An Empirical Evaluation to Identify Conflicts Among Quality Attributes in Web Services Monitoring

Jael Zela Ruiz[1,2] and Cecilia M. F. Rubira[1]

[1] Institute of Computing,
University of Campinas, Sao Paulo, Brazil
jael.ruiz@students.ic.unicamp.br
cmrubira@ic.unicamp.br
[2] National University of Saint Agustin, Arequipa, Peru

Abstract. Web service monitoring tools have become an essential component for Service Level Agreement (SLA) because they can collect real quality values of quality attributes to estimate the quality level of services. Since users monitor more than one quality attribute at the same time, quality levels are prone to vary during monitoring time, producing a conflict among quality attributes. In this study, we conduct an empirical study in order to identify potential conflicts among quality attributes during Web services monitoring. For this purpose, we monitor a *TravelAgent* service in two scenarios: 1) monitoring attributes in isolation, and 2) monitoring attributes in pairs. Bootstrapping is used to estimate the quality level for each scenario, then we compared the quality levels in order to identify degradation. The results have shown that Response Time and Accuracy are the most conflicting attributes during monitoring.

Keywords: Web Service, Quality of Service, QoS Conflict, Monitoring

1 Introduction

Quality of Service (QoS) is a major issue in Web Services because services are ussually third-party software integrated in large systems. QoS is a set of quality attrbutes (e.g. availability and performance). The QoS analysis has become a crucial task for service users, since they do not know the real quality level provided for Web services. For this task, monitoring tools are used to collect their quality values and evaluate the quality levels of Web services. Monitoring tools collect quality values by means of metrics defined for quality attributes, and applying two monitoring strategies [1]: a) *passive monitoring* is a strategy based on sniffing the interaction between Web services and their users, in order to minimize the interaction with them, and b) *active monitoring* is based on sending requests to Web services, the monitor acts as a client in order to collect the quality values of service responses. Although monitoring tools are a useful mechanism for collecting quality values, they can become a quality degradation factor for Web services by creating a stressful environment.

X. Franch, J. Ralyté, R. Matulevičius, C. Salinesi, and R. Wieringa (Eds.):
CAiSE 2017 Forum and Doctoral Consortium Papers, pp. 145-152, 2017.

The paper is organized as follows: Section 2 and 3 define our research problem and related work, respectively. Section 4 exposes the planning and execution of our evaluation. The results and analysis are in Sections 5. Threats to validity and Conclusions are presented in Section 6. and 7.

2 Problem Statement

Today monitoring tools are part of Service Oriented Architecture (SOA) infrastructure. In general, they can be set up on the service side, on the client side or as "man in the middle". Since quality attributes have different metrics to measure their quality levels, a different monitor should be used for each attribute separately. Thus, monitors can use different methods to collect quality values depending on the quality attribute they monitor. However, these methods can conflict between them producing a degradation in the quality level for one or more quality attributes. In order to identify these potential conflicts, we propose an empirical study to evaluate potential conflicting quality attributes during Web service monitoring from a viewpoint of their users.

3 Related Work

Mairiza *et al.* [2] constructed a catalogue of conflicts among 26 types of non-functional requirements (NFR) based on an extensive systematic literature review. They defined three categories of conflicts between two NFRs: 1) *Absolute conflict*, NFRs are always in conflict, 2) *Relative conflict*, NFRs are sometimes in conflict, and 3) *Never conflict*, NFRs are never in conflict.

On the other hand, Zheng *et al.* [3] conducted a large-scale distributed evaluation of many real-world Web services. They evaluated the response time and throughput of 21,358 Web services located in 89 countries. The results showed that a long response time is caused by a long transferring time or a long request processing time. While a poor average throughput is caused by poor network conditions in the client side or server side.

Mairiza *et al.* [2] identified many conflicts between NFRs during analysis phase and not during runtime execution of the systems. So, these conflicts do not necessarily correspond to emergent conflicts by the monitoring executed during runtime. On the other hand, the evaluation conducted by Zheng *et al.* [3] is not focused on the identification of conflicts between quality attributes.

