = price, 2 = duration, 3 = availability, 4 = success rate, and 5 = reputation. Given a task t_j in a composite service, there is a set of candidate (elementary) Web services $S_j = \{s_{1j}, s_{2j}, \ldots, s_{nj}\}$ that can be used to execute this task. By merging the quality vectors of all these candidate Web services, a matrix $Q = (Q_{i,j}; 1 \leq i \leq n, 1 \leq j \leq n)$.

In this case n=5. This matrix is built, in which each row Q_j corresponds to a Web service s_{ij} while each column corresponds to a quality dimension.

A Simple Additive Weighting (SAW) technique is used to select an optimal Web service. There are two phases in applying SAW:

- Scaling Phase:

Some of the criteria could be negative, i.e., the higher the value, the lower the quality. This includes criteria such as execution time and execution price. Other criteria are positive, i.e., the higher the value, the higher the quality. For negative criteria, values are scaled according to (1). For positive criteria, values are scaled according to (2).

$$V_{i,j} = \begin{cases} \frac{Q_j^{max} - Q_{i,j}}{Q_j^{max} - Q_j^{min}} & \text{if } Q_j^{max} - Q_j^{min} \neq 0 \\ 1 & \text{if } Q_j^{max} - Q_j^{min} \neq 0 \end{cases} \quad (1)$$

$$V_{i,j} = \begin{cases} \frac{Q_{i,j} - Q_j^{min}}{Q_j^{max} - Q_j^{min}} & \text{if } Q_j^{max} - Q_j^{min} \neq 0 \\ 1 & \text{if } Q_j^{max} - Q_j^{min} \neq 0 \end{cases} \quad (2)$$

hspace-1cm In the above equations, Q_j^{max} is the maximal value of a quality criteria in matrix Q, i.e., $Q_j^{max} = Max(Q_{i,j}), 1 \leq i \leq n$.

While Q_j^{min} is the minimal value of a quality criteria in matrix Q, i.e.,

$Q_j^{min} = Min(Q_{i,j}), 1 \leq i \leq n$.

By applying these two equations on Q, we obtain a matrix $V = (V_{i,j}; 1 \leq i \leq n, 1 \leq j \leq n)$,

in which each row Vj corresponds to a Web service sij while each column corresponds to a quality dimension.

- Weighting Phase :

The following formula is used to compute the overall quality score for each Web service:

$$score(p_i) = \sum_{j=1}^{n} (V_{ij} * W_j) \quad (3)$$

where $W_j \in [0,1]$ and $\sum_{j=1}^{n} W_j = 1$.

Wj represents the weight of criterion j. End users express their preferences regarding QoS by providing values for the weights W_j.

For a given task, the NA will choose the Web service which satisfies all the user preferences for that task and which has the maximal score. If there are several services with maximal score, one of them is selected randomly. If no service satisfies the user preferences for a given task, an execution exception will be raised and the IA will propose the user to change his preferences.

4.3 The Second Framework

We propose an improvement for the first framework to enable the support of different execution paths and combinations (second framework represented in fig 6). We add a new agent category, the Combination Coordination Agent (CCA).

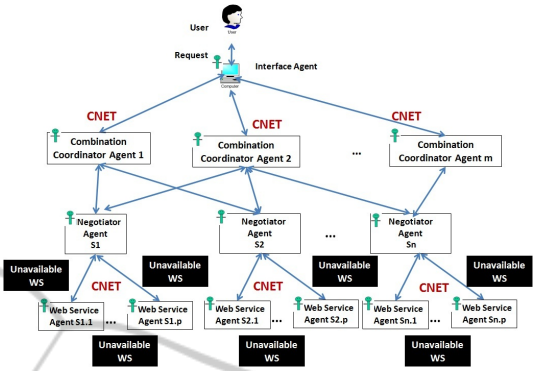

Figure 6: Second Framework.

First, the user specify the desired composite Web service and the associated requirements. Then, The IA charges each CCA for an execution path or combination of the statechart, Second, the IA negotiates via the CNET protocol for the best execution path. Finally, the IA returns to the user the best Web service composition. We explain in the following the role or each agent. We explain the negotiation process through an AUML protocol diagram (Fig 7). The negotiation between the other agents is like to the first framework.

