A QoS Optimization Model 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— **The dynamic nature of the Web service execution environment generates frequent variations in the Quality of Service offered to the consumers, therefore, obtaining the expected results while running a composite service is not guaranteed. Adaptation approaches aim to maintain functional and quality levels, by dynamically adapting composite services to the environment conditions reducing human intervention. This paper presents an adaptation approach based on self-optimization. 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 and the establishment of priorities between the service quality attributes. Experimental results show significant improvements in the global QoS of the use case scenario, providing reductions up to 16% in the global cost and 14% in response time.**

Keywords - Web service composition; adaptation; 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 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.

Development in the field of service composition has resulted in a set of dataflow models (orchestration and choreography), approaches (static, dynamic, manual and automatic) and techniques (model-driven, declarative, workflow-based, ontology-driven and AI-Planning) that enable composition from different perspectives. However, some challenges still remain open, which are closely related to automatic-dynamic service composition and include the implementation of mechanisms that enable: Quality of Service awareness, adaptive capabilities, risk awareness, conformance, security and interoperability.

The approach proposed in this paper is mainly focused on adaptive mechanisms for service composition. 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 [3].

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 (QoS) levels. By implementing adaptive mechanisms, composite services should be able to morph and function in spite of external and internal changes, searching to maximize the composition potential and reducing as much as possible human involvement.

This work presents a self-optimization solution for service composition. 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. Upon invocation, a set of tasks are executed as defined in the service workflow. QoS data evaluation from previous tasks enables the model to determine priorities between the QoS attributes, and these priorities are applied during service selection. The approach has been implemented in a framework and was evaluated empirically by analyzing the execution through a use case. The major contribution of this paper is:

- The optimization model for service composition that analyzes global QoS from previous tasks in order to determine priorities for service selection.

This paper is structured as follows: background and related work are described in Section II. Section III presents the proposed framework, service selection and optimization models. Section IV presents the experimental description and results. Conclusions and future work are given in Section V.

II. BACKGROUND AND RELATED WORK

In service composition, it is necessary to have a set of available services that offer certain functionality and also fulfil Quality of Service constraints [4].

QoS properties refer to non-functional aspects of Web services, such as performance, reliability, scalability, availability and security [5]. 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 [6], nevertheless, when this information is not available, it can be obtained by analyzing data collected from past invocations [7].

Different approaches have been presented to evaluate QoS attributes in service composition, aiming to select a set of components that optimize the global QoS. Some of these approaches are based on the works described in [7] and [8], 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.

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.

Based on the objectives of the composition and the causes and impact of adaptation, different self-adaptive properties can be selected and implemented. The most used properties in service composition approaches are self-healing [10], self-configuration [11] and self-optimization [12]. Each of these properties can be related to different attributes, like availability, survivability, maintainability, reliability, efficiency and functionality [13].

Self-healing mechanisms aim to prevent composite services from failing, from functional and non-functional perspectives. Projects such as those are presented in [14-21], apply self-healing approaches, where new services are selected and invoked after a functional failure or a QoS constraint violation.

In self-configuring approaches, like those presented in [9] and [22], service selection is performed by searching for an optimal configuration of components based upon the initial constraints.

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 [16], [21] and [22].

Although these approaches are closely related with the work described in this paper, there are meaningful differences. Firstly, the proposed optimization approach takes into consideration the QoS values measured from previous tasks at the time of selecting a new service. 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.

III. 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 desired characteristics that the component service should provide.

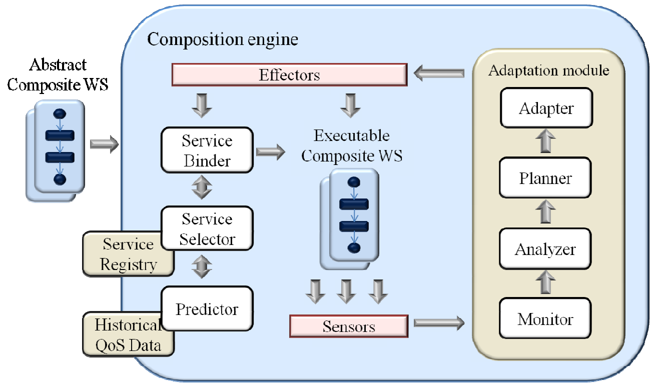

Figure 1. System model.

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 [7-8], which could be used as a differentiating point in the preference of consumers. In this work, the following quality attributes, which have been used in other approaches ([4],[14-16]), will be considered for each service:

- Response time: the time consumed between the invocation and completion of the service operation [14];
- Cost: fee charged to the consumer when invoking a service [16].

