Service Level Agreement Monitor (SALMon)

David Ameller, Xavier Franch
Universitat Politècnica de Catalunya, Spain
{dameller, franch}@lsi.upc.edu

Abstract

One of the most successful architectural styles nowadays is Service Oriented Architecture (SOA). In this type of architecture there are a lot of dependencies between services, but each service is an independent element of the system. In this situation we need some way to ensure that every service is working correctly and to take actions when something goes wrong to evolve the architecture as fast as we can. For example, if one of the lower level services of the service composition stops working, it could lead to a total or partial system malfunction. In this situation there is a need to be able to build reliable SOA systems.

Our proposal, SALMon, is based on monitoring the services for Service Level Agreement (SLA) violations. The SALMon architecture is composed of three types of components: Monitors that are composed of measure instruments, the measured quality attributes being taken from an ISO/IEC 9126-1-based service oriented quality model; Analyzers that check the SLA rules; and Decision Makers that perform corrective actions to satisfy SLA rules again. These 3 types of components are mostly technology-independent and they act as services inside of a SOA system making our architecture very scalable and comfortable for its purpose.

1. Introduction

Service-Oriented Architecture (SOA) [1] is an emerging software architecture; systems based on this architecture consist in multiple services working together. SOA systems must fulfill some Quality of Service (QoS) [2] requirements and as a result, each service QoS is specified in a contract which is known as Service Level Agreement (SLA) [3]. Services can change their QoS in runtime due to environmental issues or to changes made by the provider of the service. In this situation, SOA systems need to be adaptable in runtime, and in fact new service technologies like Web services are already prepared to substitute one Web service by another at runtime, using standard protocols like UDDI [4] and WSDL [5] and because it is common to have Web services with the same interface and the same functionality. We can do the same for other kinds of services like databases but in a non-standardized and more complex way.

In this paper we will show a concrete tool called SALMon, which uses the flexibility provided by SOA to make SOA systems capable to adapt themselves in order to maintain the requirements stated in SLA specifications. SALMon uses a monitoring technique to provide runtime QoS information that is needed to detect SLA violations.

The rest of this paper is divided into two main sections: first we provide a framework for metrics definition based on previous works and the second part is dedicated to the details of SALMon architecture. Finally there is a section for the conclusions.

2. QoS and monitorable quality attributes

The first two questions that we faced were: “What do we want to monitor?” and “What can we monitor?”. To answer the first question we have built a quality model [6] for software services based in previous work done in our group [7], and quality-related standards especially in the domain of web services. This model was part of our participation in a ITEA European project, SODA (Services Oriented Devices & Delivery Architectures, www.soda-itea.org), in which we participated with the responsibility of identifying and classifying the characteristics needed for defining the quality of Web services. The model (Figure 1) is based on the ISO/IEC 9126 standard [8]. However, since this standard focuses just on the technical aspects of software, we have used some previous work to enlarge this model including non-technical aspects [9].

We have opted by an ISO/IEC 9126-based standard due to: 1) its generic nature: the standard fixes some high-level quality concepts, and therefore quality models can be tailored to specific domains; 2) it allows creating hierarchies of quality features, which are essential for building structured quality models; 3) the standard is widespread.

