# Using CBR Method for Multi-Agent Selection of Multiple and Dynamic Composite Web Service

Fatma Siala
*University of Tunis*
*SOIE - ISG*
*Tunis, Tunisia*
*fatma.siala@gnet.tn*

Khaled Ghedira
*University of Tunis*
*SOIE - ISG*
*Tunis, Tunisia*
*khaled.ghedira@isg.rnu.tn*

***Abstract*—It is well-known that Web service technologies provide an easy way to integrate the applications within and across organizational boundaries. Web services are usually overlapping in functionality and how to make a choice based on non-functional factors becomes a problem that needs to be solved. This paper, argues that the selection of component services should be considered in a global manner based on the Web services availability and the users QoS preferences. Indeed, QoS becomes one of the most important factors for Web service selection. 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. This paper presents a framework which deals with the selection of composite Web services on the base of Multi-Agents negotiation and CBR (Case Based Reasoning) method. The objective of the agents is to find out the best Composite QoS (CQoS) based on Web services availability and elementary Web services QoS. By using CBR method, agents can memorize QoS scores. This framework supports different combinations and execution paths. The proposed Multi-Agents framework is compared to an existing approach in terms of execution time. Experiments have demonstrated that our framework provides reliable results in comparison with the existing one.**

***Keywords*-Web service; QoS; Multi-Agent System; Contract-Net Protocol; execution paths; availability; CBR technique.**

## I. INTRODUCTION

Economical context impacts companies and their Information Systems (IS). Companies acquire other competitors or develop new business skills, delocalize whole or part of their organization. Their IS are faced to these complex evolutions and have to overcome these changes. In this context, Service Oriented Architecture (SOA) offers a great flexibility to IS. Applications are seen as black boxes independently connected to an application as Enterprise Application Integration bus (EAI) with its connectors. However, this integration solution does not allow connecting heterogeneous applications or infrastructures. Web services (WS) are based on standards and they are the cheaper and simplest solution to resolve this problem.

Service-Oriented Architecture (SOA) consists of a set of design principles which enable defining and composing interoperable services in a loosely coupled way. The value of SOA lies in assuring that such compositions are easily and rapidly possible with low costs. Thus, service composition is a key to SOA [22].

Yet, with the explosion of Web services available through out the Internet, it's not easy for the end users to composite the Web services manually to meet their specific preferences.

Quality of Web Service (QoS) has become a central criterion for differentiating competing service providers considering the increasing number of services with similar functionalities. The current service optimization paradigm assumes that precise QoS values are available for selecting the competing service providers [6], [22].

Moreover, a composite service can be represented by a statechart which has multiple execution paths 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 art in technologies for Web service composition by first, taking into account the user's preferences, second, by using agents to negotiate the execution path and combination offering the best QoS value, finally, by adding to agents the capability to memorize the QoS and the availability of each Web service.

By negotiating only with available Web services providers fulfilling the QoS user requirements, we obtain a better Central Processing Unit (CPU) time. The information about the QoS is memorized in a cases base. This framework improves the existing approaches [17] [16] in terms of CPU time when the agents can memorize the Web service availability and QoS.

Our contributions are revealed when negotiating only with available Web services providers and fulfilling the QoS requirements. We are able to memorize the QoS information by using CBR (Case Based Reasoning) method. This contribution gives a better Central Processing Unit (CPU) time and also supports different execution paths and combinations for a composition. The combination concept has never been addressed by any approach.

 ISBN: 978-1-61208-200-4 

The structure of this paper is as follows: The coming section presents the major related works in the area of Web service composition based on QoS. Section 3 proposes the framework prototype based on Multi-Agent System and using CBR technique. Section 4 explains the implementation of this framework using a case study. Accordingly, Section 5 presents the experimentation results. After that, the approach is discussed in Section 6 by presenting existing approach limitations and detailing our contributions. Finally, we conclude the whole paper.

## II. RELATED WORKS

Query optimization, taking into account QoS requirements, on Web services have received considerable attention in the service computing community [3] [14] [11]. In this section, we review selected works based on their relevance for our approach.

Zeng et al. [23] 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. [7] 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. [15] 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. [4] 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 [8] 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 [1] 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. [20] 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.

Most 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. This CPU time improvement is also due to using CBR method. In fact, agents memorize QoS scores for further user.

Our approach allows to find the CQoS that fulfill the user requirements by considering different execution paths and combinations and by also taking into account the dynamical aspect of the Web (the Web service availability and QoS scores changes). Our framework takes a largely better CPU time than the existing approaches.