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 response time and cost. Due to these attributes having different units of measure, the raw values are first normalized with natural logarithms. Results are then computed using the Simple Additive Weighting formula:

$$W_i = t_i\,(w_1) + c_i\,(w_2) \qquad (1)$$

where:
t_i corresponds to the service estimated response time,
c_i corresponds to the service estimated cost,
w_1 and w_2 correspond to weights where $w_1 + w_2 = 1$ and $w_1, w_2 \leq 1$.

```
Input:
estRT → estimated accumulated response time
estC → estimated accumulated cost
rt → real response time
rc → real cost
w1, w2 → weights
ω → maximum difference between estRT and rt
ϕ → maximum difference between estC and rc

Output:
α → response time weight
β → cost weight

(1)  ψ ← calculate response time difference (estRT - rt)
(2)  δ ← calculate cost difference (estC - rc)
(3)  α ← β ← 0.5
(4)  Sort by response time
(5)  if ψ ≥ ω || -δ ≥ ϕ then
(6)      α ← w1
(7)      β ← w2
(8)  else
(9)      Sort by cost
(10)     if δ ≥ ϕ || -ψ ≥ ω then
(11)         α ← w2
(12)         β ← w1
(13) return α and β
```

Figure 2. QoS evaluation algorithm.

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 behaviour it is possible to take decisions about future actions [23]. As part of this work, at runtime 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 is 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 optimization approach is based on the service selection model previously described. It uses variable weights and performs a service reselection on the obtained set of candidates. When the accumulated response time (or cost) of the previous activity in the process is less than expected, it provides some slack that can be used while selecting the next service in the process.

Before invoking a Web service operation, the measured accumulated QoS values of the previous task are evaluated and compared to the corresponding estimated values. The algorithm presented in Fig. 2 describes the QoS evaluation method applied during optimization. After obtaining the differences between the estimated and real QoS values (steps 1 and 2), these values are compared to the maximum desired percentage of difference between real and estimated values, represented by ω and ϕ. The first comparison is performed based on response time (step 5), if there is no adaptation required, then evaluation is carried out based on cost (step 10). The algorithm returns α and β (step 13), which are the new weights to apply in the service selection process. These weights are established as float values that give priority to a certain attribute.

IV. 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 III were deployed and configured within the experimental environment. Experiments were carried out to address the following question:

- 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 is illustrated in Fig. 3. It consists of 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), and considers response time of Web services running over a Wide Area Network (WAN) when executing the local services. 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.

B. *Experiment Description*

The test case is a BPEL [24] 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.

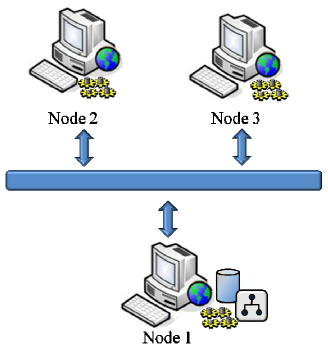

Figure 3. Experimental environment.

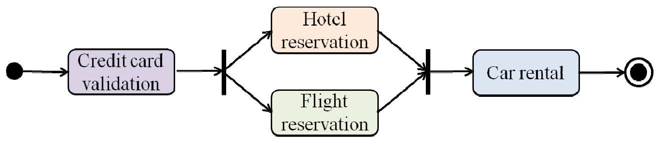

Figure 4. Travel planning process.

The travel planning service is illustrated in Fig. 4. Per each of the tasks in the process, there are 9 candidate services 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.

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.

The travel planning service was executed 50 times to analyze the behaviour of the optimization approach and evaluate its overall benefit. The maximum difference between estimated/real response time and cost was established as 10%. The service was also executed performing a simple service selection without QoS analysis.

As weights are those that provide priorities to the QoS attributes at the time of performing a service selection, values for w_1 and w_2 (algorithm in Fig. 2) were set as 0.3 and 0.7, respectively.

C. *Evaluation Results*

Initial results show that the proposed approach provides a meaningful improvement on the global QoS over a simple service selection approach. Global QoS refers to the final values of the different QoS properties (response time and cost) of the composite service. Fig. 5 and Fig. 6 present a comparison between both approaches based on response time and cost, respectively.

The first plot shows that the measured response time of the composite service executed using the optimization approach is closer to the corresponding estimated values, as compared to the behaviour of the simple selection approach, where most of the values are above the estimations. Measured average response time values correspond to 7049 ms and 7416 ms, where the proposed approach provides a mean reduction of 5%, a highest reduction of 14% and standard deviation of 7.45%.

Contrary to the behaviour of response time, cost estimations for the proposed approach are not close to the real measurements. As illustrated in Fig. 6, most values are above estimations; nevertheless, there can be found some significant cost reductions, the highest being of 16%. Average cost value was 452, with a standard deviation of 6.8%.

