Fuzzy Logic Based QoS Optimization Mechanism for Service Composition

Silvana De Gyvés Avila, Karim Djemame
School of Computing
University of Leeds
Leeds, UK
e-mail: {scsdga, scskd}@leeds.ac.uk

Abstract— **Increase emphasis on Quality of Service and highly changing environments make management of composite services a time consuming and complicated task. Adaptation approaches aim to mitigate the management problem by adjusting composite services to the environment conditions, maintaining functional and quality levels, and reducing human intervention. This paper presents an adaptation approach that implements self-optimization based on fuzzy logic. The proposed optimization model performs service selection based on the analysis of historical and real QoS data, gathered at different stages during the execution of composite services. The use of fuzzy inference systems enables the evaluation of the measured QoS values, helps deciding whether adaptation is needed or not, and how to perform service selection. Experimental results show significant improvements in the global QoS of the use case scenario, providing reductions up to 17.1% in response time, 17.38% in cost and 40% in energy consumption.**

Keywords - Web service composition; adaptation; fuzzy logic; optimization; Quality of Service.

I. INTRODUCTION

Web services are modular, self-contained and reusable software components that rely on open XML-based standards to support machine-machine interactions over distributed environments [1]. Some of the benefits offered by services include time/cost reduction during software development and maintenance. When a single service does not accomplish a consumer's requirement, different services can be used in conjunction to create a new value-added service, known as composite service, to fulfil this requirement.

A composite service provides a new software solution with specific functionalities and can be seen as an atomic component in other service compositions or as a final solution to be used by a consumer [2]. The process of developing a composite Web service is called service composition.

In service composition, it is necessary to have a set of available services that offer certain functionality and also fulfil Quality of Service (QoS) constraints [3]. QoS properties refer to non-functional aspects of Web services, such as performance, reliability, scalability, availability and security [4]. By evaluating the QoS aspects of a set of Web services that share the same goals, a consumer could identify which service meets the quality requirements of the request.

The QoS attributes of a service can be evaluated during design and execution time. At design time, these attributes help in order to build a composite service based on the QoS requirements of the user. While at execution time, they can be monitored to maintain the desired QoS level. Information about these attributes can be obtained from the service's profile [5], nevertheless, when this information is not available, it can be obtained by analyzing data collected from past invocations [6].

The dynamic nature of the Web service execution environment generates frequent variations in the QoS offered to the consumers, therefore, obtaining the expected results while running a service is not guaranteed. Web services, must be capable to adapt in response to their perception of the environment and their own behaviour, without compromising their efficiency. Composite services should be able to adapt, also based on their components performance, in order to provide the consumer the expected behaviour and result on the request.

Due to service composition nature and the variability of the environment where services are executed, different approaches and tools have been proposed not only to enable automatic-dynamic composition, but also to mitigate the impacts of unexpected events during the execution of composite services. Among them, self-adaptive proposals have stood out since they aim to maintain functional and quality levels, by dynamically adapting composite services to the environment conditions reducing human intervention.

Adaptive mechanisms provide software systems with capabilities to self-heal, self-configure, self-optimize, self-protect, etc., considering the objectives the system should achieve, the causes of adaptation, the system reaction towards change and the impact of adaptation upon the system [7]. Currently, work in self-optimization for service composition has been mainly focused on the selection of services at runtime, in order to maintain the expected QoS of the entire composition. However, it is only being considered situations where QoS decreases.

This paper introduces a self-optimization solution for service composition based on fuzzy logic. Fuzzy logic is an approximate reasoning technique suitable to deal with uncertainty [8], which can be use used to support decision making and to evaluate imprecise parameters in software systems. The proposed optimization model performs service selection based on historical QoS data and real data, which is collected at runtime during different stages of the composite service execution. Composite services are considered to be workflows conformed by tasks. The QoS of each task is evaluated previous to the selection of the service that will be linked to the following task. The use of fuzzy support systems enables the evaluation of the measured QoS values, helps deciding whether adaptation is needed or not, and how to perform service selection. The approach has been implemented in a framework and was evaluated empirically by analyzing the execution through a use case, also comparing results with a non-fuzzy approach.

The major contributions of this paper are:

- The optimization model for service composition that analyzes global QoS in order to determine the benefit of adaptation, considering situations where QoS values increases and/or decreases.
- The use of fuzzy logic as a decision making tool to determine the need of adaptation during composite service executions.

The remainder of the paper is structured as follows: background is briefly described in Section II. Section III presents the approach overview. The proposed framework, service selection and optimization models are described in Section IV. Section V presents the experimental description and results. Section VI discusses some related work. Conclusions and future work are given in Section VII.

II. BACKGROUND

A. Adaptation in Service Composition

To experience an expected behaviour during the execution of a composite service, it is important to consider the QoS aspects of the services involved, as their drawbacks will be inherited by the composite service. However, unexpected events occur, e.g., services become unavailable or exhibit discrepancies in their QoS [9], bringing the need of mechanisms such as adaptation, in order to restore and maintain the functional and quality aspects of the composition. Various aspects that can be considered as part of adaptation solutions in service composition are listed as follows [10]:

- Adaptation goal is the purpose of adaptation. Adaptation goals can be defined based on functional and/or non-functional (quality of service) requirements.
- Adaptation level defines those elements that will change in order to achieve the adaptation goal.
- Adaptation actions are those used to solve the adaptation problem.
- Adaptive mechanisms correspond to the approaches applied to implement the adaptation actions (e.g. agent-based [11], policy-based [12], rules-based, feedback-based [13], etc.).
- Stage of adaptation is the time when adaptation is performed (development time, compile/link time, load time and runtime).
- Awareness levels describe the scope of information that will be available in order to perform adaptation [14].