## III. THE PROPOSED FRAMEWORK

The first step toward 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 the technique described in [12] based on CBR method.

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. Our framework allows to select the best elementary Web services in terms of QoS using CBR method and based on Web services availability and supports different execution paths and combinations.

Several motivations lead to use Multi-Agent Systems (MAS) [2]. In fact, a Web service suffers from 3 main

 ISBN: 978-1-61208-200-4 

deficiencies : it acknowledges only itself, passive until it is invoked, has only a knowledge of itself and neither adaptable nor able to benefit of new capabilities of the environment [5]. We propose to represent each service by an agent. In the MAS field, negotiation is a fundamental aspect 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 recognize the available providers which are also represented with agents and the QoS associated to each of them using CBR technique.

We proposed in [17] a first framework, named Multi-Agent Availability (MAA), which optimizes the QoS criteria for a composite service provision and improves an existing approach [20] in terms of CPU time based on Web service availability. We have also proposed in [16] a second framework , named Multi-Agent Availability by exploring multiple execution Paths (MAAP) which improves the first one by considering different execution paths and combinations for a given statechart.

We compare our proposed frameworks with Yan et al.'s one [20]. To the best of our knowledge, their work is the sole which uses agent technology. Maamar et al. [13] have used MASs for Web service composition but they don't consider the QoS.

The MAA framework improves the work of [20] 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, the MAAP framework improves the work of [20] and the MAA framework in terms of QoS by supporting multiple execution paths and combinations for a composition. The MAAP framework takes more CPU time but we 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 MAA framework optimizes the QoS at the local level and verifies after that if it ensures the QoS at the global level (CQoS). On the other side, the MAAP framework, by considering different execution paths and combinations, has more chances to ensure the QoS at the global level.

In this context, we propose a new framework named Multi-Agent Availability by exploring multiple execution Paths based on QoS ($MAAP_Q$). Experiments are conducted to prove the effectiveness of our approach when the agents well also known the QoS (using CBR method) of each Web service with their availability.

### A. $MAAP_Q$ : An agent based Framework

To better explain our approach, we present in Fig 1 the framework ($MAAP_Q$). This framework consists of an Interface Agent (IA), a Combination Coordinator Agent (CCA) and a set of Negotiator Agents (NAs) that negotiate with a set of Web Service Agents (WSAs).

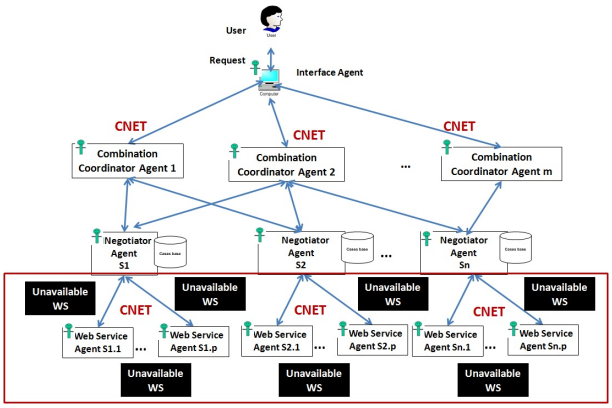


Figure 1. $MAAP_Q$'s Framework.

In the MAAP the negotiation process is as follows: First, the user specifies the desired composite Web service and the associated requirements. Then, The IA charges each CCA for an execution path or combination of the statechart. The CCA 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. 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 propose $MAAP_Q$ as an improvement of this framework when each NA posteriori knowns the WSA's QoS. So, we gain in terms of CPU time since the NA will not negotiate via the Contract-Net (CNET) protocol with WSAs (the providers) to find the best elementary Web service but prepare this information by a negotiation that takes place before this process. The NAs will only verify if this WSA fulfill the same QoS and respecting the user requirements and send the response to the IA which evaluates the results and either confirms the acceptance or repeat the negotiation.

We explain in the following the role of each agent.

*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.

 ISBN: 978-1-61208-200-4 

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 [23]. The IA will then, provide the expected service with the required QoS criteria.

*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 [23] when it receives their responses. Finally, the CCA negotiates with the IA using the CNET protocol to provide its proposal.

*3) The Negotiator Agent:* A Negotiator Agent (NA) is responsible of a Web services set offering the same functionality. The NA uses CBR to have information about WSAs that check user's preferences.

