



Industrial Data

ISSN: 1560-9146

iifi@unmsm.edu.pe

Universidad Nacional Mayor de San Marcos
Perú

SANTOS LÓPEZ, FÉLIX MELCHOR
IMPLEMENTATION OF NON INTRUSIVE SOFTWARE COMPONENT TO MONITOR WEB
APPLICATIONS IN A PUBLIC ENTITY
Industrial Data, vol. 16, núm. 2, julio-diciembre, 2013, pp. 144-152
Universidad Nacional Mayor de San Marcos
Lima, Perú

Available in: <http://www.redalyc.org/articulo.oa?id=81632390016>

- How to cite
- Complete issue
- More information about this article
- Journal's homepage in redalyc.org

redalyc.org

Scientific Information System
Network of Scientific Journals from Latin America, the Caribbean, Spain and Portugal
Non-profit academic project, developed under the open access initiative

Implementation of a non intrusive software component to monitor web applications in a public entity

RECIBIDO: 12/11/13 ACEPTADO: 15/11/13

FÉLIX MELCHOR SANTOS LÓPEZ*

ABSTRACT

The importance of measuring the response time of web applications is transcendental for decision-making and/or corrections on the part of public officials involved in the departments of information systems and technologies. This research provides the conceptual framework for the development of a component J2EE (Java 2 Enterprise Edition) non-intrusive through Web Servlet filters and Interceptors of Spring Framework. It also uses the communication protocol TCP/IP for sending frames in JSON format (JavaScript Object Notation) and its subsequent reception for a stand-alone component for the record in a database. Finally, a hypothesis test between Log4j and Socket Simple technologies is performed, achieving an average record time of the plots approximately 0.05 seconds, and report that there are no statistical evidences to affirm that one is better and faster than the other.

Keywords: component, filters, interceptor, log4j, socket

IMPLEMENTACIÓN OF NON INTRUSIVE SOFTWARE COMPONENT TO MONITOR WEB APPLICATIONS IN A PUBLIC ENTITY

RESUMEN

La importancia de medir el tiempo de respuesta de las aplicaciones Web es transcendental para la toma de decisiones y/o correctivos por parte de los funcionarios públicos involucrados en los departamentos de tecnologías y sistemas de información. Esta investigación provee el marco conceptual del desarrollo de un componente J2EE (Java 2 Enterprise Edition) no intrusivo mediante filtros -Web de Servlets e Interceptors de Spring Framework. Así mismo, emplea el protocolo de comunicaciones TCP/IP para el envío de tramas en el formato JSON (JavaScript Object Notation) y su posterior recepción por un componente stand-alone para el registro en una base de datos. Finalmente, se realiza una prueba de hipótesis entre las tecnologías Log4j y Socket Simple, logrando determinar un tiempo de registro promedio de la tramas aproximado a 0.05 segundos, así como señalar que no existen evidencias estadísticas suficientes para afirmar que una sea más óptima y/o rápida que la otra.

Palabras clave: componente, filtros, interceptor, log4j, socket

INTRODUCTION

Today the need for measuring, assessing and monitoring the services provided not only by companies but also by government bodies, is of high significance in achieving corporate goals. Furthermore, in the fields of technology and information systems there are a series of mandatory regulations, as well as a set of best practices and frameworks that allow an adequate work performance and strategic alignment of the technologic area as a support to the institutional strategic plan (Peruvian Technical Standard-NTP, ISO, SOX, COBIT, COSO, etc.).

In this regard, the need to measure response times or service processing is crucial as evidence of the current situation of the organization and thus determines whether or not they meet the goals previously established by senior management. In the specific case of information technology, measuring response times of the applications or services offered via the Web, both to citizens and internal users, is an important source of data and information to make decisions, perform corrections and develops management plans of the specialized areas.

Additionally, a well-known framework like COBIT® suggests for instance the use of the Balanced Scorecard for Information Technology because it *"defines clear goals and the effect of the impact of IT also demonstrates and communicates to senior management effectiveness and delivering business value"* [10]. In other words, the information obtained from a measurement to quantify the real contribution of the information technology area to the entity and obtain a first indicator of the satisfaction level of service offered to users.

Therefore, this article illustrates the case study of the implementation of a component with the library type that adds in a non-intrusive way to monitor web applications. Its main target is to measure the initial and final time of a service, and then throw a message in the form of JSON (JavaScript Object Notation) a frame to a remote server that contains a stand-alone component responsible for receiving this message, interpret and record in a statistical

* Informatics Engineer, PUCP. Postgraduate Diploma in Audit and Security of Information Technology, UNMSM. Postgraduate Diploma in Software Engineering, UNI. J2EE Analyst at SUNAT. E-mail: fsantos@pucp.edu.pe

database. As an encoding solution arises in J2EE (Java 2 Enterprise Edition) web filter for the case of Servlets and Interceptor by AOP (Aspect Oriented Programming) for the case of Spring Framework.