Besides these aspects, it is also important to consider the set of self-* properties that can be selected and implemented in adaptive SOA systems. Self-* properties are related to the objectives of the composition and the causes and impact of adaptation. Some self-* properties applied in service composition approaches are self-healing, self-optimizing and self-configuring.

Self-healing services can monitor themselves, predict/detect the causes of failure and make the adjustments to restore their states to normal [15]. Self-healing is related to availability, survivability, maintainability and reliability [16]. Self-optimizing systems have the ability to select the best available services, as part of a composition, and define the most appropriate QoS levels in order to maximize benefits and reduce cost [17]. Self-optimization is related with efficiency and functionality [16]. Self-configuring services can leverage services and resources to compose an optimal configuration based on user requirements and the characteristics of the system [18]. Self-configuration is related to maintainability, functionality, portability, and usability [16].

B. Fuzzy Logic

Fuzzy logic is a method based on multi-valued logic which aims to formalize approximate reasoning [8]. It is used to deal with different types of uncertainty in knowledge-based systems.

Some of the relevant characteristics of fuzzy logic are fuzzy sets, linguistic variables and fuzzy rules. A fuzzy set is a collection of objects characterized by a membership function with a continuous grade of membership which can be ranged between zero and one [19]. A linguistic variable is a type of variable that uses words instead of numbers to represent its values (e.g. slow, medium, fast) [8]. The values used to define linguistic variables are called terms and the collection of terms is called term set.

Fuzzy rules (IF-THEN) are used to represent human knowledge in fuzzy systems. A fuzzy IF-THEN rule is a conditional statement structured as [20]:

IF <*fuzzy proposition*>, THEN <*fuzzy proposition*>.

Where a <*fuzzy proposition*> is a statement used to associate linguistic variables and terms.

During the execution of a fuzzy system, crisp inputs are converted to linguistic variables, this process is known as fuzzification. The variables values are then evaluated using fuzzy rules, generating the linguistic values for the outputs. Finally, the defuzzification method uses these values to obtain crisp outputs values.

III. APPROACH OVERVIEW

Adaptation mechanisms aim to target situations where the behaviour of a composite service is deviated from what the consumer is expecting. Nevertheless, triggering adaptation after every variation in the behaviour of the composition will not warranty the best possible QoS values. Reason why it is important to consider the following questions: Is adaptation needed? When does the composite service need to adapt? What is the benefit of adaptation? What is the cost of adaptation?

Aiming to give an answer to some of these questions, in this work it is proposed the use of fuzzy logic as a tool to support the decision making process, helping determining whether adaptation is needed or not and how to perform the service selection process.

The approach uses two fuzzy support systems. The first system assesses the QoS values of the composite service on each step of the composition, using the global QoS measured after the execution of the previous task and historical QoS data. The system takes the QoS parameters as inputs and based on fuzzy rules provides the benefit of adaptation.

The second system is used to determine the weights to apply to the different QoS attributes during the service selection process. It uses the value of the benefit of adaptation and the errors between the estimated and the measured QoS as inputs, providing as a result the values for the weights to be used during the service selection process. Both systems will be explained in detail as part of the optimization model in the next section.

IV. SYSTEM MODEL

The implementation and evaluation of the proposed approach requires to setup an environment in which QoS aware and adaptive composition can be executed. The system model illustrated in Fig. 1 has been developed with this purpose. Its core components are described as follows:

- Service Binder: binds dynamically each of the tasks in the composition to executable services. These services are selected using functional and QoS criteria.
- Service Selector: by using required functional and quality information, this module searches in the service registry for those elements that fulfil functional and quality requirements.
- Predictor: obtains estimates for the QoS attributes of the selected services by using predictive algorithms and a collection of historical QoS data.
- Sensors: collect information about different events at run time and send it to the adaptation module. Events are related to quality aspects of the involved compositions' elements.
- Adaptation module: monitors and analyzes the behaviour of composite services at runtime and according to its analysis, determines when it is needed to perform certain changes in order to improve/maintain the offered QoS of the compositions.
- Effectors: apply the actions provided by the adaptation module, enabling composite services to adapt at runtime.
- Composition engine: executes the composite services (processes' definitions).

Composite services are considered to consist of a series of abstract tasks that will be linked to executable services at runtime. To obtain these services, for each task the service binder invokes the service selector (SS) and it requests the

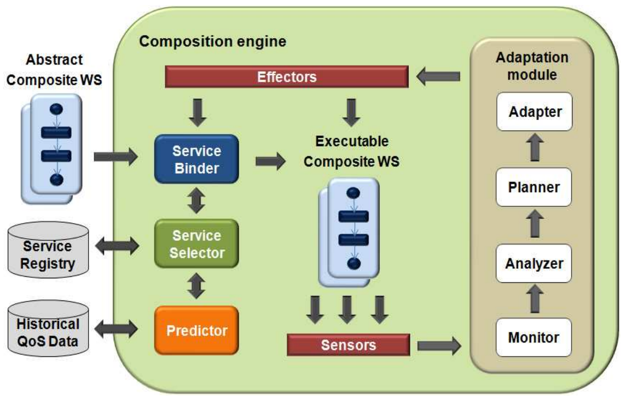

Figure 1. System model.

desired characteristics that the component service should provide.