If there is no case in the CBR that fulfill the user's requirements, the NA 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's acquaintances are the available Web service providers. For a negotiation, the NA must consult its list containing available Web services. This negotiation takes place before beginning the negotiation with the user. Each NA has its cases base for fulfillment with the QoS value associated with each QoS criterion.

*4) The Web Service Agent:* A Service Level Agreement (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 1, there are unavailable Web services represented by agents that will not negotiate with the NA.

### B. How to apply CBR in $MAAP_Q$ ?

Using CBR by the NA consists of three steps the case representation, the case research and the case update.

*1) Case representation:* In case-based reasoning, an existing solution should have some similarity with the problem that is being solved in order to be reused. The $MAAP_Q$ considers properties associated to the available WSAs when matching it with existing instances of plans. These properties are the QoS' scores associated to each criterion of QoS (table I). Hence, to match with a Web service that fulfill a user requirements, an existing WSA must have the same or better value. Thus, checking that two WSAs are the same is to check whether they have the same QoS' scores. To make the matching of the WSAs efficient, a function defined in [21] is used to generate the digest of the BPEL file describing the workflow of a composite service. Thus, by comparing the digests of scores of two Web services, we can decide whether the Web service fulfill the user requirements. The NA will computes the QoS score based on the value associated to each criterion and the user preferences.

The instances of QoS scores stored in the CBR repository comply with Kolodner's work [10]. Each instance consists of three elements:

- Problem: A Web service;
- Solution: A selected WSA;
- Evaluation: the QoS score.

Table I
CASE REPRESENTATION.

| | $QoS_1$ | $QoS_2$ | ... | $QoS_n$ |
|---|---|---|---|---|
| $WSA_i$ | $value_1$ | $value_2$ | ... | $value_n$ |

*2) Case research:* When a NA wants to select a WSA, it first checks the CBR repository to find out the WSAs that fulfill the user's requirements. The NA searches the request in the table I). If there are matching WSAs the NA computes its associated QoS fulfilling he user's preferences. The NA chooses the best provider (that fulfill the QoS requirements) and verifies its availability and if it fulfills the same values of QoS. In this case, The NA returns the result to the CCA. The NA considers 10 percent of the cases. Otherwise, if there are no cases that satisfy the user's request in the CBR, the NA negotiates with the WSAs via the contract-Net protocol.

*3) Case update:* After each negotiation between the NA and the WSAs, the NA updates the CBR with the new case for possible future use.

### C. The Negotiation

*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 [19], 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

 ISBN: 978-1-61208-200-4 

further wishes to optimize a function that characterizes this task. We use the function proposed by [23]. For a given task, any number of participants may respond with a proposal, the rest (agents that cannot respect QOS requirements) must refuse.

*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 combination or an execution path of a composite Web service with its QoS weights to the CCAs. The second task that the IA performs is to confirm negotiation results. As the extended negotiation protocol suggests, when various CCAs 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 [23]. Each CCA charges the NAs to find the best elementary Web services.

If the NAa does not find in the cases base WSAs that fulfill the user requirements, it 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 blocks 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 CCA that the current deal is acceptable. Subsequently, the CCA acknowledges the acceptance of the proposal to the NAs that must inform the selected providers (WSAs). When the user finishes using Web services, the IA informs the NA to unlock the selected WSA.

In case 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 CCA to re-start negotiation. Each corresponding NA then sends the modified CFP to WSAs and begins a new negotiation process

This research refers to the QoS model presented in [23] which proposes a formula to compute the overall QoS score for each Web service.

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.

## IV. 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 (Java Agent Development Framework) [9] which is a middleware that implements an agent platform and a development framework. This framework supports CBR and 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 2.

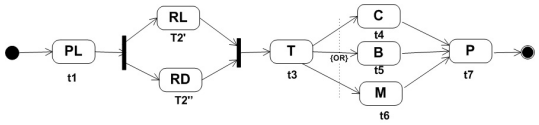


Figure 2. Example of a statechart.

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. A Phone Location Service ($S_{PL}$), a Route Calculation Service ($S_{RL}$), a Route Description Service ($S_{RD}$), a Traffic Service ($S_T$), a Car Service ($S_C$), a Bus Service ($S_B$), a Metro Service ($S_M$) and a Metro Service ($S_P$). 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 above 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. These weights will be used for the computation of the QoS score of each elementary Web services using a formula proposed in [23]. If the user does not specify weights, the system will consider a weight value = 0.25 for each criterion.

