Proactive Adaptation in Service Composition using a Fuzzy Logic Based Optimization Mechanism

Silvana de Gyvés Avila and Karim Djemame
School of Computing, University of Leeds, LS2 9JT, Leeds, U.K.

Keywords: Web Service Composition, Proactive Adaptation, Fuzzy Logic, Optimization, Quality of Service.

Abstract: The importance of Quality of Service management in service oriented environments has brought the need of QoS aware solutions. Proactive adaptation approaches enable composite services to detect in advance, according to their QoS values, the need for a change in order to prevent upcoming problems, and maintain the functional and quality levels of the composition. This paper presents a proactive adaptation mechanism that implements self-optimization based on fuzzy logic. The optimization model uses two fuzzy inference systems that evaluate the QoS values of composite services, based on historical and freshly collected data, and decide if adaptation is needed or not. Experimental results show significant improvements in the global QoS of the use case scenarios, providing reductions of up to 8.9% in response time and 14.7% in energy consumption, and an improvement of 41% in availability; this is achieved with an average increment in cost of 11.75 %.

1 INTRODUCTION

A composite service is a software solution with specific functionalities that can be seen as an atomic component in other service compositions, or as a final solution to be used by a consumer. The process of developing a composite service is called service composition, which consists in combining, in a structured way, the features provided by different services (Dustdar and Schreiner, 2005).

The nature of service composition, dynamicity offered by the environments where services are executed and growing amount of available services (that may provide the same functionality), have brought the need of mechanisms focused in ensuring that the consumer will obtain the expected results when invoking a composite service. To achieve this goal, it is important to consider the QoS aspects of the services involved in the composition, as their drawbacks will be inherited by the composite service. However, knowing the QoS of the components is not enough to warranty the behaviour of the composition, as unexpected events may occur at runtime, for example, services becoming unavailable or showing discrepancies in their QoS (Châtel et al., 2010). As a result, various adaptive mechanisms have been proposed in order to restore and maintain the functional and quality aspects of the composition. The aim of adaptive mechanisms is to provide composite services with capabilities that enable them to morph and function in spite of internal and external changes, searching to maximize the composition potential and reducing as much as possible human involvement (Zeginis and Plexousakis, 2010). Based on the moment when adjustments take place, adaptation approaches are classified as either *reactive* or *proactive*. The former corresponds to adaptation actions performed in response to an incident, while the later is related to actions taken in advance, before an incident impacts the system (Metzger, 2011).

This paper introduces a proactive adaptation approach for service composition that implements a self-optimization solution based on fuzzy logic. Fuzzy logic is an approximate reasoning technique suitable to deal with uncertainty (Zadeh, 1994), which can be used to support decision making in software systems. Current work in proactive adaptation for service composition is mainly focused on dealing with the decrease of the QoS values and service failures (Aschoff and Zisman, 2012, Yuelong et al., 2012, Leitner et al., 2010). On the other hand, approaches related to self-optimization are focused on the selection of services that provide the most appropriate QoS levels for the composition

de Gyvés Avila S. and Djemame K..
Proactive Adaptation in Service Composition using a Fuzzy Logic Based Optimization Mechanism.
DOI: 10.5220/0004820902570267
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 257-267
ISBN: 978-989-758-019-2

(Ardagna et al., 2011, Calinescu et al., 2011, Cardellini et al., 2012).

The proposed optimization model combines the analysis of historical QoS data and fresh data (collected at runtime from the different stages of the composite service execution) in order to identify the need of adaptation, which can be related to prevent a decrease in the global QoS of the composition, but also, the possibility of improving the global QoS levels. Composite services are considered to be workflows formed by tasks, and tasks to be Web service invocations. The use of the fuzzy support systems enables the evaluation of the QoS parameters and helps deciding whether adaptation is needed or not. If adaptation is needed, the fuzzy systems provide the parameters to be used during the service selection process.

The approach has been implemented in a service composition framework and evaluated through the execution of two test cases. Results were compared with a non-adaptive approach. The major contributions for this paper are:

- The optimization approach for service composition that evaluates the benefit of adaptation.
- The use of fuzzy logic as a decision making tool to determine the need of adaptation in the context of proactive adaptation in service composition.

The remainder of the paper is structured as follows: background is briefly described in Section 2. The proposed framework, service selection and optimization models are described in Section 3. Section 4 presents the experimental description and results. Section 5 discusses some related work. Conclusion and future work are given in Section 6.

2 BACKGROUND

2.1 Adaptation in Service Composition

Adaptive mechanisms provide software systems with capabilities to: self-heal, self-configure, self-optimize, self-protect, etc., which are implemented considering the objectives the system should achieve, the causes of adaptation, the system reaction towards change and the impact of adaptation upon the system (Cheng et al., 2009). Adaptation in service composition aims to mitigate the impact of unexpected events that take place during the execution of composite services, maintaining functional and quality of service levels. Important aspects that can be considered as part of adaptation solutions in service composition are listed as follows (Cardellini et al., 2012):

- Adaptation goal is the purpose of adaptation, functional and/or non-functional (QoS).
- 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, policy-based, rule-based, 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 adapt (Dustdar et al., 2010).

