Performance and Accuracy conflict in Monitoring Tools for Web Services: a case study

Jael Zela Ruiz
Intitute of Computing
University of Campinas
Campinas, Sao Paulo, Brazil
jael.ruiz@students.ic.unicamp.br

Cecília M. Rubira
Institute of Computing
University of Campinas
Campinas, Sao Paulo, Brazil
Email: cmrubira@ic.unicamp.br

***Abstract*—Web services have become one of the most used technologies in service-oriented systems. Its popularity is due to its property to adapt to any context. As a consequence of the increasing number of Web services on the Internet and its important role in many applications today, Web service quality is a crucial requirement and demanded by service consumers. Terms of quality levels are written between service providers and service consumers to obtain some degree of quality. The use of monitoring tools to control service quality levels is very important. Quality attributes suffer variations in their values during runtime, this is produced by many factors such as memory leak, deadlock, race data, inconsistent data, etc. However, sometimes monitoring tools can impact negatively affecting the quality of service when they are not properly used and configured. This paper aims to show the impact of monitoring tools over service quality, when they are not used properly. The relationship between performance and accuracy is presented and evaluated on web services. Conflict was found between performance and accuracy, where performance was the most affected, because it presented a degradation in its quality level during monitoring.**

Keywords*—*Web Services, SOA, Quality of Service, Quality Attributes, Performance, Accuracy, Monitoring Tools.

I. Introduction

In recent years, Web service has became the most popular and used technology to build SOA applications [1]. Web services are based in a set of protocols and standards as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and Integration). Web services are distributed components which are self-contained, discoverable, reusable, composable, and have a transparent location [2]. As a result to its popularity, a increasing number of functionally similar Web services can be found on the internet [3], which entails to the service consumer to ask the question: "what are the better services?" or "which of them fits my needs?" [4]. Service consumers have a difficult task to choose an appropriate service for their requirements. Quality of Service (QoS) has become the most appropriate criterion to distinguish non-functional characteristics between equivalent Web services.

QoS is described as a number of properties, named quality attributes, which take in play the Web service quality. Some of these attributes are: availability, throughput, robustness, and integrity. A set of quality attributes compose a quality model. Several quality models have been proposed both in the research and in the industry [2] [4] [5] [6]. Web services promise quality levels based in quality models. A negociation between the service provider and the service consumer is carried out, in order to assure a specific level of QoS for Web services. A Service Level Agreement (SLA) is the result of this negotiation, where quality is defined, negotiated and tasks to assure quality are established [7]. Nevertheless, afterwards a SLA is arranged for both parties, a new question is asked: How can we be sure that the supposed QoS defined on the SLA is really satisfied?. As a consequence, monitoring tools emerged to control the Web services quality levels. Monitoring tools are based on quality model. They are used to capture, collect, filter, and analyse information from the Web service during runtime [8]. Currently, there are many monitoring tools which come from the research and the industry, such as Dynamo [9], Cremona [10], SALMon [11], WebInject [12], SOAP Monitor [13], Webmetrics Web Services Monitoring [14], etc.

However, because of the dynamic and unpredictable nature of Web services [15], quality attributes can suffer variations in their values during runtime. The relationship among quality attributes can produce conflicts between them when they are monitored at the same time. For example, in a response time and throughput scenario, response time can be a better quality value when it is monitored in isolation, than in parallel with throughput, the reason is because Web service receives a larger number of requests, producing that the Web service takes more time in respond to the user. On the other hand, throughput is also affected, because an smaller number of requests are attended by unit of time, this is due to the required time to respond each request. These conflicts are produced mainly for scalability reasons, Web services can have a lot of service consumers sending many service request at the same time. Monitoring tools become a factor else for quality attributes conflicts.

Monitoring tools can become a double-edged sword, because they are an useful QoS control tool, but they can become the principal reason for conflicts when they are not correctly configured. They can turn out to intrusive agent to the Web service, creating an stressful environment. This is important to know what is being measured, where you are monitoring, how it is being monitored, and how frequently it is monitoring. An active monitoring not correctly configured can overload the Web service and produce a drop in the response time, throughput, or availability to current consumers. The goal of this paper is discover this strife between two important quality attributes, performance and accuracy using FlexMonitorWS

Tool in an active mode over Web services.

This paper is organized as follows. In Section 2, we define monitoring tools and present FlexMonitorWS. In Section 3, we present quality models and define performance and accuracy. We present our conflict scenario to evaluate and results obtained in Section 4. And finally, in Section 5, we provide the conclusions and future works.

II. Monitoring Tools and SOA

In this section we provide a brief description about SOA and monitoring tools. We also present FlexMonitorWS, a monitoring tool for Web services used in this paper.

A. Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural style widely used in distributed applications. Different functional units, services, are connected using standardized and well-defined interfaces [16]. SOA applications are dynamic, heterogeneous, distributed and autonomous.

SOA presents three primary roles: the service provider, the service consumer, and the service broker, as shown in Figure 1. The service provider defines a service in a WSDL file, it is published in the service broker using UDDI, so the service is discoverable to service consumers. Service consumers ask for the service to the services broker, this provides the WSDL file, and the service consumer consumes the service directly. [6].

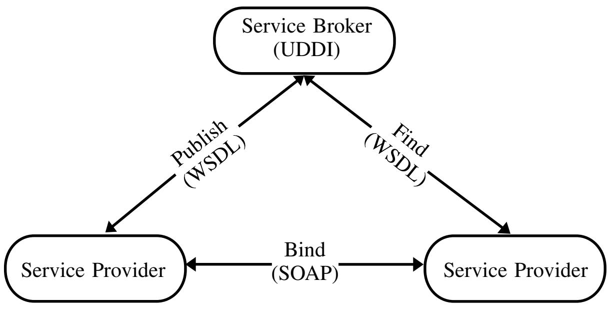

Fig. 1: SOA Architecture

B. Web Service Monitoring Tools

Monitoring tools are systems that capture, collect, filter, and analyze information from a software system in runtime [8]. On Web services, monitoring tools are used to [17]: (1) Improving the process of Web service selection and discovering, this enables QoS-based searches among functionally similar services. (2) Self-healing technique such as dynamic adaptation and dynamic recuperation applied to some quality attributes (availability, scalability, capacity and reliability) when some of then not meet the desired level. (3) To detect violations in SLA, quality metrics based on SLA are used to evaluate and control the Web service.

Monitoring tools can be used according two strategies [4]. *Passive monitoring*, monitor is an sniffer and intercept the exchanged messages between service provider and service consumer, with the aim of obtain the QoS; in this strategy a direct interaction with service provider or consumer is minimized. *Active monitoring*, monitor sent service requests directly to the service provider, acting as a consumer. Monitoring systems can be set up differently according to three components [4], as shown in Figure 2.

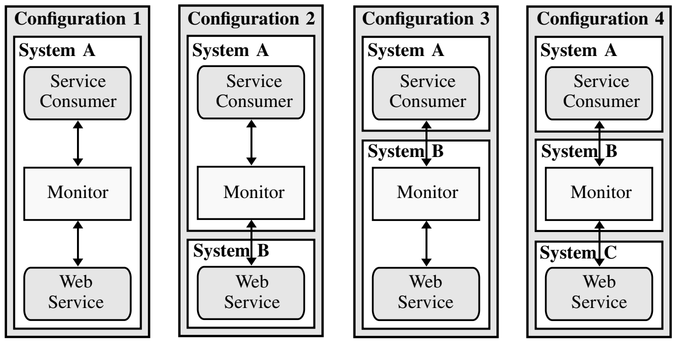

Fig. 2: Monitor configurations.

- Configuration 1: The service consumer, the monitor, and the Web service are in the same system (System A).
- Configuration 2: The service consumer and the monitor are in the same system (System A), and the Web service is in another system (System B).
- Configuration 3: The service consumer is in a system (System A), and the monitor and Web service are together in another system (System B).
- Configuration 4: The service consumer is in the System A, the monitor and the Web service are in the System B and System C, respectively.

C. Monitoring Tools Effects

Depending on how monitoring tools operate over monitored systems, they can produce risk and problems over this last one. Many researchers have reported an intrusion problem of monitoring caused by monitoring tools. This problem is due to two main reasons [18]: (1) monitoring tools consume resource, CPU, memory; (2) potential defects in monitoring tools brings risk to the monitored system, in our case, Web services.

Different methods are used to extract and collect information from Web services. These methods are divided in three types: instrument method, interceptor method, and agent approach [18]. Instrument methods are used by testing techniques. In this method, monitoring code is embedded inside monitored system, Web service, this is inserted manually by the programmers and this can be inserted in any location of the monitored code (e.g. Javassist, AspectJ). Interceptor methods are used in the middleware, they get details about all sent and received messages to the monitored system, Web service. (e.g. Interceptor in CORBA, Handler in AXIS, JVMTI in JVM). This method is more independent but it is executed in the same process with the monitored system. Agent methods are totally independent from the monitored system, Web service, running in its own process [18].

Methods bring intrusive effect to the target system in different degrees. When there are multiple monitors inserted on the target system, this become more complex, instrument mechanisms make the target code difficult to understand and maintain [18]. While interceptor mechanism can lead to performance decrease, because the monitor runs in the same process. On the other hand, the agent approach is the less intrusive method compared with the others, because it runs separately from the target system.