For a user's request, each IA sends at the same time a CFP to the CCAs for an execution path or combination in the statechart of the composition. In our case, we have three execution paths. Each NA (height NAs) associated to an elementary Web service checks the QoS' user requirements with the WSAs existing in its cases base. For example, if the user specifies the following weights values : price=0.15, duration=0.35, success rate=0.40 and reputation=0.10. The NA will apply the formula proposed by [23] to choose the best WSA. This formula is based on the QoS scores and their associated weights. Table II represents an example of different values offered by various WSAs associated to $S_{PL}$.

After that, each NA verifies if these providers are again available and fulfill the user requirements in term of QoS.

 ISBN: 978-1-61208-200-4 

If this verification is true, the NA returns the result to the CCA. The NA considers 10 percent of the cases. Else, (if there is no cases that satisfy the user's request in the CBR or the WSAs are not available), the NA negotiates with the WSA via the Contract Net protocol. In this case we use the MAAP process [16].

Table II
EXAMPLE OF CASE REPRESENTATION.

| WSA, QoS | *price* | *duration* | *success* | *reputation* |
|---|---|---|---|---|
| $S_{PL1}$ | 40 | 55 | 70 | 80 |
| $S_{PL2}$ | 60 | 65 | 69 | 73 |
| $S_{PL3}$ | 70 | 68 | 64 | 71 |

Negotiation is as follows: Each NA communicates with providers simultaneously. The negotiation results for each service are summarized in Table III. In all cases, 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 comprehensively simulated using the prototype.
The following results are confirmed: $S_{PL}$: 66, $S_{RL}$ : 98, $S_{RD}$ : 80, $S_T$ : 79, $S_C$:87, $S_B$ : 73, $S_M$ : 85 and $S_P$ : 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 [23].

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

Table III
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 |

## V. EXPERIMENTATION

The series of tests were conducted to compare the CPU time of $MAAP_Q$ framework with MAA.

In the experimentation, we have calculated the CPU time for each approach by varying the number of elementary services in a composition from 5 to 50 with steps of 5 and varying the number of service providers from 10 to 50 with steps of 10. We have calculated the CPU time 10 times for each case and we have considered the average. We present via a 3D chart in Figure 3 the results of these experiments. The results of *MAA* are represented in blue and the result of $MAAP_Q$ approach are represented in green. These experiments show that the $MAAP_Q$ takes a far better result than *MAA*. For example, if the number of Web services providers is equal to 50 and the number of elementary services in a composition is equal to 30, the $MAAP_Q$ takes 2300 ms but *MAA* takes 3035 ms. We gain in terms of CPU time.

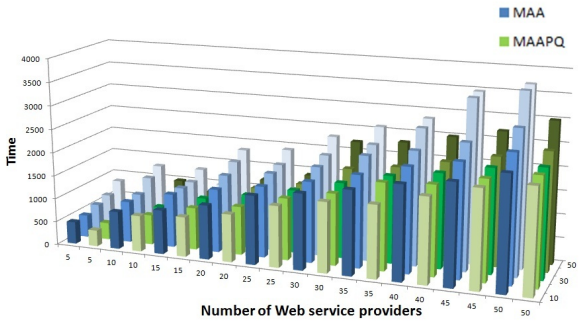


Figure 3. Comparison of CPU time.

We show through Figure 4 the gain percentage of the $MAAP_Q$ compared to the MAAP framework (using formula 1).

$$Gain = \frac{CPU_{time}(MAA) - CPU_{time}(MAAP_Q)}{CPU_{time}(MAA)} * 100 \qquad (1)$$

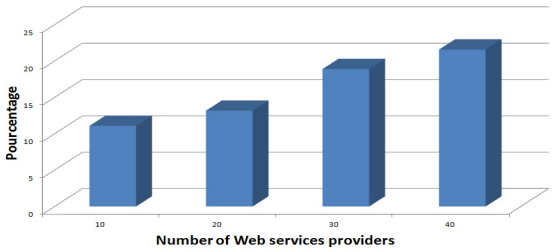


Figure 4. Gain Percentage.

## VI. DISCUSSION

The MAA 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 MAAP framework, by considering different execution paths and combinations, has more chances to ensure the QoS at the global level. By using CBR method, agents memorize QoS scores for further user. this technique allows us to gain in term of CPU time. It is a method to limit the research space.