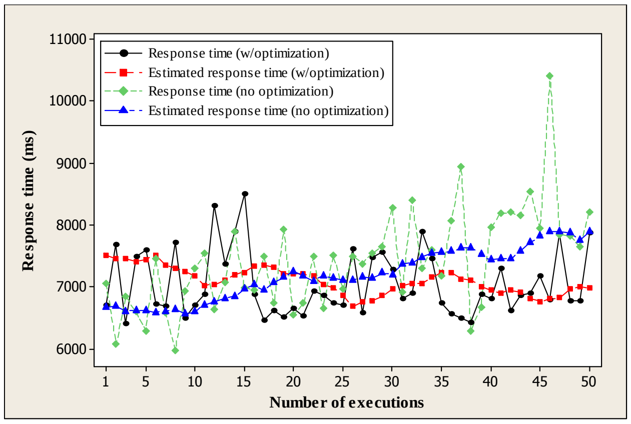

Figure 5. Composite service response time comparison between optimization and simple selection approaches.

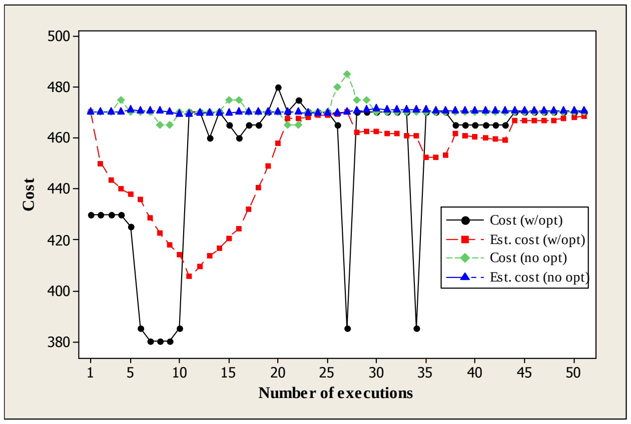

Figure 6. Composite service cost comparison between optimization and simple selection approaches.

To summarize the behaviour of both approaches, Fig. 7 presents a plot where response time and cost values were normalized and related using the Simple Additive Weighting formula presented in Section III. For both QoS attributes weights were established at 0.5.

From a global perspective, results demonstrate that using the proposed approach provides better QoS values, in most of the service executions.

It was noticed during the evaluation stage, that the overhead caused by the use of a service registry and predictive algorithms oscillate between 1500 and 2000 ms, which represent an important delay at runtime.

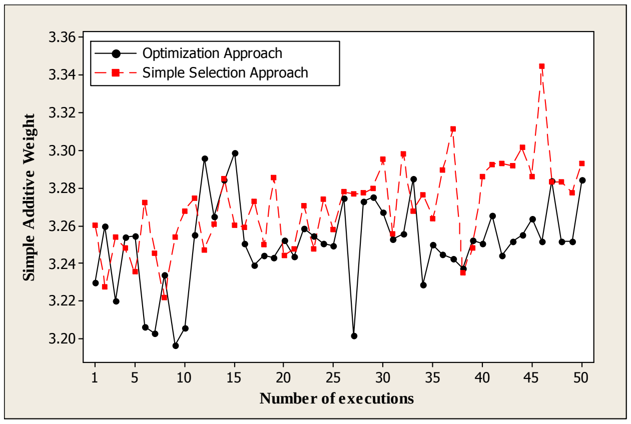

Figure 7. Composite service Simple Additive Weight comparison between optimization and simple selection approaches.

V. Conclusion and Future Work

The execution of a composite service can be compromised by changes in the behaviour of its components. Mechanism such as adaptation, focus upon reducing the impact of these changes.

Adaptation in service composition aims to maintain/improve functional and quality levels while executing composite services. Thus, the development of adaptation mechanisms for service composition is an important task.

This work presents an adaptation approach for service composition that implements a self-optimization mechanism. During composite service execution, QoS attributes are monitored and optimization is triggered if there is a difference between estimated and real values.

In summary, evaluation indicates that by using the proposed approach, there can be achieved significant improvements in the global QoS of the composite services.

This paper is part of an ongoing research. Future work includes the extension of the quality criteria, considering other key QoS attributes like availability and reliability. Also, it is planned to investigate different self-adaptive properties and extend the actual framework, in order to increase the coverage of events that can occur at runtime.

References

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

[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] 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.

[4] 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.

[5] W3C Working Group. "QoS for Web Services: Requirements and Possible Approaches". Available: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/ [May, 2012].

[6] 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.

[7] 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.

[8] L. Zeng, et al., "QoS-Aware Middleware for Web Services Composition," IEEE Trans. Softw. Eng., vol. 30, pp. 311-327, 2004.

[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, 2010, pp. 227-234.

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

[11] 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.

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

[13] 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.

[14] 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.

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

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

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

[18] 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 '07), Vienna, Austria, 2007, pp. 449-455.

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

[20] 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, 2008, pp. 162-167.

[21] 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.

[22] R. Calinescu, et al., "Dynamic QoS Management and Optimization in Service-Based Systems," IEEE Trans. Softw. Eng., vol. 37, pp. 387-409, 2011.

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

[24] OASIS. "Web Services Business Process Execution Language Version 2.0". Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html [May, 2012].