0-7695-3091-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCBSS.2008.13

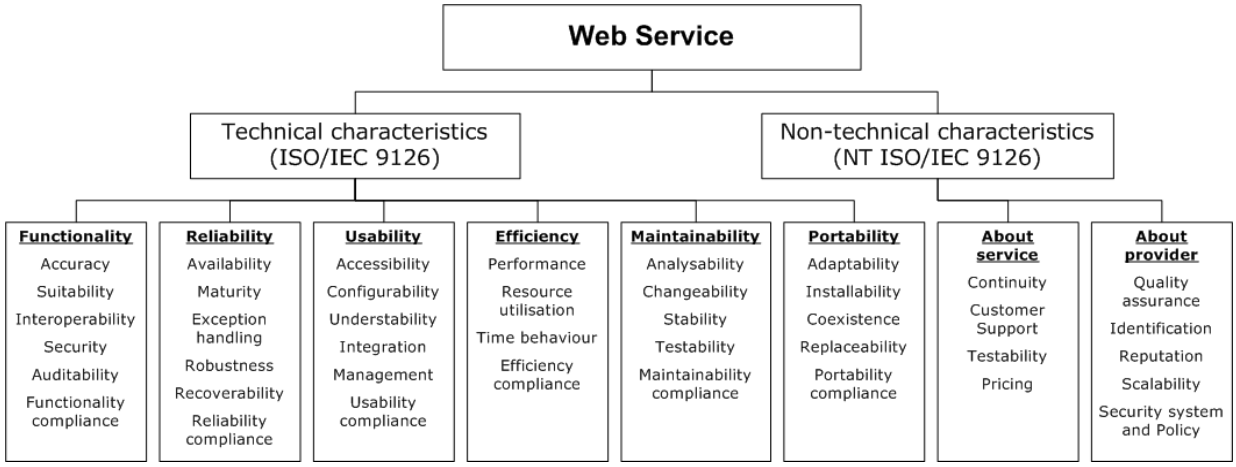

Figure 1: Quality model for services

ISO/IEC 9126-1 specifically addresses quality model definition and its use as a framework for software evaluation. A 9126-1-based quality model is defined by means of general software characteristics, which are further refined into subcharacteristics, which in turn are decomposed into attributes, yielding to a multilevel hierarchy. At the bottom of the hierarchy appear measurable software attributes, whose values are computed using some metric. Throughout this paper, we refer to characteristics, subcharacteristics, and attributes as quality entities.

In the proposed quality model, as an example, one characteristic is Efficiency and one of its subcharacteristics is the Time Behaviour, but time behaviour itself is not a single measurable concept, therefore we need to define attributes to decompose this subcharacteristic. The attributes are normally dependent on what we want to measure. In our case, since we are focusing on Web services, Response Time and Execution Time are good examples of measurable attributes for Time Behaviour.

At this point we have a lot of attributes that can be measured in some way, but we are interested only in those that can be measured using a monitoring technique. Maintainability, portability, usability and reliability are groups of characteristics that cannot be monitored, basically because they are software design characteristics, they are not supposed to change during execution time. Therefore, we concluded that just a small set of attributes are monitorable, namely those related to the subcharacteristics Availability, Time Behaviour and Accuracy. We remark that Accuracy may be difficult to measure because it needs a lot of information of the concrete Web service; to monitor the accuracy we need to know the concrete functionality of the service and have available concrete predefined tests to run on it.

Next it is necessary to define metrics for these three monitorable attributes. The Table 1 is an example of metrics that could be used for the Response Time attribute belonging to the Time Behaviour Characteristic.

	Metric	Description
Response time	**Current response time**	It measures the current response time in milliseconds to access to a Web Service.
	Minimum response time	It measures which is the lowest response time in milliseconds to access to a Web Service.
	Maximum response time	It measures which is the maximum response time in milliseconds to access to a Web Service.
	Average response time	It measures which is the average response time in milliseconds to access to a Web Service.

Table 1: Response time metrics

3. SALMon Architecture

The architecture of our tool is a SOA; this decision makes SALMon very easy to install on a running SOA system. SOA is a component-based architecture; this means that we can change some of the components by others that have the interfaces defined for the SALMon architecture.

In the Figure 2 the proposed architecture is shown. We may observe that it is composed of three types of services: Monitor, Decision Maker and Analyzer.

The Monitor service is composed of Measure Instruments; these components will bring the measures to the Monitor that has the responsibility to maintain this information updated. The update process is an iterative call to each Measure Instrument in different intervals of time, saving the results in a database. The intervals of time are part of the information provided with each metric.