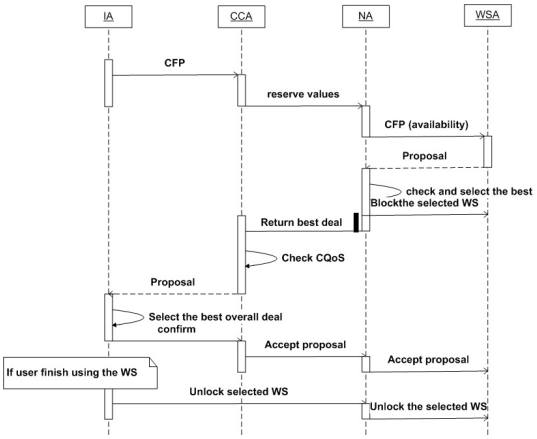

Figure 7: Negotiation protocol success.

In the case that the overall QoS requirements are not satisfied based on the current best deals, the user modify the requirements and the IA amends the reserve values for the NA to re-start negotiation. Each corresponding NA then sends the modified CFP to WSA and begins a new negotiation process, as shown in Fig 8).

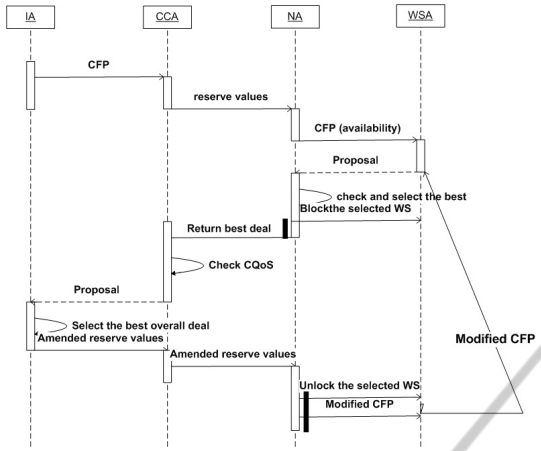

Figure 8: Negotiation protocol failure.

4.3.1 The Interface Agent

The Interface Agent (IA) represents the interface that allows the user to access the framework to specify the desired service and QoS criteria. In fact, the user can indicate each criterion with a weight since different users may have different requirements and preferences regarding QoS. The IA charge each CCA for an execution path or combination of the statechart and will, thereafter, select using the CNET protocol the best proposal from CCAs.

4.3.2 The Combination Coordinator Agent

The Combination Coordinator Agent (CCA) interacts with the IA to receive an execution path or combination of the statechart for the service composition and the user requirements. Each CCA is responsible for an execution path or combination of a composition. This agent attributes a NA for each Web services and computes the CQoS score using a formula proposed by (Zeng et al., 2004) when it receives their responses. Finally, the CCA negotiates with the IA using the CNET protocol to provide its proposal.

4.3.3 The Negotiator Agent

A Negotiator Agent (NA) has the same role as in the first framework however, in this case, it negotiates with the CCAs instead of the IA. The selected WSA is blocked (reserved).

The Web Service Agent saves the same role as in the first framework. Once again, we note that in Fig 6, there are unavailable Web services represented by agents that will not negotiate with the NA. The WSA has the same role as in the first framework.

When the user finishes using the Web services, the IA inform the NA which should also inform the WSA to unlock the WSA.

5 IMPLEMENTATION AND CASE STUDY

To show the key ideas presented in this paper, a prototype has been implemented for the proof-of-concept purpose using the FIPA compliant JADE Agent Framework [3] which is a middleware that implements an agent platform and a development framework. This framework supports agents' negotiations through an Agent Communication Language (ACL).

During the negotiation, the IA, the CCAs, the NAs and the WSAs exchange a number of messages.

We explain our approach through a case study. A simplified statechart specifying a scenario in the tourism industry composite Web service is depicted in Fig 1. In this scenario, a tourist who holds a mobile device can request the full description of the route information from his/her current position to a selected attraction. We have height different services, that will be invoked, defined as follows:

- A Phone Location Service(S_{PL}): to find the current position of the tourist.
- A Route Calculation Service (S_{RL}): to calculate the route from the tourist's position to the attraction.
- A Route Description Service(S_{RD}): to generate and present graphical and textual description of the route to the tourist.
- A Traffic Service (S_T): to present the traffic of the route from the tourist's position to the attraction.
- A Car Service (S_C): to generate the duration that could be taken at this moment from the tourist's position to the attraction, if the tourist uses a car.
- A Bus Service (S_B): to generate the duration that could be taken at this moment from the tourist's position to the attraction, if the tourist uses a bus.
- A Metro Service (S_M): to generate the duration that could be taken at this moment from the tourist's position to the attraction, if the tourist uses a metro.
- A Metro Service (S_P): to generate the cost that the tourist should pay.