4 Experiment Planning

In this section, we present the objective and hypothesis of our empirical study, as well as the independent and dependent variables, subjects and experiment planning.

4.1 Objective and Hypotheses

The objective of our empirical study is defined by following the analysis model GQM (Goal-Question-Metric) [4]. Table 1 shows the goals of our study, the questions that describe the way to achieve the goals, and the metrics for each question to validate the results in a quantitative way.

Table 1: Goal-Question-Metrics

Goals	G1. Identify potential conflicts between quality attributes during Web services monitoring.
Questions	Q1. *What is the quality level of each quality attribute monitored in isolation?* Q2. *What is the quality level of each quality attribute monitored in pairs?* Q3. *What are the quality attributes with a degraded quality level during monitoring in pairs?*
Metrics	M1. The confidence interval of the sampling distribution for the quality attributes monitored in isolation. M2. The confidence interval of the sampling distribution for the quality attributes monitored in pairs. M3. The difference between the confidence intervals of monitoring in isolation and in pairs.

We formulate our hypotheses as follows:

Null Hypotheses, H_0: The quality level of a quality attribute is degraded when monitored along with another attribute with respect to its quality level monitored in isolation.
H_0: $Q^{level}(S, A_i) > Q^{level}_{A_j}(S, A_i)$

Where $Q^{level}(S, A_i)$ represents the quality level of the attribute A_i monitored in isolation, and $Q^{level}_{A_j}(S, A_i)$ represents the quality level of the attribute attribute A_i monitored along with the attribute A_j.

4.2 Variables Selection

The independent variables for our study are a set of quality attributes with their metrics and the monitoring tool. The dependent variable is the quality level of the Web service.

Quality Attributes: Our context is to monitor Web service at runtime. So, our study focus on five quality attributes which are measurable at runtime and can be easily observable by users (Table 2).

Table 2: Quality Attributes

Attribute	Definition	Metric
Accuracy	The error rate produced by the Web service.	$(1 - \frac{nFaults}{totalRequest}) \times 100$
Availability	The probability of the Web service to be up and ready.	$(\frac{upTime}{totalTime}) \times 100$
Response Time	The required time for the Web service in response a request.	$T_{response} - T_{request}$
Reliability	The ability to perform its required function under stated conditions. Mean Time Between Failures (MTBF)	$\frac{\sum_1^{nFailures-1}(FT_i - FT_{i+1})}{nFailures}$
Robustness	The degree of a Web service to work correctly in the presence of invalid, incomplete or conflicting inputs.	$(1 - \frac{acceptedFaults}{nFaults}) \times 100$

Monitoring Tool: Currently there are many monitoring tools reported in the scientific community, such as Cremona [5], SALMon [6], WebInject [7], and FlexMonitorWS [8]. Since we are interested in the quality level perceived for users, we look for a monitoring tool which collects quality values from the viewpoint of the users. Cremona is a tool integrated into Web services which collects quality values from the service side viewpoint. WebInject is a standalone application deployed on the client side, but it only works for the *Response Time*. SALMon and FlexMonitorWS support their deployment on the service side, on the client side and as "man in the middle". For our study, we selected FlexMonitorWS since it support the creation of different monitors using monitoring profiles based on five features [8]: 1) Monitoring target (Web service, server application, server, or network), 2) Quality attributes, 3) Operation mode (interception, invocation, or inspection), 4) Monitoring frequency (continuous or periodic), and 5) Notification mode (sending messages or writing in a log file).

Quality Level: Firstly, we define formally *quality value*. A quality value is a numeric value, which is the result of applying a metric of a quality attribute in a Web service in an instance of time. We present a quality value as:

$$Q^{value}(S, A_i, M_{ij}, t) = v \tag{1}$$

where $v \in \mathbb{R}$ is the quality value of the quality attribute A_i with $i = 1, ..., n$ using the metric M_{ij} with $j = 1, ..., m$ and $n, m \in \mathbb{N}^+$ in the time t in the Web service S.