2.2 Reactive vs Proactive Adaptation

In service-based applications, reactive adaptation is triggered after problems have occurred, when situations like the use of faulty services or services that present undesirable QoS have already affected the application (Hielscher et al., 2008). The use of reactive mechanisms may cause increases in the execution time and financial loss, which can lead to user and business dissatisfaction (Aschoff and Zisman, 2012). Proactive approaches aim to deal with some of these drawbacks by detecting the need for a change, before reaching a point where a problem may occur.

Situations that can be predicted in proactive adaptation approaches for service composition include: the impact of a new requirement, misbehaviour of a service and the existence of new services (Aschoff and Zisman, 2012). Techniques such as data mining, online testing, statistical analysis, runtime verification and simulation are applied during the prediction stage of the process with the aim of accurately predict the future behaviour of the system (Metzger, 2011).

2.3 Fuzzy Logic

Fuzzy logic is a method based on multi-valued logic which aims to formalize approximate reasoning (Zadeh, 1994). 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 (Zadeh, 1965). A linguistic variable is a type of variable that uses words instead of numbers to represent its values (e.g. slow, medium, fast) (Zadeh, 1994). The values used to define linguistic variables are called terms and the collection of terms is called the term set. A fuzzy rule is used to represent human knowledge using the form of IF-THEN within a fuzzy system (Li-Xin Wang, 1997).

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.

3 SYSTEM MODEL

An overview of the system model considered in this work is illustrated in Figure 1, which shows its core components: composition engine, adaptation manager, service binder, service selector, predictor and the sensors; and their interactions. This model was implemented with the aim of evaluating the proposed approach, enabling the execution of QoS aware service composition in an environment with proactive capabilities.

The composition engine is the software platform responsible for executing the composite services (processes' definitions) and hosting the components in charge of the adaptation process. Composite services are considered to consist of a series of abstract tasks that will be linked to executable services at runtime.

The adaptation manager works semi-independent of the rest of the components and is constantly monitoring and analyzing not only information collected by the sensors, but also the historical data. The use of historical data helps understanding the service behaviour and enables the detection of any possible deviation in the values of the QoS parameters. During the execution of a composite service, sensors collect fresh data, looking at activity and service levels, and send this information to the monitor. The monitor queries the historical database to obtain information about previous executions and states of the current service, then, sends this information to the analyzer, which evaluates both, fresh and historical data, in order to determine the need of adaptation. If adaptation is needed, the analyzer sends a request of adaptation to the planner, which obtains the adaptation values that will be sent

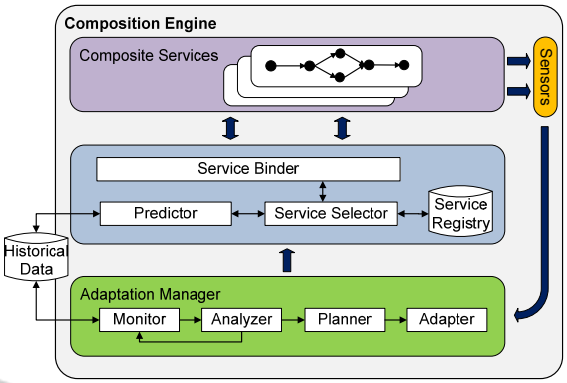

Figure 1: System model.

to the adapter. This information is forwarded to the service binder, in order to maintain/improve the QoS of the composition.

For each task in the composite service, the service binder invokes the service selector with the desired characteristics that the component service should provide. The service selector performs a search in the service registry based on the provided functional requirements. For each of the pre-selected services (candidates), the service selector invokes the predictor to obtain its estimated QoS. This information is sent to the service binder, which compares the candidates and selects the service that suits the request. If the need of a change was identified by the adaptation manager, the binder uses the adaptation values to perform the ranking and selection tasks.

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.

3.1 QoS Model

Services that offer the same functionality may be associated with several QoS attributes (Cardoso et al., 2004, Zeng et al., 2004), providing different QoS levels. By evaluating these attributes within a set of services that share the same goals, consumers can search/select components to be used in their applications.

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 (Hwang et al., 2007), nevertheless, when this information is not available, it can be obtained by analyzing data collected from past invocations (Cardoso et al., 2004).

In this work, the quality attributes that will be considered for each service are response time, cost, energy consumption and availability.

- Response time: time consumed between the invocation and completion of the service operation (Dai et al., 2009);
- Cost: fee charged to the consumer when invoking a service (Cardellini et al., 2012);
- Energy consumption: amount of power consumed by a server over a period of time (Buyya et al., 2010);
- Availability: probability that the service is up and ready for immediate consumption (W3C Working Group, 2003).