The SS performs a search into the service registry based on the provided functional requirements. For each of the pre-selected services (candidates), the SS module invokes the predictor to obtain its estimated QoS. The SS compares the results and sends the information about the service that suits the request to the binder.

When the composite service is being executed, sensors capture information about the behaviour of the service and its components, QoS data is being stored in the historical database. Sensors send this information to the adaptation module, which determines if adaptation is needed and the appropriate adaptation strategy. Finally, it sends the actions to be performed to the corresponding effectors, in order to maintain/improve the QoS of the composition.

It is considered that at the time of invoking a composite service, the system has available data from previous executions of the different possible components, in order to obtain accurate predictions about these components' quality characteristics. Also, for each task of the composite service, there exist at least two concrete services to invoke.

A. Service selection model

Different QoS attributes can be associated with Web services [6, 21], which could be used as a differentiating point in the preference of consumers at the time of searching/selecting components for certain application. In this work, the quality parameters that will be considered for each service are response time, cost and energy consumption.

- Response time: time consumed between the invocation and completion of the service operation [22];
- Cost: fee charged to the consumer when invoking a service [10];
- Energy consumption: amount of power consumed by a server over a period of time [23].

Considering response time and cost enables the selection of faster and cheaper services, providing a competitive advantage [6]. Both parameters have been used in other approaches, like those presented in [3, 10, 22, 24].

The amount of energy used by data centres has not only economical but also environmental impacts. Energy efficiency is becoming a key topic due to high energy costs and governments' pressure to reduce carbon footprints [25]. Energy consumption has been selected as the third parameter because of the importance of energy efficiency when managing computing infrastructure and services.

Estimation of QoS values is a key step during service selection process. Estimated values are calculated using historical QoS data recorded from previous executions. This data is filtered, discarding values considered as outliers and the average of the last N executions of the remaining subset is obtained.

Concrete services are searched in the registry by name, assuming that this parameter includes/describes the service's functionality. The resulting set of candidate services is sorted according to the relationship between their estimated QoS values. Due to these attributes having different units of measure, the raw values are first normalized using the following formula:

$$V_i = \frac{max_i - q_i}{max_i - min_i}. \quad (1)$$

Where max_i and min_i correspond to the maximum and minimum values of the evaluated QoS parameter, respectively; and q_i correspond to the estimated value for the next execution. When $max_i = min_i$, then $V_i = 1$.

Results are then computed using the Simple Additive Weighting formula:

$$W_i = t_i\,(w_1) + c_i\,(w_2) + e_i(w_3). \quad (2)$$

Where t_i is the service estimated response time; c_i is the service estimated cost; e_i is the service estimated energy consumption; and w_1, w_2 and w_3 correspond to assigned weights where $w_1, w_2, w_3 \leq 1$ and $w_1 + w_2 + w_3 = 1$.

B. Optimization Model

Monitoring the execution of services is a critical task in the adaptation process. By monitoring and collecting data from services executions, based on their performance it is possible to take decisions about future actions [26].

As part of this work, QoS information is collected from service, task and process perspectives, where service corresponds to concrete Web services; task to elements within the composite service that invoke services; and process to the entire composition (service workflow).

Response time is measured during each stage of the process, while cost and energy consumption are obtained from the WSDL files of the services. The QoS values of a task are registered as an individual invocation and as the accumulated QoS of the composition at the time of executing the task.

The proposed optimization approach uses the service selection model previously described and it is based on fuzzy systems to asses the QoS values of the composition, in order to decide if adaptation is needed or not and to establish the weights to be used during the service selection process.

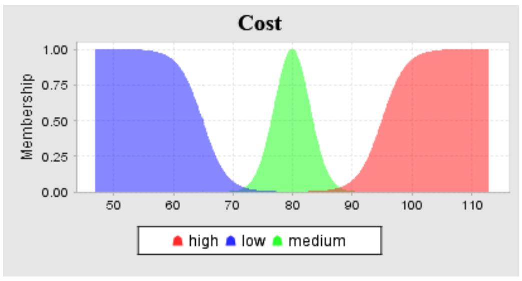

Figure 2. Term set of an input linguistic variable.

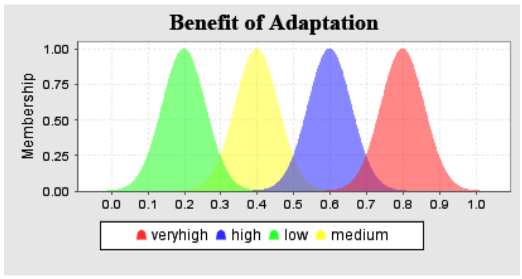

Figure 3. Term set of the benefit of adaptation variable.

It considers situations where a number of the accumulated QoS values of the previous activity in the process are better than expected, providing some slack that can be used while selecting the next service in the process, improving other QoS parameters.

The idea of using fuzzy logic is to understand the relationship between the QoS values of the composite service and the need of adaptation. In this context, QoS parameters can be expressed using linguistic variables.

Two inference engines have been defined to 1) obtain the benefit of adaptation, 2) obtain the weights to be use during service selection. Each of these systems uses its own linguistic variables and rules.