Finally, sending JSON messages is done by **log4j**, a technology that allows asynchronous sending frames as “log” using TCP/IP sockets. It also evaluates the performance of **simple socket** and **log4j** through hypothesis testing by first calculating the sample size of $n=599$, then based on these samples yields an average transaction log approximately a twentieth of a second ($\bar{x} \equiv 0.05seconds$) and a significance level of $\alpha=0.05$ it is concluded that there is not statistic evidence to affirm ta superiority in performing on the part of **log4j** about **socket simple**.

1. DEFINITION OF THE SOFTWARE COMPONENT

According to Sommerville I. [14], software engineering is applied to component-based approach in the definition, implementation and composition of loosely coupled independent components within the systems. Flexible coupling means to invoke the feature via software component interfaces calls. By independent component is interpreted to be capable of being deployed in a distributed manner without the need to call external services. Here figure 1 shows a component software diagrammed with UML (Unified Modeling Language) notation.

On the other hand, among the select group of authors J2EE development experts, Dereck C. [4] defines software components as “*generic utility classes and static to be used by many applications*”, also provides the necessary guidelines to generate high quality components themselves. It recommends

that should be a short development time and effort, maintenance short expectation that are useful in many applications because they are generic as well as a reliable availability and stability.

2. WEB FILTERS AND INTERCEPTORS WITH AOP

The concept of filters appears from the 2.3 Servlet specifications. In technical terms it is established that a filter “*is an object that intercepts a message from a data source and a recipient*” [1]. That is, the filter performs an interception when it is invoked a procedure or method. In the development of web applications, a filter is a component that resides on the web server by filtering the requests and responses are passed between the client and a resource. Additionally, in the Java application development allows implementation of multiple filters which are linked and executed in a chain way. Figure 2 shows a graphic description of the application of filters.

In parallel, Spring Framework has its own implementation to conduct interceptions of calls that are called “interceptors”. As noted by H. Harrop [8] in his book, the basis of these interceptors is given by the paradigm of aspect-oriented programming (AOP).

As seen in figure 3, AOP presents a slight variation regarding the traditional paradigm of object-oriented programming (OOP). When a call is made to the technical services from applications, then it is applied an “aspect” in charge of intercepting different moments of the called to the service requesting for example the official moment and the last one of an invocation to an operation of a Java class.

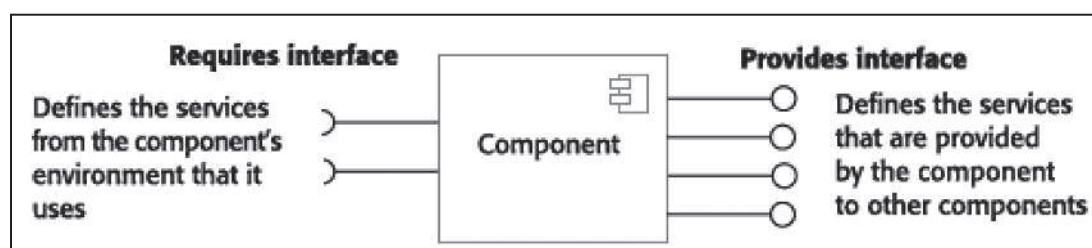


Figure 1. Graphical specification of a software component.

Source: Taken from Sommerville I. [14]

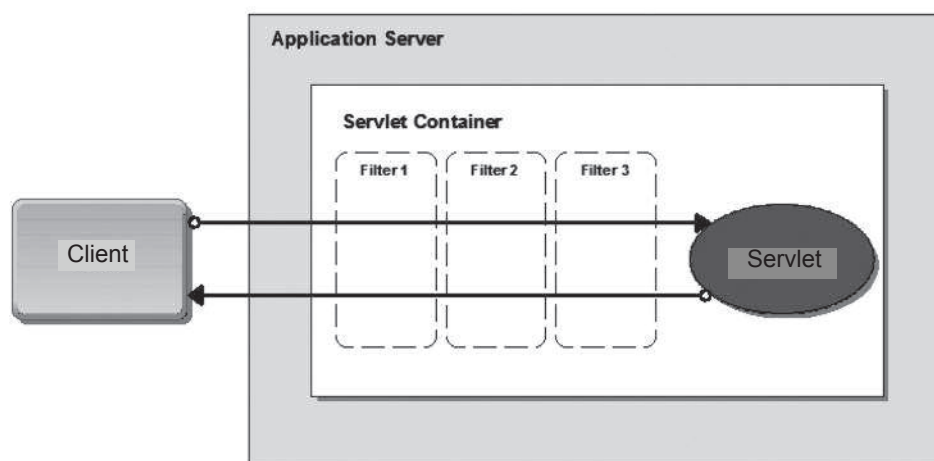


Figure 2. Servlet filters scheme.