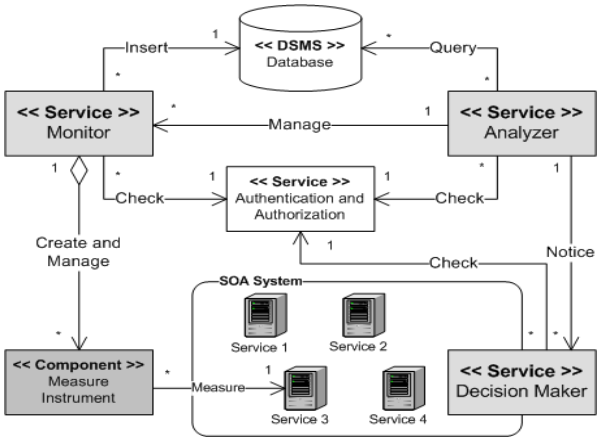

Figure 2: SALMon architecture

Measure Instruments are components instantiated in each monitored service to get all the *basic metrics* of the selected quality attributes (basic metrics are the ones that allow to calculate the rest of derived metrics, for example Current Response Time is the basic metric for Response Time attribute; others metrics such as the Minimum, Maximum and Average Response Time may be computed from it). While the interface for the Measure Instruments is independent of the technology, their implementation is technology-dependant because they are built to support one kind of services (e.g., Web services, HTTP services, DBMS services). They can be seen as plugins to support specific service technology.

Measure Instruments have the responsibility to minimize the number of interactions performed with the monitored service.

The Decision Maker service selects the best treatment to solve the SLA violations detected by the Analyzer in a concrete SOA system. Each Decision Maker is related with only one SOA System and it is preferred to place the service inside the concrete SOA system where it is taking decisions for security reasons.

The Decision Maker service could use a repository of treatments and alternative services for a concrete SOA system and it will automatically select and execute the best treatment for the reported SLA violations.

The Analyzer manages Monitors and checks for SLA violations in concrete SOA systems. When a violation is detected it is notified to the Decision Maker of the affected SOA system. In general an Analyzer can handle multiple SOA systems using one Monitor and one Decision Maker for each one. Anyway the use of Decision Maker services is optional but in this way the SALMon user is limited to monitoring.

The SLA can be configured manually with the interface provided by the Analyzer or automatically with a SLA standard document for each service (e.g., WSLA [10] for the case of Web services). We understand SLA as a set of conditions that must be true in some time interval. A condition is composed of the evaluated metric, a relational operator and a value for the comparison (i.e. "current response time < 100ms" is a condition that must be true for the specified service during the specified time interval).

The SALMon architecture includes the use of two services that are common to the majority of SOA systems therefore they can be shared. The first one is a database used by the Monitors to store the measures which are processed by the Analyzer in order to detect SLA violations. This service is mandatory. The second external service is for authentication and authorization of SALMon users; these users could be normal users or administrators. Normal users will be able to set SOA systems and SLA while administrators will set the configuration of the SALMon system. This service is optional.

The first implementation of SALMon is part of a joint work between our Software Engineering for Information Systems Group (GESSI) at the Universitat Politècnica de Catalunya (UPC), Spain, and the Institute for Systems Engineering and Automation (SEA) at the Johannes Kepler University (JKU) in Linz (Austria). Some details of this collaboration may be found at [11]. In this context, SALMon architecture will be only focused on Web services (see Figure 3). Measure Instruments will be prepared to measure Response Time and Availability (Accuracy is omitted in this first implementation due to its complexity as commented above). This combination will make this implementation easy to install on a running Web service-based SOA system.

For the database we have selected a stream database for two reasons, first because the type of information to be stored fits with the one expected in this kind of database to perform queries efficiently, and second because the kind of queries that the Analyzer needs are easy to express using the extended SQL provided by this technology.

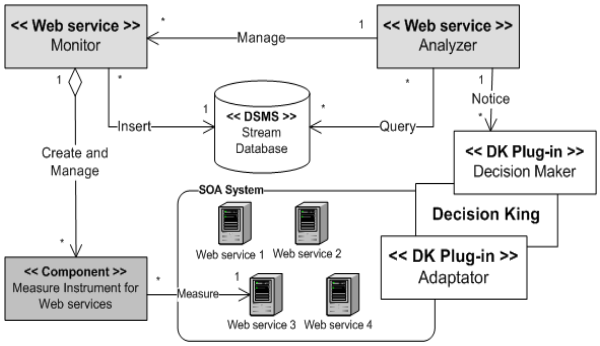

Figure 3: Implementation of SALMon architecture

Finally the Decision Maker service will be developed as a set of two plug-ins for an existing tool called Decision King [12], the first plug-in is to make an interconnection layer between the Analyzer and Decision King and the second one is for the adaptation of the monitored SOA system.