[3]JADE (Java Agent Development Framework), Telecom Italia Lab, version available at: http://sharon.cselt.it/projects/jade/

In this example, after the service S_{PL} is completed, the service S_{RL} is done in parallel with S_{RD}. After these services are completed, the S_T is triggered and either S_C, S_B or S_M is invoked. Finally, the service S_P is generated (Fig 1).

The tourist can also specify some QoS requirements when making his/her request. For example, the tourist can request that the score of CQoS is delivered with a value top then 70. Obviously, the tourist can also require the QoS score for the elementary Web services such S_V, S_B, S_M, etc.

The tourist should also indicate the weight associated to each QoS attribute. Example of weights values : price=0.15, duration=0.35, success rate=0.40 and reputation=0.10. These weights will be used for the computation of the QoS score of each elementary Web services using formula 3. If the user does not specify weights, the system will consider a weight value = 0.25 for each criterion.

So, each IA send at the same time a CFP to the CCAs for an execution path or combination in the statechart of the composition. In our case (Fig 2), we have three execution paths. Each NA (height NAs) associated to an elementary Web service negotiates with the available WSAs for one service in the composition. The WSAs randomly generate proposals (QoS score) with the value between 10 and 100.

Each NA negotiates with providers simultaneously. The negotiation results for each service are summarized in Table 1. After the reception of offers from available WSAs, the height NAs select the best offer, block the WSAs associated to the selected Web services and return their best offers to the CCAs. Each CCA (three CCAs) calculates its CQOS score. Supposing that we have these results after the negotiation, EP1= 65; EP2 = 84 and EP3=40. The IA will choose the EP2 that has the best CQoS. The whole process is simulated using the prototype.
The following results are confirmed:
Metro Service (S_{PL}) : 66
Phone Location Service (S_{RL}) : 98
Metro Service (S_{RD}) : 80
Phone Location Service (S_T) : 79
Route Calculation Service (S_C) :87
Route Description Service (S_B) : 73
Traffic Service (S_M) : 85
Bus Service (S_P) :92

These results are associated to the best QoS of each elementary Web service. Finally, the IA checks if the best offers can jointly fulfill the user's request (the desired value of CQoS is 75) using a formula also proposed by (Zeng et al., 2004).

When the tourist finishes using the Web service, the IA informs the NA to unlock the selected (reserved) WSA.

Table 1: QoS values (negotiation results) for each WSA.

	S_PL	S_RL	S_RD	S_T	S_C	S_B	S_M	S_P
1	65	46	80	25	35	73	85	92
2	33	39	40	48	28	54	77	45
3	66	24	50	38	66	49	76	22
4	40	45	26	79	32	46	80	21
5	22	98	44	75	87	24	37	64

6 EXPERIMENTATION

The first series of tests were conducted to compare the CPU time of our approach with the approach of (Yan et al., 2007) to find the best QoS for each elementary service and to perform the CQoS of the first framework.

In the experimentation, we have calculated the CPU time of each approach (the Yan et al.'s approach (Yan et al., 2007), the first and the second frameworks) by fixing the number of elementary services in a composition to 25 and varying the number of service providers from 10 to 100 with steps of 10 for 50 execution paths. We have calculated the CPU duration 10 times for each case considering the average. The results of these experiments are presented via a chart (Fig. 9).

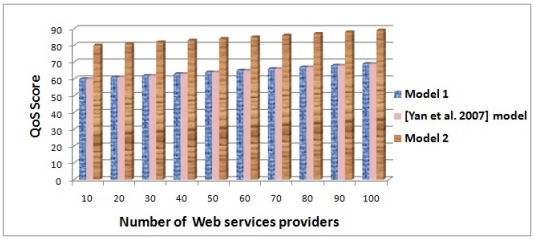

Figure 9: Comparison of CPU time.