D. FlexMonitorWS Tool

FlexMonitorWS is a Web service monitoring tool based on Software Product Lines (SPL) [19]. This tool is based on the creation of a family of monitors to monitor different points in a Web service application and different quality attributes using different modes of monitoring. It was developed in Java language using FeatureIDE. FlexMonitorWS tool exploits the flexibility property by means of the creation of monitoring profiles which serve to a specific target and user requirements [17].

Monitoring profiles are built according to a feature model (Figure 3) which is divide in (a) monitoring target, (b) quality attributes, (c) operation mode, (d) monitoring frequency, and (e) notification mode [17] [19]. Monitoring target specifies where the monitoring takes place, they can be *Web service*, *server application*, *server*, and/or *network*. Quality attributes indicate what needs to be monitored conforming what is sought, like *availability*, *performance*, *reliability*, *accuracy*, *robustness*, *hardware state*, *failures in log file*, and/or *network QoS*. Operation mode establishes the strategy to be used, a passive monitoring by means of message *interception*, or active monitoring over *invocation* the service directly or by means of *inspection* of log files. Monitoring frequency can be *continuously* or *periodically*. Notification mode setup the method to notify the monitoring generated results, by sending *message* or *log file*. According to the selected features is generated a product, a monitor (jar file), this is executed using a properties file containing the Web service specifications.

Figure 4 shows how FlexMonitorWS works. First the monitoring target is retrieved and its type is recognized, after the monitoring frequency is configured with the help of a *Timer*, the execution is initiated based on the selected operation mode, many samples are captured and recollected continuously or periodically from the Web service, then the information is processed and analyzed according the selected quality attributes. At the end, the results are sent to the interested parties using messages or log file [17] [19].

III. Quality Attributes in SOA

Currently quality models are used in different stages of SOA system lifecycle, such as development, service discovery and selection, and service composition. There are, in the literature, different approaches available to built and select a quality model: *stakeholder approach*, this is classified according the stakeholder concerns (service consumer, service provider and service broker) [2]; *service selection approach* [6] [20]; and *monitoring approach* [4]. Many organizations have also proposed their own quality model [5] such as W3C, OASIS, WS-I, IBM, etc.

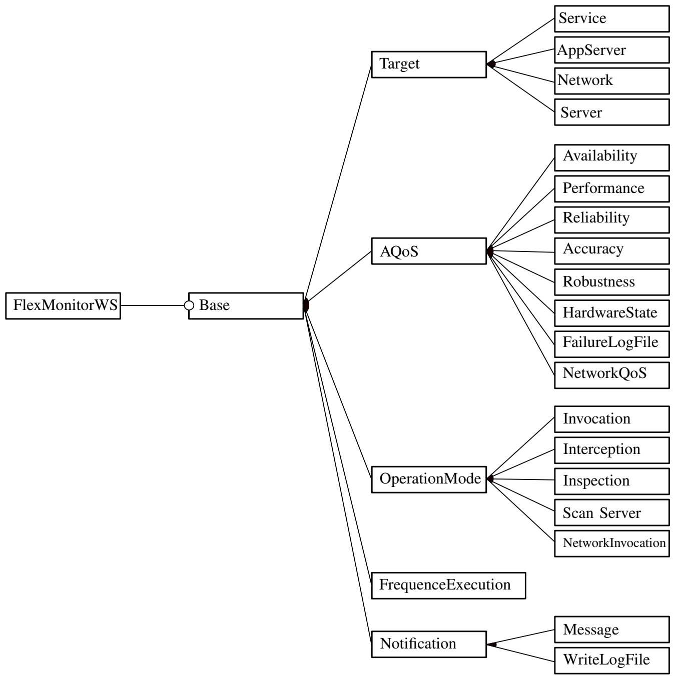

Fig. 3: FlexMonitorWS Feature Model

In this paper, we present two quality attributes, performance and accuracy. While performance is concerned in how quickly a service request can be completed, accuracy is concerned with if the service response is correct. But, correct responses are not always produced quickly. We measure the quality level for each quality attribute to evaluate their relationship and how they are affected for monitoring tools, in particular, using FlexMonitorWS tool.

A. Performance

Performance of a web service represents how fast a service request can be completed [21]. Performance can be measured in terms of throughput, response time, latency, execution time, and transaction time. Response time was selected for this work, because it is the main concern for both service consumers and service providers. It is a critical quality attribute because if a service consumer sees a large delay after send a request to the service, he or she is likely to change to another faster web service [22].

Response Time is the required time to complete a Web service request [5] [21]; the time between sending a request to the Web service and receiving the response. Response time depends primarily on two factors: network delay and server side latency. Response time is measured by the following equation:

$$ResponseTime = T_{response} - T_{request} \tag{1}$$

Where $T_{request}$ is the time (timestamp) when service request is send to the Web service, and $T_{response}$ is the time (timestamp) when the service response is received from the Web service.