Source: Own elaboration with Oracle Business Process Architect®

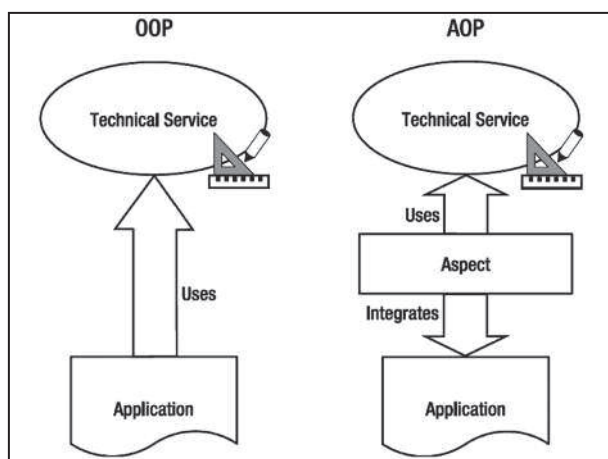


Figure 3. OOP vs. AOPP.

Source: Taken from Pawllak R [12].

3. CONNECTION BY LOG4J

Nowadays, there are no companies or organizations that do not implement the use of logs to record the register of events and/or failures occurred in their enterprise applications deployed in production. Commonly, these logs of information are stored in files that are generated and automatically renaming every day. However, not only can be stored and streamed to files, but also in libraries “logs” that provide other forms of storage and transmission to a JMS queue (Java Message Service), console, email, telnet and sockets. In the case of J2EE, the most used library in public institutions and companies is **log4j.jar**. Then the Figure 4 is shown the lines of classic configuration for the use of a log through a socket.

```

service-payment.config: Bloc de notas
Archivo Edición Formato Ver Ayuda
log4j.rootLogger=INFO, file
log4j.rootLogger=DEBUG, com.pe.timer.filter socket

log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=/logs/apps/service-payment/service-payment.log
log4j.appender.file.MaxFileSize=1MB
log4j.appender.file.MaxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=[%d] [%t] [%m]%n

log4j.appender.socket=org.apache.log4j.net.SocketAppender
log4j.appender.socket.remoteHost=192.168.1.116
log4j.appender.socket.port=9010
log4j.appender.socket.locationInfo=true
  
```

Figure 4. Log file of configuration.

Source: Own elaboration.

Gupta S. [5] mentions in his publication that *“the transfer of information to a log from one machine to another is a scenario of distributed storage”*. In other words, log4j allows to send frames to a remote server using the protocol TCP/IP.

4. JSON FORMAT FOR SENDING FRAMES

According to the findings by N. Zakas [15], JavaScript Object Notation (JSON) was technically specified as IETF RFC 4627 in 2006 by Douglas Crockford, although it had previously been used since 2001. It is noteworthy that JSON is not a programming language or a software development technology, but is a standard format for sending and receiving messages in plain text. Below in figure 5 it is shown an example of the JSON format for the proposed solution in this article:

```
{ "a" : 5,
  "b" : "2013-03-26 02:43:51.804",
  "c" : "2013-03-26 02:43:52.016",
  "d" : "10428489069FSANTOSL",
  "e" : "saveOperation",
  "f" : "{\"num_ope\" : 45034008}",
  "g" : 1 }
```

Figure 5. JSON example

Source: Own elaboration

It is noted that the text in square brackets is framed both the beginning and end of the message. Fields that are sent are described in double quotes, followed by a colon and the value to send. Each group of fields with their respective values is separated by commas.

5. SOLUTION PROPOSED FOR THE IMPLEMENTATION

Applying the concepts explained in the previous sections, we proceed to give solution to implementing a non-intrusive software component to monitor web applications.

The following items must be considered for a successful implementation:

- In the file configuration of log4j file it must be specified the classes whose package *“logs”* will be sent via a socket. For example, in line 2 of figure 4 is declared explicitly the packet **com.pe.timer.filter** and the socket appender.