Considering response time and cost enables the selection of faster and cheaper services, providing a competitive advantage (Cardoso et al., 2004). Both parameters have been used in other approaches, like those presented in (Dai et al., 2009, Ying et al., 2009, Ardagna and Mirandola, 2010, Cardellini et al., 2012).

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 (Kaplan et al., 2009). Energy consumption has been selected as the third parameter because of the importance of energy efficiency when managing computing infrastructure and services.

The last parameter that has been selected is availability. By knowing the availability values of the different services, it is possible to select a subset of components that will provide a composition with higher probabilities to be fulfilled. Work that considers availability has been presented in (Huang et al., 2009, Canfora et al., 2008, Zeng et al., 2004).

To compute the values of these parameters at execution time, three situations have been considered within the composite service structure: single, sequential and concurrent service invocations. When computing the QoS parameters of a single service invocation, the QoS values of the activity that performs the invocation corresponds to the QoS values of the invoked service. For activities in a sequential structure, the values of response time, cost and energy consumption are summed for the different activities with service invocations, while availability is obtained by multiplying them.

$$Rt(P) = \sum_{i=1}^{n} Rt(s_i) \quad (1)$$

$$C(P) = \sum_{i=1}^{n} C(s_i) \quad (2)$$

$$Ec(P) = \sum_{i=1}^{n} Ec(s_i) \quad (3)$$

$$Av(P) = \prod_{i=1}^{n} Av(s_i) \quad (4)$$

For activities in a concurrent/parallel structure, the value of response time is considered as the maximum response time of the completed activities; values of cost and energy consumption are summed; and availability is the minimum availability value among the service invocations within the structure.

$$Rt(P) = max_{i=1,..,n}\, Rt(s_i) \quad (5)$$

$$C(P) = \sum_{i=1}^{n} C(s_i) \quad (6)$$

$$Ec(P) = \sum_{i=1}^{n} Ec(s_i) \quad (7)$$

$$Av(P) = min_{i=1,..,n}\, Av(s_i) \quad (8)$$

For equations (1) to (8), s_i corresponds to an activity with a service invocation within the composite service P.

3.2 Service Selection Model

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, their raw values are normalized before being processed and ranked. The following formula is used to normalize response time, cost and energy consumption, which are negative parameters (lower the value, higher the quality).

$$V_i = \frac{max_i - q_i}{max_i - min_i} \quad (9)$$

A different formula is used for availability, as it is a positive parameter (higher the value, higher the quality).

$$V_i = \frac{q_i - min_i}{max_i - min_i} \quad (10)$$

In both equations, 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 $V_i = 1$. After normalizing the corresponding values, results are computed using the Simple Additive Weighting formula:

$$W_i = rt_i(w_1) + c_i(w_2) + ec_i(w_3) + av_i(w_4) \quad (11)$$

Where rt_i is the service estimated response time; c_i is the service estimated cost; ec_i is the service estimated energy consumption; av_i is the service estimated availability; and w_1, w_2, w_3 and w_4 correspond to assigned weights where $w_1, w_2, w_3, w_4 \leq 1$ and $w_1 + w_2 + w_3 + w_4 = 1$.

3.3 Optimization Model

The proposed optimization model works as part of a proactive adaptation mechanism. It combines the analysis of historical and fresh data. QoS information of the different services and states of the composition 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). Based on this information, it is possible to take decisions about future actions.

The QoS parameters are obtained when the service invocation is performed. Response time is measured during the service's execution; the values of cost and energy consumption are retrieved from the service's WSDL file; while the value of availability is obtained based on historical data. According to the structures of the composite service, the QoS values of each task are computed using equations (1) to (8) and stored in the historical QoS data base, considering both individual values and accumulated. These values are used in order to obtain the global QoS of the composite service.

The service selection model previously described uses as weights for equation (11) the results of the optimization model evaluation. This model is based on two fuzzy support systems, which assess the QoS of the composition, determine the need of adaptation and, when adaptation is needed, obtain the weight values to be used during service selection. The optimization mechanism identifies when the QoS of the composition is decreasing. It also considers situations where a number of the accumulated QoS values of the previous activity in the process are better than expected, which provides the possibility of 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 used during service selection. Each of these systems uses its own linguistic variables and rules.

The first fuzzy support system evaluates the QoS of the composite service every N milliseconds, in order to identify as soon as possible the need of adaptation. It uses as inputs the measured QoS values collected from the composite service execution. The defined input variables are response time, cost, energy consumption and availability, 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. 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.

The second fuzzy support 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:

$$e\,(p_i) = \frac{x\,(p_i) - x_0\,(p_i)}{x_0(p_i)} \quad (12)$$

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 [-1, +1]. Benefit of adaptation is expressed with four terms, as defined in the first fuzzy system.