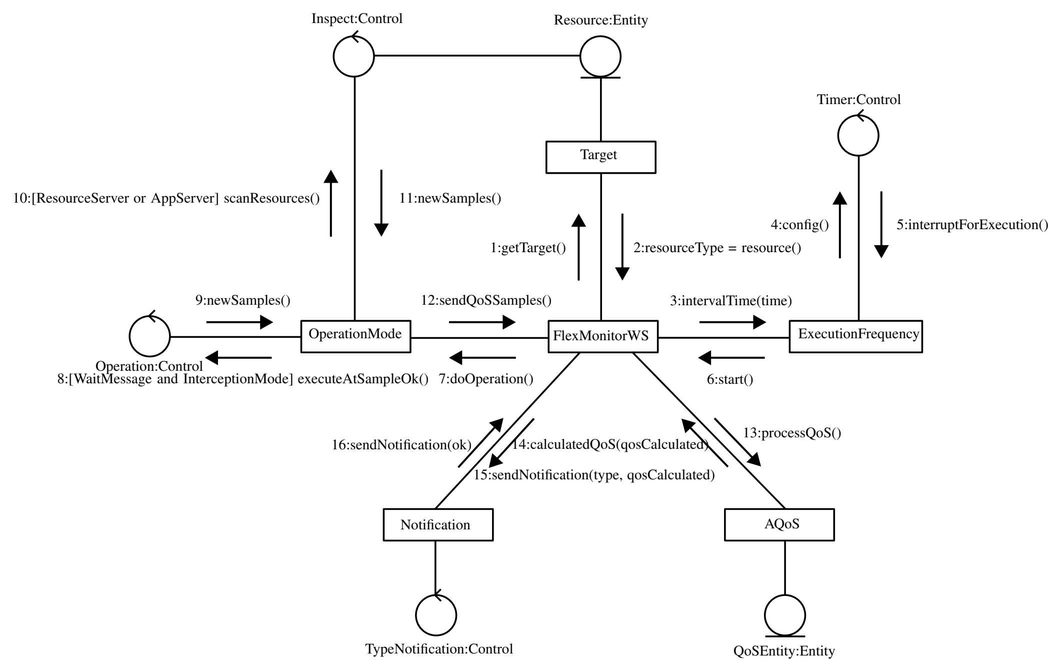

Fig. 4: FlexMonitorWS communication diagram.

B. *Accuracy*

Accuracy is the level of accurate results that Web service can give to services requests [16]. It is measured by the number of errors (error rate) produced for the Web service over a period of time [17] [21]. Accuracy is concerned about the correctness of the service response, when accuracy value is close to one, it said to be accurate, if it is close to zero, Web service is not accurate; so it loses credibility of its service consumers, when accuracy for a Web service is high.

Accuracy is measured by the following equation:

$$Accuracy = 1 - \frac{nFaults}{totalRequest} \tag{2}$$

Where $nFaults$ is the number of errors returned for the Web service, and $totalRequest$ is the number of service request sent to the Web service.

IV. Case Study Description

A Web service called *PatientService* to the domain of clinical test delivery, was developed in order to carry out our experiments. *PatientService* presents two operations: *getPatientName()* to get the patient name from a patient, and *getPatients()* to retrieve a list of the recent attended patients.

Given the Web service above, we perform experiments in order to discover possible conflicts between performance and accuracy during monitoring. Our research question have been formulated as follow:

1) What are the performance and accuracy quality levels? each one measured in isolation.
2) Is the accuracy quality level degraded when it is monitored in parallel with performance?
3) Is the performance quality level degraded when it is monitored in parallel with accuracy?

Two products (monitors) were generated using a different set of features. The first monitor called "PerfMonitor" was configured to monitor the performance attribute, where the following features were selected from the feature model in Figure 3:

- Target: *Service* (operation: *getPatients()*)
- Quality attribute: *performance*
- Operation Mode: *invocation*
- Frequency: *30 seconds*
- Notification mode: *WriteLogFile*

Second monitor called "AccMonitor" was configured to monitor the performance attribute, where the following features were selected from the feature model in Figure 3:

- Target: *Web service* (operation: *getPatientName()*)
- Quality attribute: *accuracy*
- Operation Mode: *invocation*
- Frequency: *30 seconds*
- Notification mode: *log file*

The Web service was developed using Java language and it was deployed in Apache Tomcat 7.0, and it was installed in a machine with the following configuration: AMD Phenom(tm) II P920 Quad-Core 1.60 GHz processor, with 6,00 GB main memory, Windows 7 + SP1 as system operating, and JDK 1.8. Monitors were executed in the following environment: Intel(R) Core(TM)2 Duo CPU 2.66 GHz processor, with 4.00 GB main memory, Windows 7 + SP1 system operating, and JDK 1.7.