The first system assesses the QoS values of the composite service during each step of the composition. It uses as inputs the QoS values collected from the composite service previous to the moment of selecting a new service.

The defined input variables are response time, cost and energy consumption, which are expressed with three terms low, medium and high. To establish these terms for each of the linguistic variables, an interval is defined at runtime using data collected from previous executions. Historical data is analyzed, obtaining maximum/minimum values and standard deviations from each of the QoS parameters. Sigmoidal functions (open to the left and right) are used to define the low and high terms, while gauss function is used to define the medium term, as illustrated in Fig. 2.

The system takes the inputs and based on the corresponding fuzzy rules, provides the estimated benefit of adaptation. Four different levels of benefit of adaptation (low, medium, high and very high) were established, falling in the interval [0, 1], and defined with gauss functions (see Fig. 3).

TABLE I. BENEFIT OF ADAPTATION RELATED RULE.

```
IF (responseTime IS high AND cost IS low AND energy IS low)
OR (responseTime IS low AND cost IS high AND energy IS low)
OR (responseTime IS low AND cost IS low AND energy IS high)
THEN BenefitofAdaptation IS veryhigh
```

Table I shows one of the rules used to obtain the benefit of adaptation. Four compound rules were constructed combining the input variables and their relationship with the different levels of benefit of adaptation. These rules describe the scenarios that can take place at runtime.

The second system uses the value of the benefit of adaptation (output of the first system) and the errors between the estimated and the measured QoS as inputs. The error value is computed per each parameter using the following formula:

$$e\ (p_i) = \frac{x\ (p_i) - x_0\ (p_i)}{x_0(p_i)}. \quad (3)$$

Where $x(p_i)$ is the estimated data; and $x_0(p_i)$ is the real measured data.

Input variables corresponding to the QoS errors are expressed with three terms, low, medium and high, falling in the interval [-0.5, +0.5]. Benefit of adaptation is expressed with four terms, as defined in the first fuzzy system.

TABLE II. QOS EVALUATION ALGORITHM.

```
Input:
rt → response time
cost→ cost
ec → energy consumption
eRt → response time error
eCost→ cost error
eEc → energy consumption error

Output:
ω → benefit of adaptation
α → response time weight
β →cost weight
γ →energy consumption weight

(1)   Sort by response time
(2)   rt ← Obtain measured response time
(3)   eRt ← Obtain response time error
(4)   Sort by cost
(5)   cost ← Obtain measured cost
(6)   eCost ← Obtain cost error
(7)   Sort by energy consumption
(8)   ec ← Obtain measured energy consumption
(9)   eEc ← Obtain energy consumption error
      //fuzzy system 1
(10)  ω ← Obtain benefit of adaptation
(11)  if ω >= medium then
          //fuzzy system 2
(12)      α ← Obtain response time weight
(13)      β ← Obtain cost weight
(14)      γ ← Obtain energy consumption weight
(15)      Adjust weights
(16)  else
(17)      α ← β ← γ ← 0.333
return α, β and γ
```

By evaluating the different errors and the benefit of adaptation, the system provides the values to be used as weights during the service selection process. Output variables (response time weight, cost weight and energy consumption weight) are expressed with five terms, very low, low, medium, high and very high, falling in the interval [0,1] and are defined using gauss functions.

The algorithm presented in Table II describes the QoS evaluation method applied during optimization, which involves the use of the fuzzy systems previously described.

Before selecting the new service to be invoked, QoS measured values of the previous task are collected and errors are computed (steps 1 to 9). The measured QoS values are used as inputs for the first fuzzy system. The benefit of adaptation is obtained (step 10) and evaluated (step 11); if it is medium or higher then there is a need of adaptation. If not, weights to be used during the service selection are set to 0.333 (step 17).

When adaptation is needed, the system determines the new weights to be use during the service selection process. This action is performed by the second fuzzy system (steps 12 to 14). Weights are then adjusted, to fulfil the restriction $\alpha + \beta + \gamma = 1$ (step 15). Finally, the algorithm returns the weight values α, β and γ (step 18).

V. EVALUATION

In order to asses the effectiveness of the proposed optimization approach, an experimental environment was setup and a composite service was developed as use case. Elements described in Section IV were deployed and configured within this environment.

Experiments were carried out to address the following questions:

- How does the evaluation of the benefit of adaptation influence the adaptation process?
- Is there any improvement in the global QoS when using variable weights during service selection as part of a self-optimization mechanism?

A. Experimental Environment

The experimental environment consists of three nodes, one computer with Windows Vista, 4GB RAM and one Intel core2 duo 2.1GHz processor (node 1); and two virtual machines with lubuntu 11.10, 512 Mb RAM and one processor (node 2 and 3).

Node 1 hosts the BPEL engine (Apache ODE 1.3.4), service registry (jUDDI 3.0.4), historical data base (MySQL 5.1.51) and one application server (Tomcat 6.0.26). Node 2 and 3, host one application server each (Tomcat 6.0.35). Web services are allocated in the application servers.

This environment works in a Local Area Network (LAN). However, in further experiments it is important to perform a detailed analysis of the behaviour of Web services (e.g., faults, availability, latency) over a WAN, in order to obtain results closer to a realistic scenario.