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, energy consumption weight and availability 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 1 describes the QoS evaluation method applied during the optimization process, which involves the use of the fuzzy systems previously described. Once the execution of a composite service starts, the adaptation manager constantly evaluates its QoS and obtains the errors between estimated and measured values (steps 1 to 12). The measured QoS values are used as inputs for the first fuzzy system. The benefit of adaptation is obtained (step 13) and evaluated (step 14); if it is medium or higher then there is a need of adaptation. When adaptation is needed, the system determines the adaptation weights. This action is performed by the second fuzzy system (steps 15 to 18). Weights are then adjusted, to fulfil the restriction $\alpha + \beta + \gamma + \phi = 1$ (step 19). Finally, the algorithm returns the weight values α, β, γ and ϕ (step 20). These values are sent to the service binder to be used at the moment of selecting the next service. When adaptation is not needed, the service binder ranks the services using fixed weight values.

Table 1: QoS evaluation algorithm.

Input:
rt → response time
cost→ cost
ec → energy consumption
av →availability
eRt → response time error
eCost→ cost error
eEc→ energy consumption error
eAv→ availability error

Output:
ω → benefit of adaptation
α → response time weight
β →cost weight
γ →energy consumption weight
ϕ →availability 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
(10) Sort by availability
(11) av ← **Obtain** measured availability
(12) eAv ← **Obtain** availability error
//fuzzy system 1
(13) ω ← **Obtain** benefit of adaptation
(14) **if** ω >= medium **then**
//fuzzy system 2
(15) α ← **Obtain** response time weight
(16) β ← **Obtain** cost weight
(17) γ ← **Obtain** energy consumption weight
(18) ϕ ← **Obtain** availability weight
(19) **Adjust** weights
(20) **return** α, β, γ and ϕ

4 EVALUATION

To evaluate the proposed optimization approach, an experimental environment was setup and two composite services were developed as test cases. Experiments were carried out to address the following question:

- Does the use of a proactive adaptation approach based on self-optimization helps improving the global QoS of a composition?

4.1 Experimental Environment

The experimental environment consists of 4 nodes configured on a WAN, distributed between United Kingdom and Germany, with estimated values for bandwidth and latency around 32Mbit/s and 29ms, respectively. Node 1 is a computer with Windows Vista, 4GB RAM and one Intel core2 duo 2.1GHz processor (located in United Kingdom). This node hosts the BPEL engine (Apache ODE 1.3.4), service registry (jUDDI 3.0.4) and historical data base (MySQL 5.1.51). It is in charge of coordinate the execution of the compositions and record all the gathered information. Nodes 2 to 4 are virtual machines setup on remote servers (located in Germany), each of the VM's uses Debian Squeeze x86 and 1GB RAM. These nodes host one application server (Tomcat 6.0.35.0) each, which contains 3 sets of Web services. In total there are 9 Web services deployed per composition's activity.

The initial values of the QoS parameters were established based on the node where the service is running and the corresponding set. Delays are inserted on some of the service sets, to obtain different response times, not only based on the network latency, but the Web services performance. This information is shown in Table 2.

Values of the cost and energy consumption change over time, or between services' executions. This adds dynamicity to the test environment and helps obtaining sensible results; also avoiding the invocation of only one service per each of the tasks in the composite services. To turn cost into a dynamic QoS value, the number of times a service

has been invoked within a period of time is evaluated continuously. Based on this information, it is possible to establish a new cost based on the demand (number of times the service is invoked), assuming that higher the demand, higher the cost. Cost value is updated on the WSDL file of each service.

Regarding energy consumption, each of the servers where the Web services are executed is assumed to have different hardware and software configurations. Servers information and their characteristics were selected from the Energy Star report (Energy Star, 2012). Using the model proposed in (Buyya et al., 2010), which is based on the percentage of CPU usage, it is possible to determine an approximate value to the server energy consumption.

$$P(u) = P_{max} \cdot k + (1 - k) \cdot P_{max} \cdot u \quad (13)$$

$$E = \int_t P(u(t)) \quad (14)$$

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 equations (13) and (14) and the data presented in Table 2.

Table 2: QoS parameters initial setup.