Monitors were executed in three different cases: (1) "PerfMonitor" monitoring performance quality level over *getPa-*

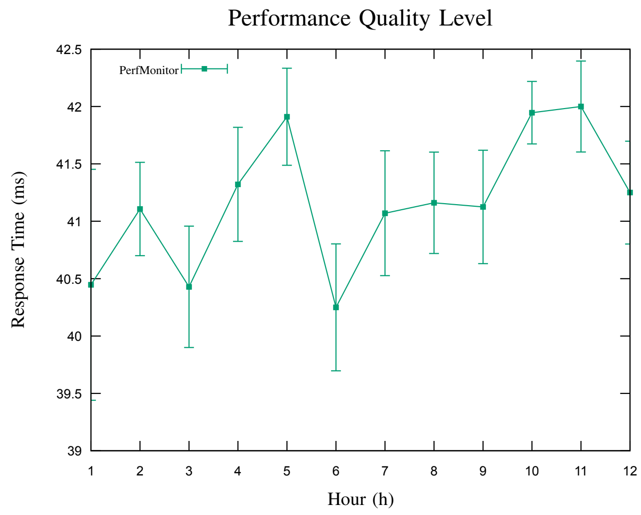

Fig. 5: Performance quality level executing "PerfMonitor".

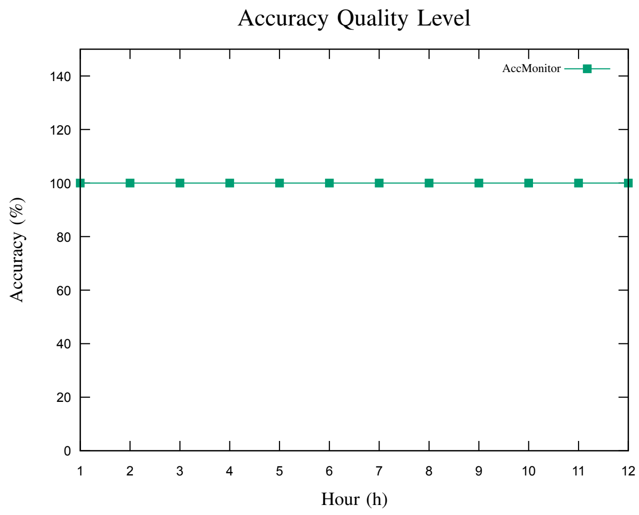

Fig. 6: Accuracy quality level executing just "AccMonitor".

tients() operation. (2) "AccMonitor" monitoring accuracy quality level over *getPatientName()* operation. (3) "PerfMonitor" monitoring performance and "AccMonitor" monitoring accuracy over *getPatients()* operation and *getPatientName()* respectively at the same time.

A. *PerfMonitor Execution*

"PerfMonitor" was executed over *getPatients()* operation of the *PatientService* Web service, to measure its performance quality level, during 12 hours. Figure 5 shows the average time by monitoring hour, time taken for the Web service to response a request from the service consumer. As an answer to our first research question, the average response time for all was 41.467 milliseconds.

B. *AccMonitor Execution*

"AccMonitor" was executed over *getPatientName()* operation on *PatientService* Web service, to measure its accuracy quality level, during 12 hours. Figure 6 shows the average accuracy percentage calculated by monitoring hour. The accuracy for *PatientService* was 100%, this responds to our first research question.

C. *PerfMonitor and AccMonitor: Parallel Execution*

"PerfMonitor" and "AccMonitor" were executed over *getPatients()* and *getPatientName()* operations respectively on *PatientService* Web service, to measures its performance and accuracy quality levels, during 12 hours. Figure 7 and Figure 8 show the average time to response a request and the average accuracy percentage by hour, when response time and accuracy are monitored in parallel. The accuracy was 100% and the response time for all was 41.208 milliseconds.

Responding to our second research question, it is easy to see that Accuracy quality level remains unchanged during the isolated monitoring and with performance monitoring in parallel, in both cases the accuracy was 100%. It means that the Web service is too strong accurate, and this ensure the correctness about its functionality.