- Lines below are stated the properties required by the socket appender as the remote host's IP number **192.168.1.116** and the listening port **9010** on that server. It was chosen the most stable communication protocol TCP/IP (Transmission Control Protocol / Internet Protocol) that *“belongs to the transport layer/network which were developed by the U.S. Department of Defense's Advanced Research Projects Agency Network (ARPANET) by Vinton Cerf and Bob Khan in 1974”* [3].
- As noted by Gustafson D. [6], the design is *“artistic or creative part of the software development process.”* This process converts the *“what”* into *“how”*, i.e. understanding and documenting software requirements. For this research are developed in figures 6 and 7 the design class diagrams that provide support for coding level solution.
- To design class diagrams it is used the UML (Unified Modeling Language), which allows *“bind class static content through relationships. Also shows the variables and member functions of the class”* [2]. In figure 6 it is defined the class **“BeanTimer”** with the following fields:
 - **A**→ Service code
 - **B**→ Start date of the process
 - **C**→ End date of the process
 - **D**→ Transaction user
 - **E**→ Name of the method or monitored operation
 - **F**→ Field of 255 characters to store any chain
 - **G**→ Indicator if the process was conducted successfully or not
- An example of **BeanTimer** using JSON is displayed in Figure 5. Letters are used for this bean because these data travels through the network and is required to occupy the fewest possible bits thus are used as short descriptions letters.
- On the other hand, Figure 7 shows the classes necessary to build stand-alone component responsible for receiving the frames and record on a relational database. The class diagram ListenerLog4j that by operating handles go **recibirMensaje** receiving frames them is coming to this component. It uses an infinite loop to achieve this purpose. Then ListenerLog4j creates what in the Java programming language is called *“Thread”*.

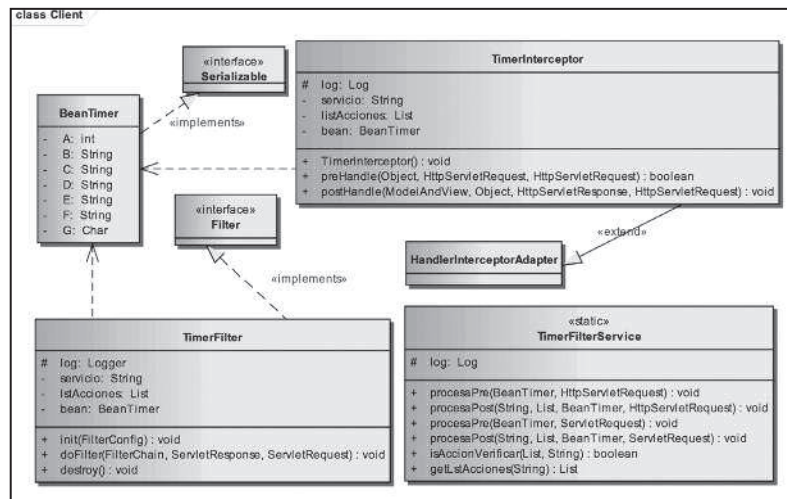


Figure 6: Class Diagram for the Client-side

Source: Own elaboration with Enterprise Architect®

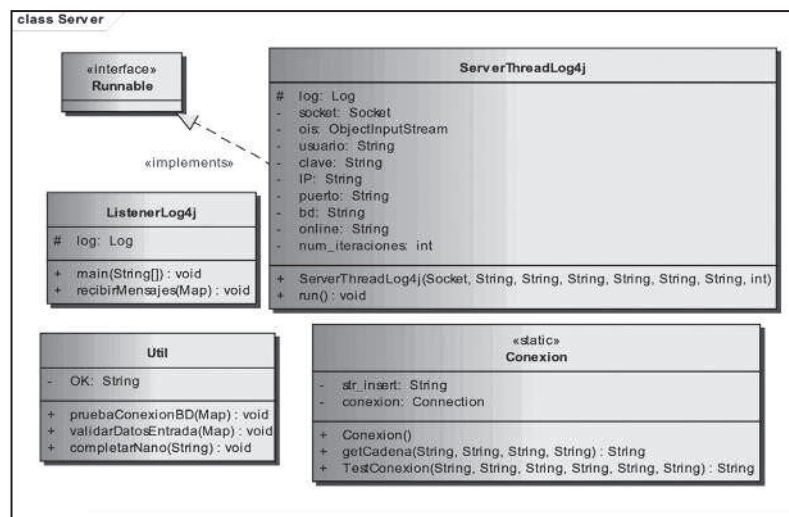


Figure 7. Class Diagram for the Client-server.