4. Conclusion

Being able to build self-adaptive SOA systems is a major undertaking that requires tools to evolve. In the context of SOA systems the dynamic changes are needed in order to keep fulfilling the QoS requirements stated in SLAs. SALMon provides a method based in the current SLA standards and the monitored information to make self-adapting SOA systems.

The SALMon architecture can be used for all type of services due to its high technologic independence.

SALMon services have general interfaces that allow us to adapt existing tools to be used as part of our architecture. We are demonstrating this in our current implementation for Web services.

Because SOA systems many times are composed of services with different technologies, as future work we plan to support monitoring of multiple types of services using the same monitor with different kinds of Measure Instruments, so we will be able to monitor an entire heterogeneous SOA system. On the other hand, our current monitoring strategy can be labeled as active measurement, it means that we are establishing a connection to the monitored service. This method has its benefits but it is not always the best choice because it could interfere with the obtained QoS measurements, for this reason we plan to build measure instruments capable to work according to conservative strategies which won't need to establish connections but require to be placed nearer in the client or the service network.

5. Acknowledgments

This work has been supported by the research projects UPIC, TIN2004-07461-C02-01, MCyT, Spain, and SODA FIT-340000-2006-312 (PROFIT programme).

6. References

[1] M.P. Papazoglou. "Service-Oriented Computing: Concepts, Characteristics and Directions". In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, p.3, December 10-12, 2003.

[2] P. Ferguson, G. Huston, *Quality of service: delivering QoS on the Internet and in corporate networks*, John Wiley & Sons, Inc., New York, NY, 1998

[3] D. Verma, *Supporting Service Level Agreement on IP Networks*. New York: Macmillan, 1999.

[4] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, C. Riegen. 2002. "Universal Description Discovery & Integration (UDDI) Specification", http ://www.oasis-open.org, February 2005.

[5] E. Christensen et al., "Web Services Description Language (WSDL) 1.1," W3C Note, 15 Mar. 2001; http://www.w3.org/TR/wsdl (current June 2002).

[6] International Organization for Standarization. *ISO Standard 8402: Quality management and quality assurance-Vocabulary*, 1986.

[7] J.P. Carvallo, X. Franch, C. Quer. "Determining Criteria for Selecting Software Components: Lessons Learned". *IEEE Software*, 24(3), 2007.

[8] International Organization for Standarization. *ISO/IEC Standard 9126: Software Engineering – Product Quality, part 1*. 2001.

[9] J.P. Carvallo, X. Franch, C. Quer. "Towards a Unified Catalogue of Non-Technical Quality Attributes to Support COTS-Based Systems Lifecycle Activities". In *Procs. 6th International Conference on COTS-Based Systems* (ICCBSS), Banff (Canada), 2007.

[10] A. Keller, H. Ludwig (IBM). "The WSLA Framework: Specifying and Monitoring of Service Level Agreements for Web Services", IBM research report RC22456, 2002.

[11] R. Clotet, X. Franch, P. Grünbacher, L. López, J. Marco, M. Quintus, N. Seyff, "Requirements Modeling for Multi-Stakeholder Distributed Systems: Challenges and Techniques". In *Procs. 1st Int. Conf. on Research Challenges in Information Science* (RCIS), Quarzazate (Morocco), 2007.

[12] D. Dhungana, P. Grünbacher, R. Rabiser, "DecisionKing: A Flexible and Extensible Tool for Integrated Variability Modeling.". In *Procs. 1st International Workshop on Variability Modelling of Software-intensive Systems* (VaMoS), Limerick (Ireland), 2007.