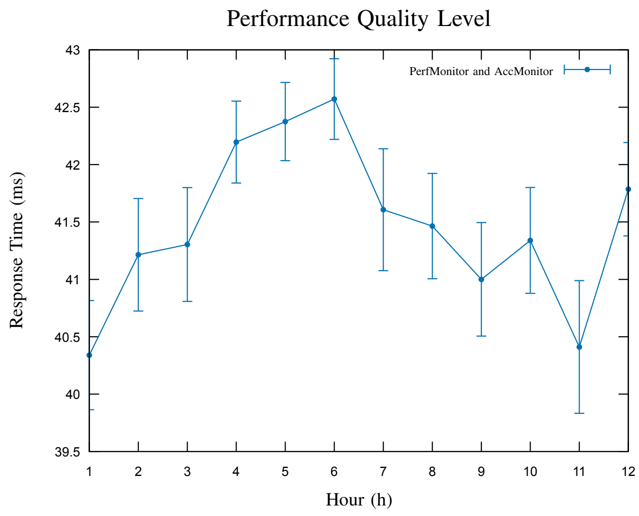

Fig. 7: Performance quality level executing "PerfMonitor" and "AccMonitor" at the same time.

On the other hand, performance quality level is affected when it is executed with accuracy monitor in parallel. Responding our third research question we found a decrease of the quality value in 0.259 milliseconds in the performance. Figure 9 shows the comparison by hour between these two cases. In order to support the difference in the results, we assess the statistical significance between "PerfMonitor" in isolation and "PerfMonitor" with "AccMonitor" results by means of a per-query paired t-test with 95% of confidence. The results of paired t-test confirms that the difference in performance is statistically significant. The Web service presents statistically a better performance when it is monitored just for "PerfMonitor".

D. *AccMonitor Execution with Fault Injection: PerfMonitor and AccFaultMonitor*

In order to perform some dependability measures and collect evidences our assumption about the strong accuracy of *PatientService*, A new monitor was generated using fault

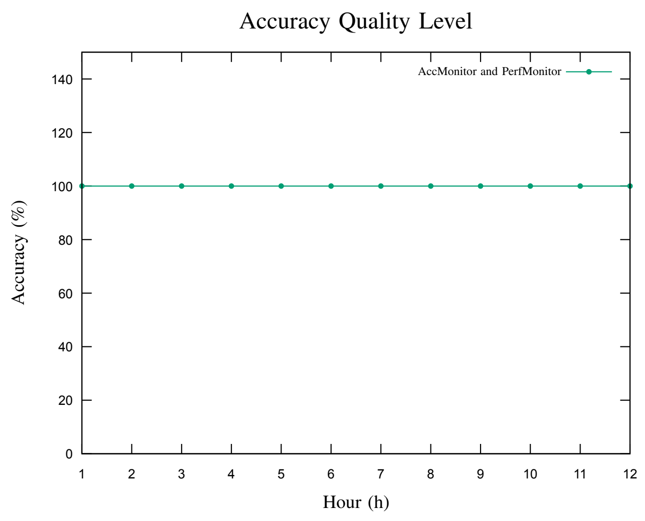

Fig. 8: Accuracy quality level executing “PerfMonitor” and “AccMonitor” at the same time.

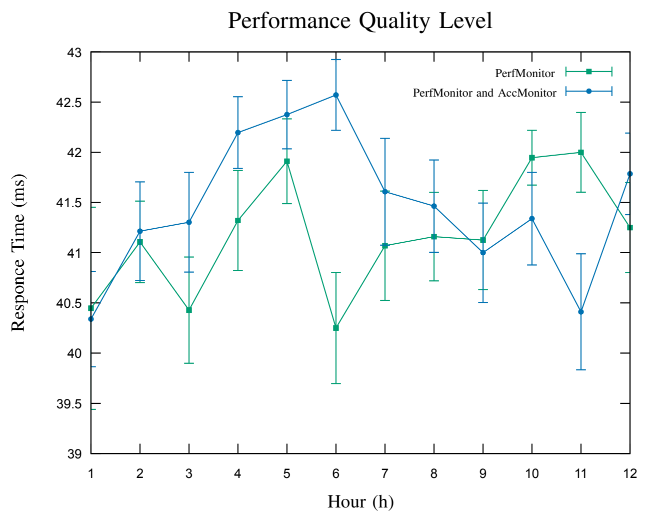

Fig. 9: Performance quality level comparative.

injection, “AccFaultMonitor”. XML injection [23] was used to generate interface faults [24]. We used two kind of injection: Parameters corruption injection and structure corruption injection. For example, we corrupted the patient code sent to *getPatientName()* operation as follows:

Fault 1:

`<arg0>PAT-0239</arg0>` to `<arg0>9320-TAP</arg0>`

we also corrupted the XML structure of the request, inverting opening and closing XML tags as follows:

Fault 2:

`<arg0>PAT-0239</arg0>` to `</arg0>PAT-0239<arg0>`

“AccFaultMonitor” was executed by 12 hours in parallel with “PerfMonitor”. Parameters corruption injection was executed in the first 4 hours, structure corruption injection for the next four hours, and in the last 4 hours, both parameter and structure corruption injections were executed.