The quality level for a quality attribute is composed of a set of quality values collected in an interval of time. Since quality values present smooth variations in the time, quality levels are represented by a range of quality values within a confidence factor. For example, the response time is in the range [15 ms, 45 ms] for 95% of the cases. The quality level is defined by:

$$Q^{level}(S, A_i, M_{ij}, t1, t2) = [v_{min}, v_{max}] \text{ for } C\% \text{ of the cases} \tag{2}$$

where $t1$ and $t2$ are the interval of time (start and end time, respectively) collecting quality values, v_{min} and v_{max} are the minimum and maximum quality values, and C is the confidence factor of the quality level.

Additionally, we define three types of degradation comparing two samples of the same quality attribute collected in a different period of time: *Absolute degradation*, when there is not an intersection in the quality level of the two samples. *Relative degradation*, when there is an intersection in the quality level of the two samples. *No degradation*, when the quality level of the two samples is the same. We also define three types of conflicts between two quality attributes based on Mairiza *et. al* [2]. *Absolute conflict*, when at least one attribute has an absolute degradation. *Relative conflict*, when at least one attribute has a relative degradation. *Never conflict*, when the attributes have no degradation.

4.3 Selection of Subjects

The subject of our study is a *TravelAgent* service. This service look for hotel rooms, airline tickets and renting cars. *TravelAgent* is a service orchestration composed by three third-party services: *CarReservarion*, *FlightReservation*, and *HotelReservation*. In order to reproduce a real scenario, Web services were allocated in different locations and hosted in different operating systems. It was rented four virtual machines in Google Cloud Platform[3] (Table 3). *TravelAgent* service was installed in Apache ODE [4] and deployed in Apache Tomcat[5]. All third-party services were developed in JAVA and deployed in Apache Tomcat.

Table 3: Technical Features of *TravelAgent* service environments

	br-campinas-sp	**us-east1-c**	**us-central1-c**	**europe-west1-c**	**asia-east1-c**
Web service	– (Monitor)	TravelAgent	FlightReservation	CarReservation	HotelReservation
Location	Campinas, SP, Brazil	South Carolina, USA	Iowa, USA	St. Ghislain, Belgium	Changhua County, Taiwan
SO	GNU/Linux	Microsoft Windows	GNU/Linux	GNU/Linux	GNU/Linux
SO Distribution	Fedora release 21	Windows Server 2012 R2 Datacenter	Ubuntu 16.04 LTS	Debian GNU/Linux 8.4 (jessie)	CentOS Linux release 7.2.1511 (Core)
Number Cores	2	1	1	1	1
Processor	Intel(R) Core(TM) i3-2130 CPU @ 3.40GHz	Intel(R) Xeon(R) CPU @ 2.30GHz	Intel(R) Xeon(R) CPU @ 2.30GHz	Intel(R) Xeon(R) CPU @ 2.50GHz	Intel(R) Xeon(R) CPU @ 2.50GHz

4.4 Experiment Design

Our experiment was designed into two steps:

Step 1: Creation of Different Monitors. An independent monitor was created for each quality attribute according to Table 2 using FlexMonitorWS. Table 4 shows the monitoring profiles for each quality attribute.

[3] `https://cloud.google.com/`

[4] Orchestation Director Engine which execute business process defined in WS-BPEL.

[5] A open source web server and servlet container developed by Apache Software Foundation.