Server	Set	Time delays (ms)	Cost	Energy Consumption (W/sec)		Availability
				Idle	Load	
Node 2	S1	0	120	50.75	129.5	0.9
	S2	350	80			0.9
	S3	200	100			0.9
Node 3	S1	0	150	45.27	81.9	0.64
	S2	350	100			0.62
	S3	200	120			0.63
Node 4	S1	0	100	210.85	388.3	0.5
	S2	350	60			0.46
	S3	200	80			0.48

4.2 Experiment Description

Two test cases have been modelled in order to asses the proposed approach. These models are BPEL (OASIS, 2007) processes that represent typical examples for service composition scenarios. Test

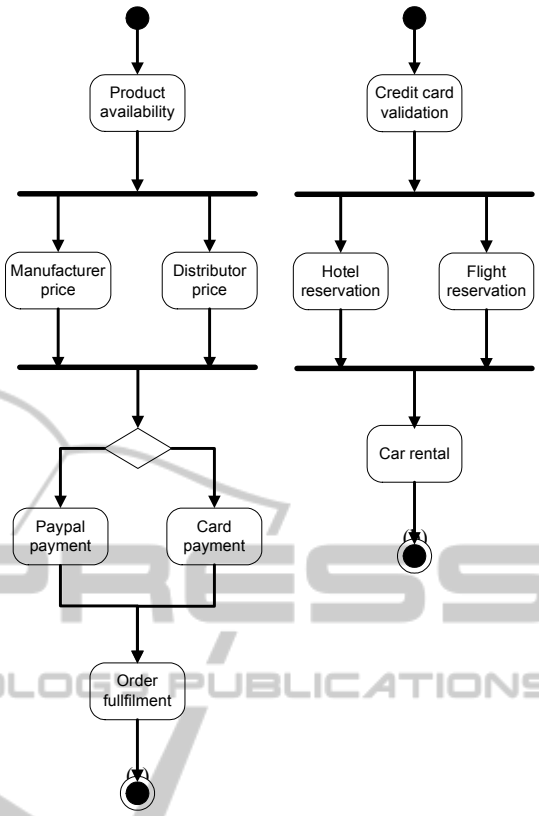

Figure 2: Test cases. (a) Order booking process. (b) Travel planning process.

case 1 is illustrated in Figure 2(a), it implements an order booking process that validates the product availability, obtains the best price of the product from two different providers, selects the best provider, performs the payment, and finally completes the order. Test case 2 implements a travel planning process, as shown in Figure 2(b). It validates a credit card, performs flight and hotel reservations in parallel, and finally invokes a car rental operation. For matter of simplicity, both diagrams only depict those activities that involve service invocations.

Per each of the tasks in the processes, there are 9 candidate services, distributed among the servers (nodes) that fulfil the required functionality and offer different QoS, giving a total of 45 candidate services to be used in test case 1 and 36 for test case 2. 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. Both processes are hosted and invoked from Node1.

In order to evaluate the proposed approach, both test cases where executed 100 times. These executions were performed from two different perspectives: 1) using the proactive optimization

mechanism; 2) using a non-adaptive method. The experiment was repeated 5 times to assess the consistency of the results based on statistical analysis.

4.3 Evaluation Results

The behaviour of the proactive optimization mechanism was compared with a non-adaptive approach, where service selection was performed using fixed weights set to 0.25. Initial results show improvements in the global QoS values of the composition when using the proposed approach. Global QoS refers to the final values of the different QoS properties (response time, cost, energy consumption and availability).

The plots shown in Figure 3 depict the behaviour of the order booking process, showing the mean values of the different QoS parameters after performing 5 sets of 100 runs. For the proposed approach, the values of cost and energy consumption change over time, as previously described, while for the non-adaptive approach, remain constant. For both cases, the value of availability changes according to the behaviour of the component services.

After analyzing the value of each of the QoS parameters, in both processes, it was identified that, in order to improve response time, energy consumption and availability, there was an increment in the composition's cost.

In test case 1, results show that the proposed approach provides a mean reduction of 2% with a standard deviation of 6.7% in the measured response time values. Also, it can be noticed from Figure 3(a), that it presents a more stable behaviour, without showing high peaks, as compared to the non-adaptive approach. This is due to the constant evaluation of the QoS parameters during execution.

In terms of energy consumption, it is important to notice that this value is not only based on power consumption, but also influenced by time. As a result, a small response time may produce a small energy consumption value. Figure 3(b) shows the values corresponding to energy consumption, which have a similar behaviour to response time, and provide a mean reduction of 14.7% with a standard deviation of 18.9%. Results also indicate that there is a significant improvement in the processes' availability, presenting a mean increase of 41% with a standard deviation of 35%. The availability values corresponding to the order booking process are illustrated in Figure 3(c). Regarding cost, it can be