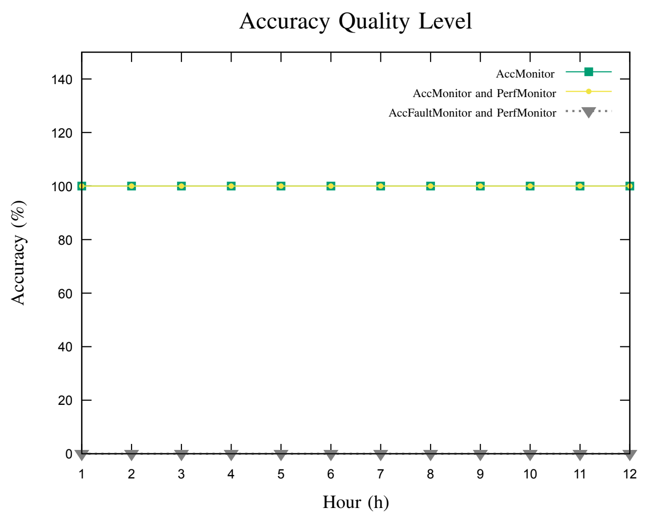

Fig. 10: Accuracy quality level in all scenarios

Figure 10 shows a comparative of the accuracy quality level in all tested scenarios. The calculated percentage for accuracy with fault injection is also displayed by monitoring hour. Corruption of the parameter values produced `java.lang.NullPointerException`. This can be a reasonable behavior because invalid data was sent to the Web service, but on the other hand, it is not a good response, because it is not an adequate response to the service consumer.

Structure corruption faults were all detected by the Web service, and it replied immediately rejecting as malformed SOAP message. Same response was obtained from the last 4 hours, injecting parameters and structure injections.

Figure 11 shows the average time by hour executed in simultaneous with “AccFaultMonitor”. Performance continue suffering a degradation in its quality level as in the previous experiment. Paired t-test between “PerfMonitor” in isolation and “PerfMonitor” with “AccFaultMonitor” in 95% confidence confirms that the Web service present statistically a better performance when performance is monitored in isolation. This is because Web service take more time trying to interpret a corrupted request, holding the processor for more time and also the used memory. This reduces the resources needed for “PerfMonitor”, and therefore taking longer to respond.

V. Conclusions

Web service monitoring tools are an important issue for the quality of Web service. When we use monitoring tools, we need identify: what is our monitoring target? what do we need monitor?, how monitor it? how often monitor it? and how notify the monitoring results?.

It is necessary to pay attention to the monitor configuration for Web service monitoring, because it can be the main reason for quality level degradation of Web services. Active monitoring (invocation) is a configuration which produce quality level degradation in Web services.

Our study has shown that response time and accuracy have conflict between them, when they are monitored at the same time. Performance is the most affected quality attribute,

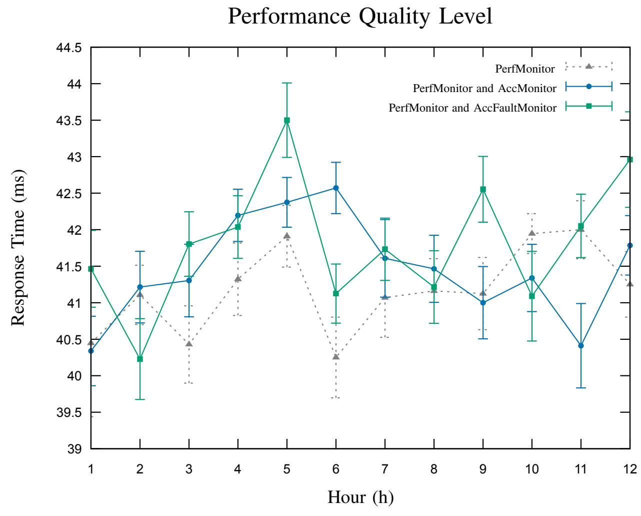

Fig. 11: Performance quality level in all scenarios

because Web service serves a higher number of requests when it is monitored in parallel with accuracy. Statistic tests confirmed a better performance level during monitoring only performance. On the other hand, accuracy was not affected because it remained unchanged in both cases, monitoring in isolation and with performance monitor in parallel.

Injection faults was added to the accuracy monitor to confirm the accuracy of *PatientService*. Parameter and structure corruption injections were not accepted by the Web service, although the exceptional responses were not adequate to the service consumers, it shows that our case study is highly accurate. However, performance needed more time to respond every request, decreasing, even more, its quality level.

Acknowledgment

This work was supported by the Distributed Systems Laboratory, at Institute of Computing, UNICAMP, Campinas, Brazil, and it could not written without support from Laura Cruz and all my friends.

References

[1] Z. Zheng, Y. Zhang, and M. Lyu, "Investigating qos of real-world web services," *Services Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 32–39, Jan 2014.