TABLE III. COST EVALUATION ALGORITHM.

```
Input:
nInv → number of invocations
ω → maximum number of invocations
ϕ → minimum number of invocations

(1)   nInv ← Obtain number of invocations
(2)   if nInv >= ω then
(3)       Increase cost
(4)   else if nInv < ϕ then
(5)       Decrease cost
```

B. Dynamic QoS Parameters

To add dynamicity to the test environment, values of the QoS properties must change over time, or between services' executions. This helps to obtain sensible results and also avoids the invocation of only one service per each of the tasks in the composition.

Based on the analysis of the behaviour of Web services found on the Internet, response time of the candidate services was modified by adding random delays generated with a log-normal distribution. The distribution and its input values were determined after executing 5 services 1,000 times, collect their response times and analyze the difference between each execution.

To turn the cost of the different services into dynamic QoS values, a model which affects cost based on demand has been implemented. It is assumed that higher the cost, lower the demand (number of times the service is invoked). The algorithm used to implement the cost model is described in Table III.

The number of times a service has been invoked per a period of N minutes is evaluated continuously. Based on this information, and the values specified as the maximum and minimum number of invocations, it is possible to establish a new cost based on the demand. If the number of invocations is equal or higher than the maximum limit, the cost of the service is increased (3). On the other hand, when the number of invocations is smaller than the minimum limit, the cost of the service is decreased (5).

Each of the servers, where the Web services are executed, is assumed to have different hardware and software configurations (see Table IV). Servers information and their characteristics were selected from the Energy Star report [27].

TABLE IV. POWER CONSUMPTION DESCRIPTION PER NODE.

Server	Hardware	Operative System	Idle (W)	Load (W)
Node 1	Acer Incorporated Gateway GT310 F1	Windows Server 2008 R2 64bit	50.75	129.5
Node 2	Hitachi - HA8000/SS10	Windows Server 2008 R2	45.27	81.97
Node 3	IBM - System X3650 M3	Red Hat Enterprise Linux 5 Update 4 x64 Edition	210.85	388.3

Using the model proposed in [23], which is based on the percentage of CPU usage, it is possible to determine an approximate value to the server energy consumption.

$$P(u) = P_{max} \cdot k + (1-k) \cdot P_{max} \cdot u \quad (4)$$

$$E = \int_t P(u(t)) \quad (5)$$

Where $P(u)$ is the power consumed in an instance of time; P_{max} is the power consumed when the server is fully utilized; u is the utilization level; and k is the fraction of power consumed by the idle server. E is the total energy consumed by a node over a period of time t.

Servers' utilization is considered to be variable over time. The power consumed by a server is obtained periodically and exposed on the WSDL files of the corresponding services; it is computed using (4) and the data presented in Table IV. The energy consumed by a server at the moment the Web service is running, is obtained based on the service'

C. Experiment Description

The test case is a BPEL [28] service that implements a travel planning process. It validates a credit card, performs flight and hotel reservations in parallel, and finally invokes a car rental operation. This service is hosted and invoked from Node 1.

The travel planning service is illustrated in Fig. 4. Per each of the tasks in the process, there are 9 candidate services, distributed among the servers (nodes), that fulfil the required functionality and offer different QoS. These services were previously registered into the service registry (UDDI), and executed several times to populate the historical data base and enable the estimation of their QoS attributes.

The travel planning service was executed 50 times to analyze the behaviour of the optimization approach and evaluate its overall benefit. The benefit of adaptation is evaluated in order to determine whether adaptation is needed, or not.

To get a clear understanding on how the evaluation of the benefit of adaptation and the use of variable weights influence the results of service selection, the use case has also been executed using a non-fuzzy approach, setting weights for the service selection as fixed values corresponding to 0.333.

The experiment was executed using dynamic QoS (based on the dynamic QoS models previously described) and fixed QoS.

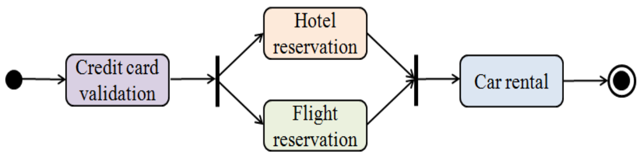

Figure 4. Travel planning process.

D. Evaluation Results

Initial results show that the proposed optimization approach improves the global QoS values of the composition. Global QoS refers to the final values of the different QoS properties (response time, cost and energy consumption).

The following plots show a comparison between the proposed approach and a non-fuzzy approach for each of the QoS parameters. When using the proposed approach, QoS values are dynamic, cost and power consumption change over time based on the models previously described. On the other hand, for the non-fuzzy approach, values for cost and power consumption remain constant. Energy consumption is obtained based on power consumption and response time, using formula (5). For both cases, response time is dynamic.

When analyzing the obtained response time values, it can be noticed that the proposed approach follows a more stable behaviour as compared with the non-fuzzy approach (see Fig. 5). This is due to the evaluation of the QoS values before a new service is selected. The system aims to maintain or if possible, improve the global QoS of the composition. Measured response time values of the proposed approach provide a mean reduction of 4.83% and a highest reduction of 17.1%.

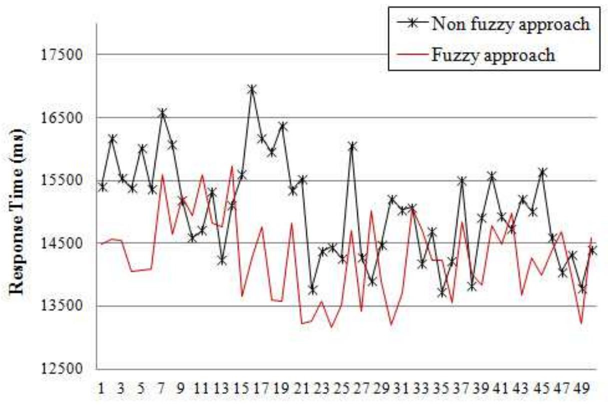

Figure 5. Response time comparison between non-fuzzy and fuzzy approaches.