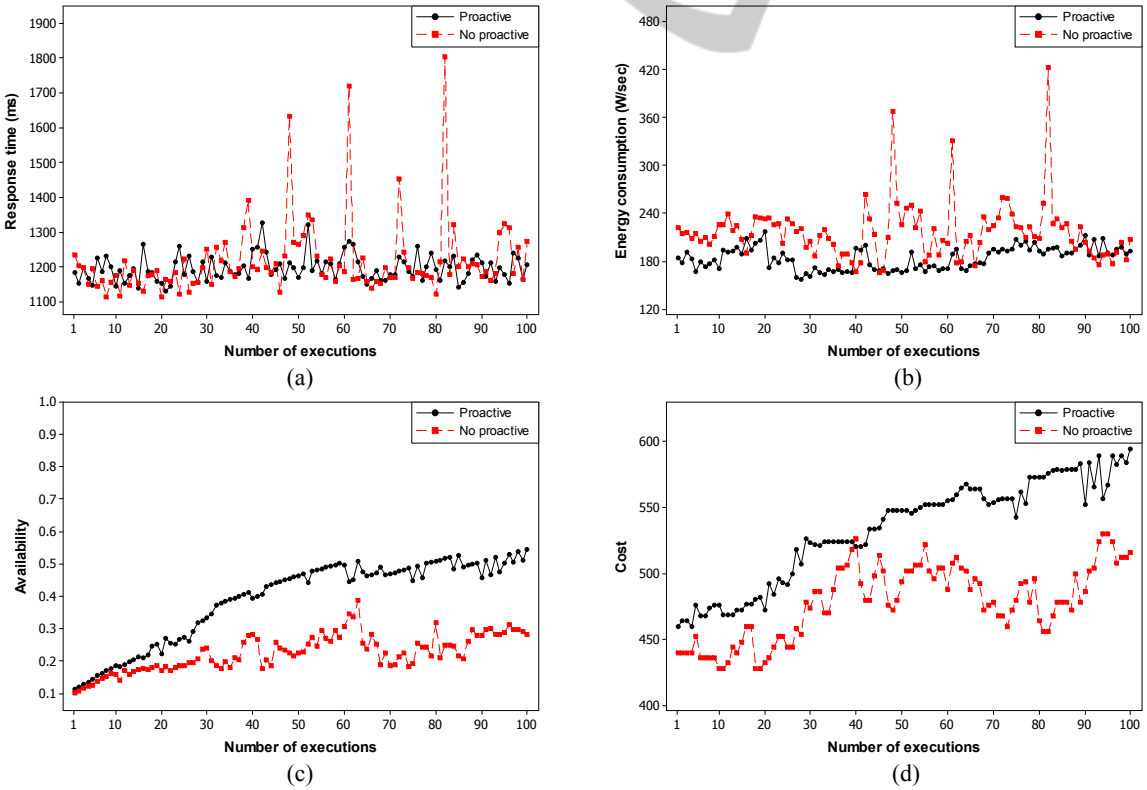

Figure 3: Order booking process results. (a) Response time. (b) Energy consumption. (c) Availability. (d) Cost.

noticed from Figure 3(d) that the use of the proposed approach turns into more expensive composite services. It shows a mean increase of 11% with a standard deviation of 8.4%.

Results obtained from test case 2 show a similar behaviour; where response time, energy consumption and availability values are improved, while cost increases. In terms of response time, it shows a mean reduction of 8.9% with a standard deviation of 16%. For energy consumption, the obtained mean reduction is 4.6% with a standard deviation of 29%. Regarding availability, it provides an improvement of 18% with a standard deviation of 25%. Finally, in terms of cost, there is an increment of 12.5% with standard deviation of 6.8%.

Based on the analysis of the weight values obtained by the optimization model and sent to the service binder, the parameter that had the higher impact within the adaptation process was energy consumption, followed by response time. Because of this, at the moment of selecting new services to be invoked, priority would be given to those that are being executed on servers with lower energy consumption, and lower response time. Which, based on the QoS configuration, are the services that also involve higher costs. Different QoS configurations may give different results; however, because of the use of multiple QoS criteria, it is likely to find that not all the parameters can be improved.

5 RELATED WORK

Several works have been proposed to mitigate the impacts of unexpected events during the execution of services, ensuring/maintaining the functional and quality levels. These approaches can be classified based on the moment when adaptation takes place into the categories: reactive and proactive. Reactive adaptation occurs after the appearance of an undesired event, while proactive adaptation aims to predict and prevent the occurrence of the problem (Aschoff and Zisman, 2012).

Some of the approaches that support reactive adaptation implement self-* properties. Self-healing mechanisms aim to prevent composite services from failing, from functional and non-functional perspectives. Projects like those presented in (Canfora et al., 2008, Erradi and Maheshwari, 2008, Dai et al., 2009, Wu et al., 2009, Ying et al., 2009, Ardagna et al., 2011, Wenjuan et al., 2010, Baker et al., 2013) apply self-healing approaches, where new services are selected and invoked after a functional failure or a QoS constraint violation. Self-optimization mechanisms 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 (Ardagna et al., 2011, Calinescu et al., 2011, Cardellini et al., 2012).