By using the CBR method, NAs have knowledge about the WSAs. So, we gain the time of conversation. The $MAAP_Q$ not only offers a QoS that fulfill the user requirements but also takes a largely lower CPU time than these frameworks.

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

 ISBN: 978-1-61208-200-4 

## VII. CONCLUSION

This paper exploited the agent technology to select composite Web service using CBR method. This technique allows agents to memorize the QoS scores for further use. The system reduces the amount of time spent on solving a service composition problem by reusing previous selected Web services. The experiment shows that, when there are sufficient amount of Web services in the CBR repository, the proposed system can outperform the existing approaches.

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. By using CBR method we gain in terms of CPU time.

The greatest limitation of this framework is its lack of scalability. Therefore, we present in [18] a new scalable framework using Case Based Reasoning.

In the future work, we will propose a new approach which improves again the execution time by discharging the NAs of many tasks and by adding horizontal communications.

## REFERENCES


[1] Mohammad Alrifai and Thomas Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *WWW*, pages 881–890, 2009.

[2] Mihai Barbuceanu and Mark S. Fox. The design of a coordination language for multi-agent systems. In *ATAL*, pages 341–355, 1996.

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

[4] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. A framework for qos-aware binding and re-binding of composite web services. *Journal of Systems and Software*, 81(10):1754–1769, 2008.

[5] Yasmine Charif, Kostas Stathis, and Hafedh Mili. Towards anticipatory service composition in ambient intelligence. In *NOTERE*, pages 49–56, 2010.

[6] Francisco Curbera, Bernd J. Krämer, and Mike P. Papazoglou, editors. *Service Oriented Computing (SOC), 15.-18. November 2005*, volume 05462 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[7] Ying Guan, Aditya K. Ghose, and Zheng Lu. Using constraint hierarchies to support qos-guided service composition. In *ICWS*, pages 743–752, 2006.

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

[9] JADE. Telecom italia lab. In *http://sharon.cselt.it/projects/jade*, 2011.

[10] J. L. Kolodner. Case-based reasoning. In *Morgan Kaufman*, 1993.

[11] Srividya Kona, Ajay Bansal, M. Brian Blake, Steffen Bleul, and Thomas Weise. Wsc-2009: A quality of service-oriented web services challenge. In *CEC*, pages 487–490, 2009.

[12] Soufiene Lajmi, Chirine Ghedira, Khaled Ghédira, and Djamal Benslimane. Wescocbr: How to compose web services via case based reasoning. In *ICEBE*, pages 618–622, 2006.

[13] Zakaria Maamar, Soraya Kouadri Mostéfaoui, and Hamdi Yahyaoui. Toward an agent-based and context-oriented approach for web services composition - appendices. *IEEE Trans. Knowl. Data Eng.*, 17(5), 2005.

[14] Arun Mukhija, Andrew Dingwall-Smith, and David S. Rosenblum. Qos-aware service composition in dino. In *ECOWS*, pages 3–12, 2007.

[15] Florian Rosenberg, Predrag Celikovic, Anton Michlmayr, Philipp Leitner, and Schahram Dustdar. An end-to-end approach for qos-aware service composition. In *EDOC*, pages 151–160, 2009.

[16] Fatma Siala and Khaled Ghédira. A multi-agent selection of multiple composite web services driven by qos. In *CLOSER*, pages 675–684, 2011.

[17] Fatma Siala and Khaled Ghédira. A multi-agent selection of web service providers driven by composite qos. In *ISCC*, pages 55–60, 2011.

[18] Fatma Siala, Soufiene Lajmi, and Khaled Ghédira. Multi-agent selection of multiple composite web services based on cbr method and driven by qos. In *iiWAS*, pages 90–97, 2011.

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

[20] Jun Yan, Ryszard Kowalczyk, Jian Lin, Mohan Baruwal Chhetri, SukKeong Goh, and Jian Ying Zhang. Autonomous service level agreement negotiation for service composition provision. *Future Generation Comp. Syst.*, 23(6):748–759, 2007.

[21] Xinfeng Ye and Rami Mounla. A hybrid approach to qos-aware service composition. In *ICWS*, pages 62–69, 2008.

[22] Tao Yu, Yue Zhang 0001, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *TWEB*, 1(1), 2007.

[23] Liangzhao Zeng, Boualem Benatallah, Anne H. H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Trans. Software Eng.*, 30(5):311–327, 2004.


 ISBN: 978-1-61208-200-4 