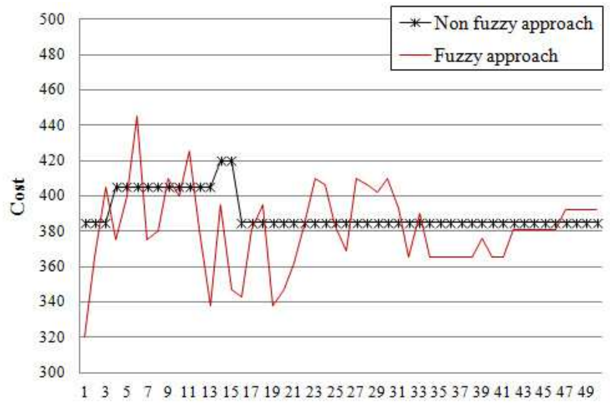

Figure 6. Cost comparison between non-fuzzy and fuzzy approaches.

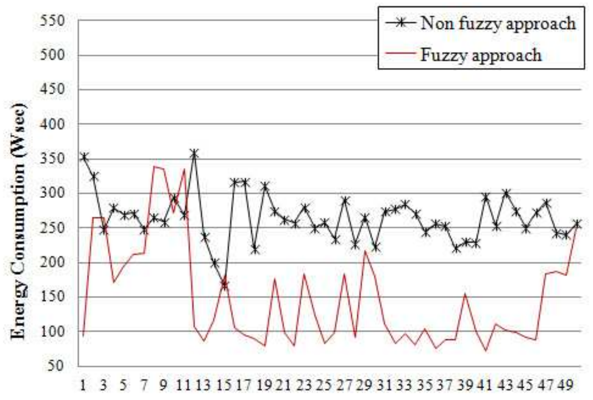

Figure 7. Energy consumption comparison between non-fuzzy and fuzzy approaches.

The obtained cost values are shown in Fig. 6. In comparison with the non-fuzzy approach, the use of the fuzzy-based system provides a mean reduction of 2.25% and a highest reduction of 17.38%.

Results also indicate that there is a significant reduction in the values of energy consumption, as illustrated in Fig. 7, providing a mean reduction of 40%. One important factor to consider is that energy consumption is not only based in power consumption, but also in time. A small response time value may generate a small energy consumption value.

Summarized results of the experiments are presented in Table V. These data was collected using a non-fuzzy approach, and the proposed optimization model with fixed and dynamic QoS values.

TABLE V. POWER CONSUMPTION DESCRIPTION PER NODE.

QoS/Approach		**Non Fuzzy**	**Fuzzy with fixed QoS**	**Fuzzy with dynamic QoS**
Response time (ms)	Max.	16970	17031	15725
	Min.	13732	13892	13166
	Avg.	15037	15258	14281
	Std. Dev.	5.32%	5.07%	4.68%
Cost	Max.	420	400	445
	Min.	385	355	320
	Avg.	390.4	363	381.16
	Std. Dev.	2.58%	2.91%	6.22%
Energy consumption (Wsec)	Max.	359.25	556.73	338.22
	Min.	166.24	261.09	71.44
	Avg.	264.81	421.73	148.25
	Std. Dev.	13.43%	12.53%	50.2%

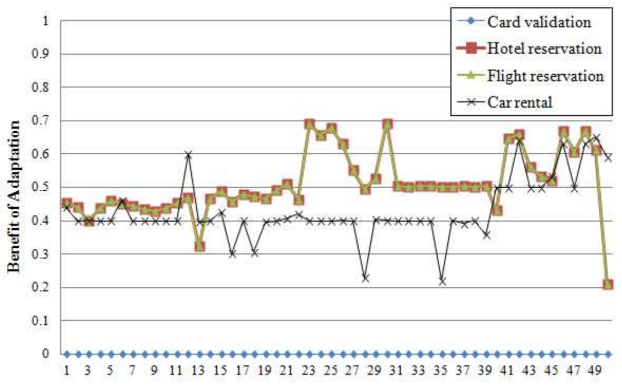

Figure 8. Benefit of adaptation per each task in the travel planning process.

When comparing the obtained results, it can be noticed that use of dynamic QoS has a strong influence in the final QoS of the composite service. Values corresponding to the proposed approach with dynamic QoS present the highest standard deviations for cost and energy consumption. This behaviour is due to the inserted dynamicity. Even though the highest cost is found in the proposed approach column, when it comes to average values, it is still lower than the non-fuzzy results.

The values of benefit of adaptation (BoA) collected per each task of the different executions of the process are illustrated in Fig. 8. These values were obtained using the proposed optimization model with dynamic QoS. For the first task of the process (card validation), as there is no QoS information from previous tasks, the BoA is equal to 0, setting the weights for service selection equal to 0.33. Hotel reservation and flight reservation are executed in parallel after card validation, reason why their BoA values are the same.