Approaches that support proactive adaptation are presented in (Hielscher et al., 2008, Tosi et al., 2009, Leitner et al., 2010, Aschoff and Zisman, 2012, Metzger, 2011, Yuelong et al., 2012, Sammodi et al., 2011). The work presented in (Aschoff and Zisman, 2012) introduces a proactive adaptation approach that enables service replacement (1-1, 1-n, n-1, n-m) when it detects situations that may cause the composition to stop its execution (unavailable or malfunctioning services); or that allow the composition to continue its execution, but not in its best way. Also it considers the emergence of better services and new requirements. The approach uses a composition template as start point and selects a set of candidate services to be used in the composition and their replacements. The approach introduced in (Yuelong et al., 2012) combines runtime information with design-time specifications (of each component service within a composition), in order to construct a k-step model of the current service states. The resulted model can be used to be compared with the desired behaviour of the composition. The work in (Leitner et al., 2010) aims to minimize Service Level Agreement (SLA) violations. It uses predictions of SLA violations generated with regressions of monitored and estimated data. These predictions are evaluated at defined checkpoints. In (Hielscher et al., 2008), a framework that uses online testing to trigger proactive adaptation in service-based applications is described. Test objects can be single or composite services. While performing online testing, if an online test fails or deviates from its expected behaviour, the framework will trigger adaptation to avoid undesirable consequences. One of the application scenarios for this approach is composite services. The work presented in (Tosi et al., 2009) proposes a self-adaptive mechanism based on the use of test cases to obtain possible mismatches between requested and provided services. When the diagnosis mechanism reveals mismatches, it triggers adaptation strategies that update the structure and the behaviour of the client application to solve the identified problems. Even though this approach is not mainly focus in service composition, it presents a proactive mechanism that works in service-based applications. In (Sammodi et al., 2011) it is presented a proactive approach for adaptation in service-based applications that

combines monitoring, online testing and quality prediction. When a service is likely to be used with a high frequency, it is selected to be tested. The use of pre-defined test cases (concrete data inputs) enables the system to collect information about the behaviour of the services and complement the data gathered during monitoring. Finally, the work described in (Metzger, 2011) discusses two main directions than can be followed in order to perform proactive adaptation in service oriented systems. The first direction is to improve the failure predictions techniques. Some prediction techniques identified by the authors include data mining, online testing, runtime verification, statics analysis and simulation. The second direction is by dynamically estimating the accuracy of the predicted failures during the operation of the service-oriented system.

Although these approaches are closely related with the work described in this paper, there are significant differences:

- The QoS parameters considered in this study are response time, cost, energy consumption and availability, while related approaches mainly focus on response time.
- Adaptation is not limited to failure prevention, but also considers the possibility of improvement in the QoS levels.
- The use of the benefit of adaptation, obtained from the measured QoS values, to determine whether adaptation is needed or not.
- The use of fuzzy logic as a decision making tool to determine the need of adaptation in the context of proactive adaptation in service composition.

6 CONCLUSION AND FUTURE WORK

This paper presents a proactive adaptation approach for service composition that implements a self-optimization mechanism, which aims to improve/maintain the quality levels of the composite services. At runtime, this mechanism performs a continuous evaluation of the composite services' behaviour and triggers adaptation when the benefit of performing this action is considered to be significant. In order to perform this decision making process, it uses fuzzy support systems and carries out an analysis on historical and fresh QoS data.

Evaluation results show that the proposed mechanism improves the global QoS values of the compositions, showing significant improvements regarding response time, energy consumption and availability. However, it was not possible to improve all the considered quality parameters, as there was an increment in the cost of the composite services. This behaviour is shown due to the relationship between the values of quality parameters exhibit by the services, as those services with lower energy consumption and higher availability, also display higher costs.

Future work includes the use of Service Level Agreements on top of the composite services, in order to provide a contract between the provider and consumer. Also, it is considered the extension of the adaptation mechanism, by adding features that enable service replacements considering different structures with the forms 1-n, n-1 and n-m. Finally, with the aim of performing a comparison, it is considered the implementation and evaluation of mechanisms proposed in the related work, using the same composition environment.

REFERENCES

Ardagna, D., L. Baresi, et al., 2011. A Service-Based Framework for Flexible Business Processes. *IEEE Software* 28(2): 61-67.

Ardagna, D. and R. Mirandola, 2010. Per-Flow Optimal Service Selection for Web Services Based Processes. *Journal of Systems and Software* 83(8): 1512-1523.

Aschoff, R. and A. Zisman, 2012. Proactive Adaptation of Service Composition. *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'12)*. Zürich, Switzerland: 1-10.

Baker, T., O. F. Rana, et al., 2013. Towards Autonomic Cloud Services Engineering via Intention Workflow Model. *Proceedings of the 10th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON'13)*, Zaragoza, Spain.

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

Calinescu, R., L. Grunske, et al., 2011. Dynamic QoS Management and Optimization in Service-Based Systems. *IEEE Transactions on Software Engineering* 37(3): 387-409.

Canfora, G., M. D. Penta, et al., 2008. A Framework for QoS-Aware Binding and Re-Binding of Composite Web Services. *Journal of Systems and Software* 81(10): 1754-1769.