Table 4: Monitoring Profiles for the Empirical Study

Mon. Profile	Mon. Target	Quality Attribute	Operation Mode	Freq.	notification
AccMonitor	Service	Accuracy	Invocation	30 seg.	log file
AvaMonitor	Service	Availability	Invocation	30 seg.	log file
ResMonitor	Service	Response Time	Invocation	30 seg.	log file
RelMonitor	Service	Reliability	Invocation	30 seg.	log file
RobMonitor	Service	Robustness	Invocation	30 seg.	log file

Step 2: Execution. In order to identify degradation in the quality levels, monitoring was executed during 24 hours considering two scenarios:

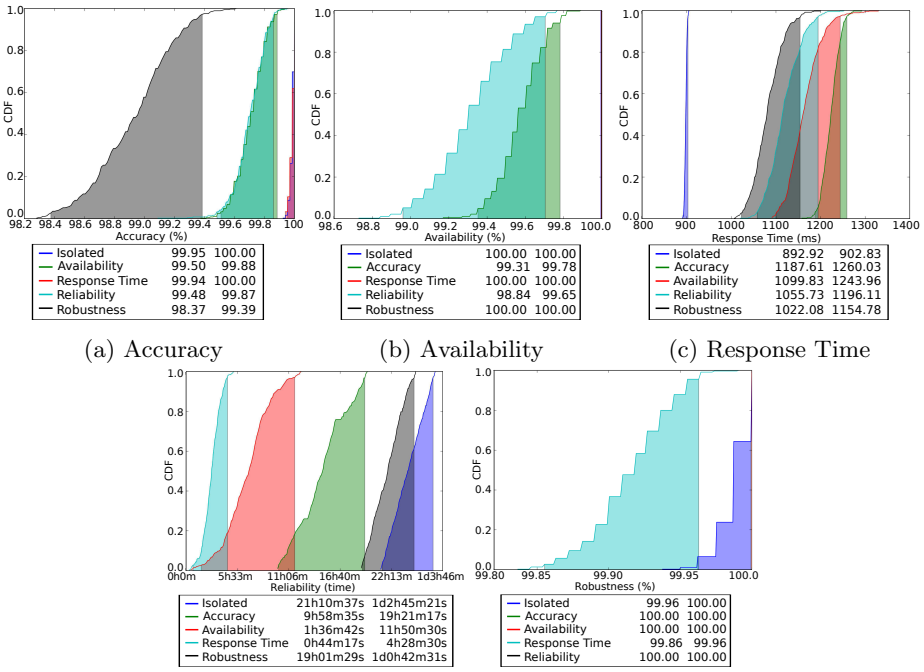
1. **Scenario 1: Monitoring in Isolation**. Every quality attribute is monitored when no other attribute is monitored. The aim of this scenario is to create a basis state of the quality level of the Web service (**Q1**).
2. **Scenario 2: Monitoring in Pairs**. All possible pairs of quality attributes are monitored at the same time in the same Web service (**Q2**). The aim of this scenario is to produce a degradation in the quality level of at least one attribute (**Q3**).

5 Results and Discussion

Since the quality level is prone to vary in the time, we estimated the magnitude of these variations and its error range by applying bootstrapping. It is a statistical technique that estimates the sampling distribution by making random re-sampling, with replacement, from the original sample [9]. The aim is to produce more samples and apply the same statistical operation (e.g. mean, median, or correlation) to every new sample. Consequently, we represented the quality level for a quality attribute using the confidence interval of the sampling distribution with a confidence factor of 95% (**M1**, **M2**). In Figure 6(a), *Accuracy* presented a quality level of [99.95%, 100.00%] when it was monitored in isolation, and [99.50%, 99.88%] when it was monitored with *Availability*. Quality levels can be also observed in the graph where every curve represents the probabilistic distribution of the quality values for each scenario[6] with a confidence factor of 95% (shadow under the curve). It is clearly observed that the quality level was degraded when it was monitored with *Availability*. A similar degradation can be observed when it was monitored with *Reliability* and *Robustness*. On the other hand, the quality level for *Accuracy* was the same when it was monitored with the *Response Time*. Degradations can be observer in every quality attributes monitored in all scenarios (**M3**). Table 5 summarizes all the identified conflicts (**G1**).