Adaptation is performed per task when BoA is larger than 0.4, which is the highest value for the medium term defined in the fuzzy system. It was noticed that in most of the cases where BoA was higher than 0.45 for hotel reservation/flight reservation tasks, BoA values were lower than medium for the last task of the process, therefore, adaptation was not needed.

VI. RELATED WORK

The importance of QoS management in service environments has brought the need of QoS aware solutions for service composition. Different approaches have been presented to evaluate QoS attributes, aiming to select a set of components that optimize the global QoS. Some of these approaches are based on the works described in [6] and [21], which proposed mathematical models to compute QoS of composite services based on the QoS of their components and consider time, cost, reliability, availability and reputation as the quality criteria to evaluate.

By using self-* properties, systems are enabled with capabilities to deal with the dynamicity of the Web service execution environment, providing the consumer with the expected QoS levels and functional results. These properties allow composite services to function despite of environmental changes, detect and react to components that not satisfy the service requirements, and select partners that increase the benefits of the composition.

According to the objectives of the composition and the causes and impact of adaptation, different self-* properties can be selected and implemented. The most used properties in service composition approaches are self-healing [15], self-configuration [18] and self-optimization [17].

Approaches like those presented in [11, 13, 22, 24, 29-32] apply self-healing mechanisms, where new services are selected and invoked after a functional failure or a QoS constraint violation. An adaptation solution that uses self-configuring features is described in [33], where service composition is performed by searching for an optimal configuration of components based upon initial constraints. Adaptation capabilities include runtime reconfiguration, and resource assignment.

On the other hand, mechanisms that implement self-optimization are closely related to the selection of services at runtime, in order to maintain the expected QoS of the entire composition. Examples of works belonging to this category are described in [10, 32-34].

A framework for QoS driven adaptation for service composition is presented in [10]. Adaptation is performed using service selection and coordination patterns. The framework uses an optimization engine to analyze the behaviour, determine an adaptation policy and ensure the composition meets the QoS goals. The solution presented in [32] proposes a QoS-aware binding approach based on Genetic Algorithms. It searches for the best possible set of services to invoke, however, at runtime the bindings can be reconsidered and sections of the composition can change. The framework described in [34] enables designers to develop BPEL workflows, in which they can define at design time the information required to adapt at runtime, including a set of candidate services and constraints. The framework selects the best available services for executing the process and defines the most appropriate QoS levels for delivering them.

Although these approaches are closely related with the work described in this paper, there are significant differences. Firstly, the proposed optimization approach takes into consideration the benefit of adaptation, obtained from the measured QoS values, to determine whether adaptation is needed or not. Secondly, optimization of QoS is also considered when the measured QoS values at certain point of the composite service execution is better than expected, enabling the improvement of other QoS attributes.

Because of the nature of fuzzy logic for solving problems and producing solutions for management purposes, it has been applied in different fields like networks, control systems and mobile applications. In the area of Web services, it has been used as a support tool for service selection, discovery and composition [35-38].

The approach presented in [35] uses two fuzzy systems to select adaptation strategies based on the overall QoS values, importance of QoS and cost of service substitution. A framework that performs matchmaking tasks for dynamic service discovery based on fuzzy logic is described in [36]. In [37], it is presented a fuzzy decision making model to locate and select services based on customer's preference or satisfaction degree. A generic model for representing and evaluating non-functional service properties is proposed in [38]. It aims to enable the selection of service compositions fitting the user's requirements.

The main difference between these approaches and the work presented in this paper is the purpose of the use of fuzzy logic. In the proposed approach, fuzzy logic is used as a tool to evaluate the measured QoS values in order to determine the benefit of performing adaptation.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents an adaptation approach for service composition that implements a self-optimization mechanism, which aims to improve the global QoS values of composite services. The mechanism is based on service selection and fuzzy logic. Fuzzy logic has demonstrated to be a useful tool in the evaluation process of the QoS attributes.

At runtime, the QoS values of the composition are monitored and evaluated in order to obtain the benefit of adaptation. Optimization is triggered if the benefit is considered to be medium or higher. It is applied in situations with QoS decrease, and also where a number of the accumulated QoS values are better than expected, providing some slack that can be used while selecting the next service in the process, improving other QoS parameters.

In summary, evaluation results indicate that by using the proposed approach, there can be achieved significant improvements in the global QoS of composite services. By obtaining and analyzing the benefit of adaptation, adaptation is not carried out each time a QoS value changes. Service selection is performed using variable weights, which influence the preferences on component services and have an impact on the global QoS of the composition.

This paper is part of an ongoing research. Future work includes the analysis of different self-adaptive properties and the extension of the actual framework, in order to increase the coverage of events that can occur at runtime. Also, it is planned to investigate different decision support tools and their efficiency when used to evaluate the benefit of adaptation.

REFERENCES

[1] W3C Working Group. (2004, May, 2012). Web Services Architecture. Available: http://www.w3.org/TR/ws-arch/

[2] S. Dustdar and W. Schreiner, "A survey on web services composition," International Journal of Web and Grid Services, vol. 1, pp. 1–30, 2005.

[3] D. Ardagna and R. Mirandola, "Per-flow optimal service selection for Web services based processes," Journal of Systems and Software, vol. 83, pp. 1512-1523, 2010.

[4] W3C Working Group. (2003, July 2010). QoS for Web Services: Requirements and Possible Approaches. Available: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

[5] S.-Y. Hwang, et al., "A probabilistic approach to modeling and estimating the QoS of web-services-based workflows," Information Sciences, vol. 177, pp. 5484-5503, 2007.