Cardellini, V., E. Casalicchio, et al., 2012. MOSES: A Framework for QoS Driven Runtime Adaptation of

Service-Oriented Systems. *IEEE Transactions on Software Engineering* 38(5): 1138-1159.

Cardoso, J., A. Sheth, et al., 2004. Quality of Service for Workflows and Web Service Processes. *Journal of Web Semantics* 1(3): 281-308.

Châtel, P., J. Malenfant, et al., 2010. QoS-based Late-Binding of Service Invocations in Adaptive Business Processes. *Proceedings of the International Conference on Web Services (ICWS'10)*, Miami, USA, IEEE Computer Society.

Cheng, B., R. Lemos, et al., 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. *Software Engineering for Self-Adaptive Systems* 5525: 1-26

Dai, Y., L. Yang, et al., 2009. QoS-Driven Self-Healing Web Service Composition Based on Performance Prediction. *Journal of Computer Science and Technology* 24(2): 250-261.

Dustdar, S., C. Dorn, et al., 2010. A Roadmap Towards Sustainable Self-Aware Service Systems. *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, Cape Town, South Africa, ACM.

Dustdar, S. and W. Schreiner, 2005. A Survey on Web Services Composition. *International Journal on Web and Grid Services* 1(1): 1–30.

Energy Star, 2012. Computer Servers Product List - Families.

Erradi, A. and P. Maheshwari, 2008. Dynamic Binding Framework for Adaptive Web Services. *Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*, Athens, Greece, IEEE Computer Society.

Hielscher, J., R. Kazhamiakin, et al., 2008. A Framework for Proactive Self-adaptation of Service-Based Applications Based on Online Testing. *Proceedings of the 1st European Conference Service Wave*, Madrid, Spain, Springer-Verlag.

Huang, A., C.-W. Lan, et al., 2009. An Optimal QoS-based Web Service Selection Scheme. *Information Sciences* 179(19): 3309-3322.

Hwang, S.-Y., H. Wang, et al., 2007. A Probabilistic Approach to Modeling and Estimating the QoS of Web-Services-Based Workflows. *International Journal of Information Sciences* 177(23): 5484-5503.

Kaplan, J., W. Forrest, et al., 2009. Revolutionizing Data Center Energy Efficiency, McKinsey.

Leitner, P., A. Michlmayr, et al., 2010. Monitoring, Prediction and Prevention of SLA Violations in Composite Services. *Proceedings of the IEEE International Conference on Web Services (ICWS'10)*, Miami, USA.

Li-Xin Wang, 1997. *A Course in Fuzzy Systems and Control*, Prentice Hall.

Metzger, A., 2011. Towards Accurate Failure Prediction for the Proactive Adaptation of Service-Oriented Systems. *Proceedings of the 8th Workshop on Assurances for Self-adaptive Systems*, Szeged, Hungary, ACM.

OASIS, 2007. Web Services Business Process Execution Language Version 2.0. Retrieved Dec. 2013, from http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

Sammodi, O., A. Metzger, et al., 2011. Usage-Based Online Testing for Proactive Adaptation of Service-Based Applications. *Proceedings of the IEEE 35th Annual Computer Software and Applications Conference (COMPSAC'11)*, Munich, Germany.

Tosi, D., G. Denaro, et al., 2009. Towards Autonomic Service-Oriented Applications. *International Journal of Autonomic Computing* 1(1): 58-80.

W3C Working Group, 2003. QoS for Web Services: Requirements and Possible Approaches. Retrieved Dec. 2013, from http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

Wenjuan, L., Z. Qingtian, et al., 2010. A Framework to Improve Adaptability in Web Service Composition. *Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET'10)*, Chengdu, China.

Wu, G., J. Wei, et al., 2009. Towards Self-Healing Web Services Composition. *Proceedings of the First Asia-Pacific Symposium on Internetware*, Beijing, China, ACM.

Ying, Y., Z. Bin, et al., 2009. A Self-Healing Composite Web Service Model. *Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC'09)*, Biopolis, Singapore.

Yuelong, Z., W. Xiaobin, et al., 2012. Predicting Failures in Dynamic Composite Services with Proactive Monitoring Technique. *Proceedings of the IEEE Eighth World Congress on Services (SERVICES'12)*, Honolulu, USA.

Zadeh, L. A., 1965. Fuzzy Sets. *Information and Control* 8(3): 338-353.

Zadeh, L. A., 1994. The Role of Fuzzy Logic in Modeling, Identification and Control. *Modeling, Identification and Control* 15(3): 191-203.

Zeginis, C. and D. Plexousakis, 2010. Web Service Adaptation: State of the Art and Research Challenges. Heraklion, Crete, Greece, Institute of Computer Science.

Zeng, L., B. Benatallah, et al., 2004. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering* 30(5): 311-327.