Most of the observed degradations were caused by the number of invalid responses returned by the service. Invalid responses were caused by lack of memory

[6] All collected quality values are in: `http://www.students.ic.unicamp.br/~ra153621/empirical-conflicts-qos-monitoring.html`.



(a) Accuracy (b) Availability (c) Response Time

(d) Reliability (e) Robustness

Fig. 1: Sampling distribution for monitoring *TravelAgent* service.

Table 5: Conflict Quality Attributes for *TravelAgent* service monitoring

	Accuracy	**Availability**	**Response Time**	**Reliability**	**Robustness**
Accuracy		x	o	x	x
Availability	x		o	x	o
Response Time	x	x		x	x
Reliability	x	x	x		*
Robustness	o	o	x	o	

x: absolute conflict; ***:** relative conflict; **o:** never conflict; **blank:** unexplored case

on the server side in order to execute a large number of requests. The observed degradation during Web service monitoring confirm our null hypothesis (H_0).

6 Threats to Validity

Threats to *conclusion validity* concerning the relationship between treatment and outcome. An incorrect execution of the experiment and incorrect treatment of the collected information can produce biases in the results. In order to minimize this threat, the experiment was designed in order to be reproducible by someone else, following the guidelines purposed by Wohlin [4].

Threats to *external validity* concern the possibility of generalizing our results. The generalization of our results depends on to take a representative sample of

monitoring quality values. The Web service was monitored for 24 consecutive hours, and generalize its quality behavior by using bootstrapping. However, this is limited by the subject evaluation because it was only for a single service.

7 Conclusions and Future Work

This work has presented an experimental study for Web services monitoring, in order to identify potential conflicts between quality attributes. For this aim, a set of quality attributes were monitored during 24 hours considering two scenarios: monitoring quality attributes in isolation and in pairs. The results showed that 1) *Response Time*and *Accuracy* were the most conflicting attributes since it presented quality degradation during monitoring with all the other attributes. 2) Fault injection used to monitor *Robustness* was the most intrusive technique because faults are sent to the services in order to produce errors in the service, and subjecting the service under stress condition or disabling it. 3) Active monitoring can become intrusive in the Web service generating degradation in the quality of service, since this strategy send request directly to services. New studies are necessary in order to evaluate more complex services and generalize the result for Web services.

References

1. O. Cabrera and X. Franch, "A quality model for analysing web service monitoring tools," in *The Sixth International Conference on Research Challenges in Information Science*, ser. RCIS 2012, Valencia, Spain, May 2012, pp. 1–12.
2. D. Mairiza and D. Zowghi, "Constructing a catalogue of conflicts among non-functional requirements," in *Evaluation of Novel Approaches to Software Engineering*, ser. ENASE 2010. Springer Berlin Heidelberg, 2011, vol. 230, pp. 31–44.
3. Z. Zheng, Y. Zhang, and M. Lyu, "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, January 2014.
4. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction.* Norwell, MA, USA: Kluwer Academic Publishers, 2000.
5. H. Ludwig, A. Dan, and R. Kearney, "Cremona: An architecture and library for creation and monitoring of WS-Agreements," in *Proceedings of the 2nd International Conference on Service Oriented Computing*, ser. ICSOC 2004. New York, NY, USA: ACM, November 2004, pp. 65–74.
6. M. Oriol, X. Franch, and J. Marco, "Monitoring the service-based system lifecycle with SALMon," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6507 – 6521, November 2015.
7. C. Goldberg. (2011, October) Web/HTTP Test & Monitoring Tool. Available in: http://www.webinject.org. [Accessed on 11/11/2016].
8. R. Franco, C. Rubira, and A. Nascimento, "FlexMonitorWS: Uma solução para monitoração de serviços web com foco em atributos de QoS," in *Congresso Brasileiro de Software: Teoria e Prática, 21th Sessão de Ferramentas*, vol. 2, September 2014, pp. 101–108.
9. B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap.* Chapman and Hall/CRC, 1994.