These experiments show that by increasing the number of Web services providers, the CPU time also increases for each approach. The first framework takes a largely better CPU time than (Yan et al., 2007)'s approach but the second framework takes a little more time than (Yan et al., 2007)'s approach. This result is evident since the Yan et al.'s approach and the first framework does not take into account different execution paths and combinations. Moreover, the first framework sends the CFP only to the available WSAs. On the other hand, the second framework supports different execution paths and combinations in a composition by also sending the CFP only to the available WSAs.

We compare in Fig. 10 the quality score given by

each approach by also fixing the number of elementary services in a composition to 25 and varying the number of service providers from 10 to 100 with steps of 10. obviously, the first framework and the (Yan et al., 2007)'s approach give the same quality score. However, the second framework gives a largely better CQoS score since it supports different execution paths and combinations and chooses the best.

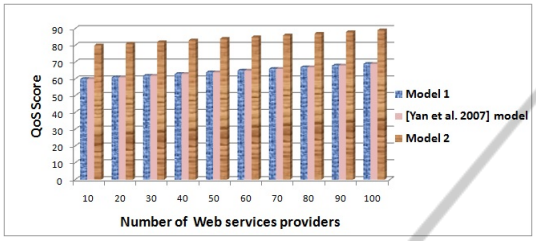

Figure 10: Comparison of QoS score.

By comparing only the CPU time or only the CQoS score, we cannot conclude either the first framework is better than the second framework or the contrary. That's what we propose to use the formula (4) which calculates a ratio between the time and the quality. We show through Fig. 11, that the ratio between the time and the QoS is largely lower than 1. This explains that although the second framework takes more CPU time than the first framework, it greatly improves the CQoS score.

$$Ratio = \begin{cases} \frac{\frac{T_2 - T_1}{T_2^{min} - T_1^{min}}}{\frac{Q_2 - Q_1}{Q_2^{min} - Q_1^{min}}} & \text{if } T_2^{min} - T_1^{min} \neq 0 \\ & and Q_2^{min} - Q_1^{min} \neq 0 \\ 1 & \text{if } T_2^{min} - T_1^{min} = 0 \\ & and Q_2^{min} - Q_1^{min} = 0 \end{cases} \quad (4)$$

In the above equation, T_1 is the average of CPU time (we test 10 values in the experimentation for the same case) taken by the first framework, T_1^{min} is the minimum value of CPU time between the 10 values taken in the experimentation for the first framework. Q_1 is the average of CQoS taken by the first framework, Q_1^{min} is the minimum value of CQoS between the 10 values taken in the experimentation for the first framework. The same notions are respectively taken for T_2, T_2^{min}, Q_2 and Q_2^{min} to the second framework.

7 DISCUSSION

We compare our proposed frameworks with Yan et al.'s one (Yan et al., 2007). To the best of our knowledge, their work is the sole which uses agent technology. (Maamar et al., 2005) have used MASs for Web service composition but they don't consider the QoS.

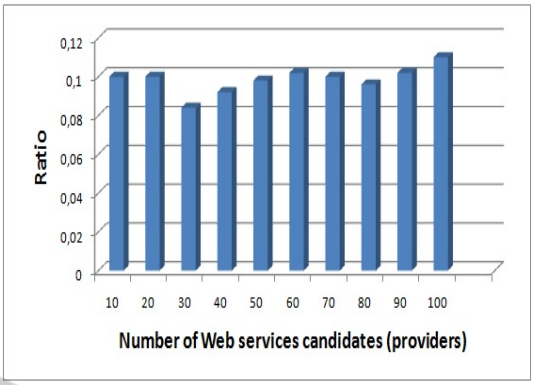

Figure 11: Comparison of ratio between the first and the second frameworks.

The (Yan et al., 2007) approach represents some limitations, when the CNET protocol is used. The NA has no knowledge about the agents. It will then send the CFP to all service providers, and thus can cause network congestion.

Our first framework improves the work of (Yan et al., 2007) in terms of CPU time. In fact, we decrease the number of negotiations (we alleviate the network) by using a variant of the Contract-Net protocol, called the directed award and sending the CFP only to the available Web Service Agents. This contribution gains on CPU time.