[6] J. Cardoso, et al., "Quality of service for workflows and Web service processes," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 1, pp. 281-308, 2004.

[7] B. H. Cheng, et al., "Software Engineering for Self-Adaptive Systems: A Research Roadmap," Software Engineering for Self-Adaptive Systems, Lecture Notes In Computer Science, vol. 5525, pp. 1-26 2009.

[8] L. A. Zadeh, "The role of fuzzy logic in modeling, identification and control," Modeling, Identification and Control (MIC), vol. 15, pp. 191-203, 1994.

[9] P. Châtel, et al., "QoS-based Late-Binding of Service Invocations in Adaptive Business Processes," in Proceedings of the 2010 IEEE International Conference on Web Services, Miami, USA, 2010, pp. 227-234.

[10] V. Cardellini, et al., "MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems," Software Engineering, IEEE Transactions on, vol. PP, 2011.

[11] L. Wenjuan, et al., "A framework to improve adaptability in web service composition," in Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET), Chengdu, China, 2010.

[12] A. Erradi, et al., "Policy-driven middleware for self-adaptation of web services compositions," in Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, Melbourne, Australia, 2006, pp. 62-80.

[13] D. Bianculli, et al., "Automated Dynamic Maintenance of Composite Services Based on Service Reputation," in Proceedings of the 5th international conference on Service-Oriented Computing (ICSOC), Vienna, Austria, 2007, pp. 449-455.

[14] S. Dustdar, et al., "A roadmap towards sustainable self-aware service systems," in Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, Cape Town, South Africa, 2010, pp. 10-19.

[15] WS-Diamond Team, "WS-DIAMOND: Web Services-DiAgnosability, MONitoring and Diagnosis," MIT press, pp. 213-239, 2009.

[16] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," ACM Transactions on Autonomous and Adaptive Systems, vol. 4, pp. 1-42, 2009.

[17] M. P. Papazoglou, et al., "Service-Oriented Computing: A Research Roadmap," International Journal of Cooperative Information Systems, vol. 17, pp. 223-255, 2008.

[18] A. C. Huang and P. Steenkiste, "Building Self-Configuring Services Using Service-Specific Knowledge," in Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, 2004, pp. 45-54.

[19] L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, pp. 338-353, 1965.

[20] Li-Xin Wang, A course in fuzzy systems and control: Prentice Hall, 1997.

[21] L. Zeng, et al., "QoS-Aware Middleware for Web Services Composition," IEEE Transactions on Software Engineering, vol. 30, pp. 311-327, 2004.

[22] Y. Dai, et al., "QoS-Driven Self-Healing Web Service Composition Based on Performance Prediction," Journal of Computer Science and Technology, vol. 24, pp. 250-261, March 2009.

[23] R. Buyya, et al., "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges," in Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications, las Vegas, USA, 2010.

[24] Y. Ying, et al., "A Self-healing composite Web service model," in Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC), Biopolis, Singapore, 2009, pp. 307-312.

[25] J. Kaplan, et al., "Revolutionizing Data Center Energy Efficiency," McKinsey,July 2009.

[26] A. Erradi, et al., "WS-Policy based Monitoring of Composite Web Services," in Proceedings of the 5th IEEE European Conference on Web Services, Halle, Germany, 2007, pp. 99-108.

[27] Energy Star, "Computer Servers Product List - Families," August, 2012.

[28] OASIS. (2007, June 2010). Web Services Business Process Execution Language Version 2.0. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[29] G. Wu, et al., "Towards self-healing Web Services Composition," in Proceedings of the First Asia-Pacific Symposium on Internetware, Beijing, China, 2009.

[30] D. Ardagna, et al., "A Service-Based Framework for Flexible Business Processes," IEEE Software, vol. 28, pp. 61-67, 2011.

[31] A. Erradi and P. Maheshwari, "Dynamic Binding Framework for Adaptive Web Services," in Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services, Athens, Greece, 2008, pp. 162-167.

[32] G. Canfora, et al., "A framework for QoS-aware binding and re-binding of composite web services," The Journal of Systems and Software, vol. 81, pp. 1754-1769, 2008.

[33] R. Calinescu, et al., "Dynamic QoS Management and Optimization in Service-Based Systems," IEEE Transactions on Software Engineering, vol. 37, pp. 387-409, 2011.

[34] D. Ardagna, et al., "PAWS: A Framework for Executing Adaptive Web-Service Processes," IEEE Software, vol. 24, pp. 39-46, 2007.

[35] B. Pernici and S. H. Siadat, "Selection of Service Adaptation Strategies Based on Fuzzy Logic," in Proceedings of the 2011 IEEE World Congress on Services (SERVICES), Washington DC, USA, 2011, pp. 99-106.

[36] C. Kuo-Ming, et al., "Fuzzy matchmaking for Web services," in Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA), Tamkang University, Taiwan, 2005, pp. 721-726 vol.2.

[37] P. Wang, et al., "A Fuzzy Model for Selection of QoS-Aware Web Services," in Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE), Shanghai, China, 2006, pp. 585-593.

[38] H. Pfeffer, et al., "A Fuzzy Logic Based Model for Representing and Evaluating Service Composition Properties," in Proceedings of the 3rd International Conference on Systems and Networks Communications (ICSNC), Sliema, Malta, 2008, pp. 335-342.