[2] Z. Balfagih and M. F. Hassan, "Quality model for web services from multi-stakeholders' perspective," in *Proceedings of the 2009 International Conference on Information Management and Engineering*, ser. ICIME '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 287–291.

[3] C. R. Choi and H. Y. Jeong, "A broker-based quality evaluation system for service selection according to the qos preferences of users," *Information Sciences*, vol. 277, no. 0, pp. 553 – 566, 2014.

[4] O. Cabrera and X. Franch, "A quality model for analysing web service monitoring tools," in *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, May 2012, pp. 1–12.

[5] M. Oriol, J. Marco, and X. Franch, "Quality models for web services: A systematic mapping," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1167–1182, Oct. 2014.

[6] M. A. Oskooei and S. M. Daud, "Quality of service (qos) model for web service selection," in *Computer, Communications, and Control Technology (I4CT), 2014 International Conference on*, Sept 2014, pp. 266–270.

[7] A. Metzger, S. Benbernou, M. Carro, M. Driss, G. Kecskemeti, R. Kazhamiakin, K. Krytikos, A. Mocci, E. Di Nitto, B. Wetzstein, and F. Silvestri, "Analytical quality assurance," in *Service Research Challenges and Solutions for the Future Internet*, ser. Lecture Notes in Computer Science, M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, Eds. Springer Berlin Heidelberg, 2010, vol. 6500, pp. 209–270.

[8] N. Delgado, A. Gates, and S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," *Software Engineering, IEEE Transactions on*, vol. 30, no. 12, pp. 859–872, Dec 2004.

[9] L. Baresi and S. Guinea, "Towards dynamic monitoring of ws-bpel processes," in *Proceedings of the Third International Conference on Service-Oriented Computing*, ser. ICSOC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 269–282.

[10] 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 '04. New York, NY, USA: ACM, 2004, pp. 65–74.

[11] D. Ameller and X. Franch, "Service level agreement monitor (salmon)," in *Composition-Based Software Systems, 2008. ICCBSS 2008. Seventh International Conference on*, Feb 2008, pp. 224–227.

[12] W. Goldberg, "Web/http test & monitoring tool," October 2011. [Online]. Available: http://www.webinject.org

[13] A. S. Fundation, "Web/http test & monitoring tool," November 2011. [Online]. Available: http://axis.apache.org

[14] N. Webmetrics, "Web/http test & monitoring tool," November 2011. [Online]. Available: http://www.webmetrics.com

[15] M. I. Ladan, "Web services metrics: A survey and a classification," in *2011 International Conference on Network and Electronics Engineering IPCSIT, vol.11 (2011) IACSIT Press, Singapore on*, 2011, pp. 93–98.

[16] T. Rajendran and P. Balasubramanie, "Analysis on the study of qos-aware web services discovery," *CoRR*, vol. abs/0912.3965, 2009. [Online]. Available: http://arxiv.org/abs/0912.3965

[17] R. J. Franco, "FlexMonitorWS: uma solução para monitoração de serviços Web com foco em atributos de QoS," Master's thesis, Institute of Computing, University of Campinas, Campinas, Sao Paulo, Brazil, 2014.

[18] Q. Wang, Y. Liu, M. Li, and H. Mei, "An online monitoring approach for web services," in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, vol. 1, July 2007, pp. 335–342.

[19] R. J. Franco, C. M. Rubira, and A. S. 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, 2014, pp. 101–108.

[20] M. A. Oskooei, S. B. M. Daud, and F. F. Chua, "Modeling quality attributes and metrics for web service selection," *AIP Conference Proceedings*, vol. 1602, pp. 945–952, 2014.

[21] K. Lee, J. Jeon, W. Lee, S.-H. Jeong, and S.-W. Park, "QoS for Web Services: Requirements and Possible Approaches," in *W3C Working Group Note 25 November 2003*, 2003. [Online]. Available: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

[22] R. Rajamony and M. Elnozahy, "Measuring client-perceived response times on the www," in *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, ser. USITS'01. Berkeley, CA, USA: USENIX Association, 2001, pp. 16–16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251440.1251456

[23] M. I. P. Salas and E. Martins, "Security testing methodology for vulnerabilities detection of xss in web services and ws-security," *Electron. Notes Theor. Comput. Sci.*, vol. 302, pp. 133–154, Feb. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2014.01.024

[24] F. Bessayah, A. Cavalli, W. Maja, E. Martins, and A. Valenti, "A fault injection tool for testing web services composition," in *Testing Practice and Research Techniques*, ser. Lecture Notes in Computer Science, L. Bottaci and G. Fraser, Eds. Springer Berlin Heidelberg, 2010, vol. 6303, pp. 137–146.