On the other hand, our second framework improves the work of (Yan et al., 2007) and the first proposed framework in terms of QoS by supporting multiple execution paths and combinations for a composition. By using formula (1), we have demonstrated that the difference of CPU time is negligible compared to the improvement of QoS. Supporting different combinations for a composition is a novel idea introduced by our work. The greatest limitation of the second framework is its lack of scalability. Therefore, we are preparing a new scalable framework using Case Based Reasoning.

The first framework optimizes the QoS at the local level and verify after that if it ensures the QoS at the global level (CQoS). On the other side, the second framework, by considering different execution paths and combinations, has more chances to ensure the QoS at the global level.

Note that our approach supports as more attributes of QoS (price, duration, reputation, success rate, availability, etc.) as we want.

8 CONCLUSIONS

In service-oriented computing, adequate support for the service consumers and the service providers to achieve agreements on QoS values is highly demanded. This important issue becomes more complex in the context of service composition provision, where the service consumer needs to achieve a set of interrelated agreements with various services providers in order to collectively fulfill the end-to-end QoS requirements. Moreover, for a composite Web service, we notice that we can have different possible combinations and multiple execution paths since it contains conditional branchings.

This paper exploits the agent technology to address this crucial issues. A first framework is proposed using agents that are able to negotiate with service providers, in a coordinated way, to ensure end-to-end QoS. Based on this framework, the negotiation follows an extended FIPA protocol. So, agents can exchange meaningful messages with service providers in a defined order. A contract Net protocol variant, called the directed award based on Web services providers availability is used. A second framework is proposed to support different execution paths and combinations for a composition.

Our approach advances the current state of the art by taking into account the Web services availability and supporting different execution paths and combinations for a composition. The proposed framework is not scalable since the number of combination or execution paths can be exponential. In the future work, we will propose a new scalable framework which improves our second framework by using Case Based Reasoning.

REFERENCES

Alrifai, M. and Risse, T. (2009). Combining global optimization with local selection for efficient qos-aware service composition. In *WWW*, pages 881–890.

Barbuceanu, M. and Fox, M. S. (1996). The design of a coordination language for multi-agent systems. In *ATAL*, pages 341–355.

Brandic, I., Pllana, S., and Benkner, S. (2008). Specification, planning, and execution of qos-aware grid workflows within the amadeus environment. *Concurrency and Computation: Practice and Experience*, 20(4):331–345.

Canfora, G., Penta, M. D., Esposito, R., and Villani, M. L. (2008). A framework for qos-aware binding and rebinding of composite web services. *Journal of Systems and Software*, 81(10):1754–1769.

Charif, Y., Stathis, K., and Mili, H. (2010). Towards anticipatory service composition in ambient intelligence. In *NOTERE*, pages 49–56.

Guan, Y., Ghose, A. K., and Lu, Z. (2006). Using constraint hierarchies to support qos-guided service composition. In *ICWS*, pages 743–752.

Hong, L. and Hu, J. (2009). A multi-dimension qos based local service selection model for service composition. *JNW*, 4(5):351–358.

Jaeger, M. C. and Ladner, H. (2006). A model for the aggregation of qos in ws compositions involving redundant services. *JDIM*, 4(1):44–49.

Kona, S., Bansal, A., Blake, M. B., Bleul, S., and Weise, T. (2009). Wsc-2009: A quality of service-oriented web services challenge. In *CEC*, pages 487–490.

Lajmi, S., Ghedira, C., Ghédira, K., and Benslimane, D. (2006). Wescocbr: How to compose web services via case based reasoning. In *ICEBE*, pages 618–622.

Maamar, Z., Mostéfaoui, S. K., and Yahyaoui, H. (2005). Toward an agent-based and context-oriented approach for web services composition - appendices. *IEEE Trans. Knowl. Data Eng.*, 17(5).

Mukhija, A., Dingwall-Smith, A., and Rosenblum, D. S. (2007). Qos-aware service composition in dino. In *ECOWS*, pages 3–12.

Rosenberg, F., Celikovic, P., Michlmayr, A., Leitner, P., and Dustdar, S. (2009). An end-to-end approach for qos-aware service composition. In *EDOC*, pages 151–160.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Computers*, 29(12):1104–1113.

Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S., and Zhang, J. Y. (2007). Autonomous service level agreement negotiation for service composition provision. *Future Generation Comp. Syst.*, 23(6):748–759.

Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Trans. Software Eng.*, 30(5):311–327.