Source: Own elaboration with Enterprise Architect®

- Quoting the statement by Sierra K. [13], a Thread is divided into two parts: On the one hand is defined as any object in Java that has variables and methods, and has a beginning and end within the pile of computer memory. On the other hand, there is the so-called thread of execution is an individual process that has its own call stack. Therefore, due to the high turnout of web applications must implement the record to the database using Threads. In this way, Figure 7 shows **ServerThreadLog4j** class that implements the Runnable to implement a Thread.
- In the Figure 8 it is shown the Java code for shipping BeanTimer class instance to the remote server. On the left side implementation is illustrated by Simple Socket should indicate where in the IP source code and listen port of destination, as well as opening and closing a connection to the socket. However, on the right side displays Log4j implementation by requiring only use **log.debug** function, because the library itself is responsible for opening and closing connections, as well as use the file of configuration shown in the Figure 4.

Socket Simple	Log4j
<pre> bean.setC(new FechaBean().getTimestamp().toString()); str_json = gson.toJson(bean); cliente = new Socket(InetAddress.getByName("192.168.1.116"), Integer.valueOf("9010")); salida = new ObjectOutputStream(cliente.getOutputStream()); salida.writeObject(str_json); cliente.close(); </pre>	<pre> bean.setC(new FechaBean().getTimestamp().toString()); log.debug(gson.toJson(bean)); </pre>

Figure 8: Socket Simple and Log4j

Source: Own elaboration

- In figure 9 is illustrated the non-intrusive deployment configuration XML files developed component and must be added to all the applications to be monitored. On the left side is the configuration using servlets and filters for the configuration right for Spring AOP Framework.
- Finally, the company's technology architecture is diagrammed in figure 10. Here are displayed

calls from browsers Internet, Extranet and Intranet applications. Then a distributed server Oracle Web Logic® application receives requests for that at that moment the component intercepts the initial and final time of the process. It also shows the JSON sending messages to the remote server.

<pre> 1 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" 2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 3 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee 4 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"> 5 6 <filter> 7 <filter-name>Timer</filter-name> 8 <filter-class>com.pe.timer.TimerFilter</filter-class> 9 <init-param> 10 <param-name>saveOperation</param-name> 11 <param-value>5</param-value> 12 </init-param> 13 </filter> 14 15 <filter-mapping> 16 <filter-name>Timer</filter-name> 17 <servlet-name>Payment</servlet-name> 18 </filter-mapping> 19 20 <servlet> 21 <servlet-name>Timer</servlet-name> 22 <servlet-class>com.pe.payment.facade.PaymentLogic</servlet-class> 23 </servlet> </pre>	<pre> 1 <beans xmlns="http://www.springframework.org/schema/beans" 2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 3 xmlns:aop="http://www.springframework.org/schema/aop"> 4 5 <bean id="timer" class="com.pe.timer.TimerFilter"/> 6 7 <aop:config> 8 9 <aop:aspect ref="timer"> 10 11 <aop:pointcut id="timerExecution" 12 expression="execution(* com.pe.payment.facade.PaymentLogic.*(..))"/> 13 14 <aop:around pointcut-ref="timerExecution" method="saveOperation"/> 15 16 </aop:aspect> 17 18 </aop:config> 19 20 </beans> </pre>
---	--

Figure 9: Web Filter and AOP Interceptor

Source: Own elaboration

6. STATISTICAL SAMPLING

Due to the high concurrency, for implementing web applications should not be ignore the so-called stress tests to analyze and verify optimal implementation of service monitoring component. Available on the market a number of tools that provide interfaces for stress testing, Jmeter® being the country's leading proceeds to use this tool with detailed usage can be found in the book of Halili E [7]. Additionally, the area of technology infrastructure of the public entity

under study delivers an estimated 200 000 daily records for this component.

Applying the concepts of classical statistics proceed to calculate the sample size for a proper study of the processing time. To do this, says Hayter A. [9], the sample size for a universe of 200 000, a significance level of 0.05 ($Z_{\alpha} = 1.96$), an estimate of p unknown for which one must assume a value of 0.5 and erroneous sample percentage not exceeding 4% ($E = 0.04$) is calculated by:

$$n = \frac{Z_{\alpha}^2 (\hat{p}\hat{q})N}{NE^2 + Z_{\alpha}^2 (\hat{p}\hat{q})} = \frac{1.96^2 (0.5)(0.5)(200\ 000)}{200\ 000 (0.04^2) + 1.96^2 (0.5)(0.5)} = \frac{192\ 080}{320.9604} = 598.4539 \approx 599$$

With the sample size of 599 proceeds to obtain that size of socket simple and log4j, using Jmeter®, as

shown in Table 1. Furthermore-, the arithmetic mean and standard deviation are calculated for each sample.

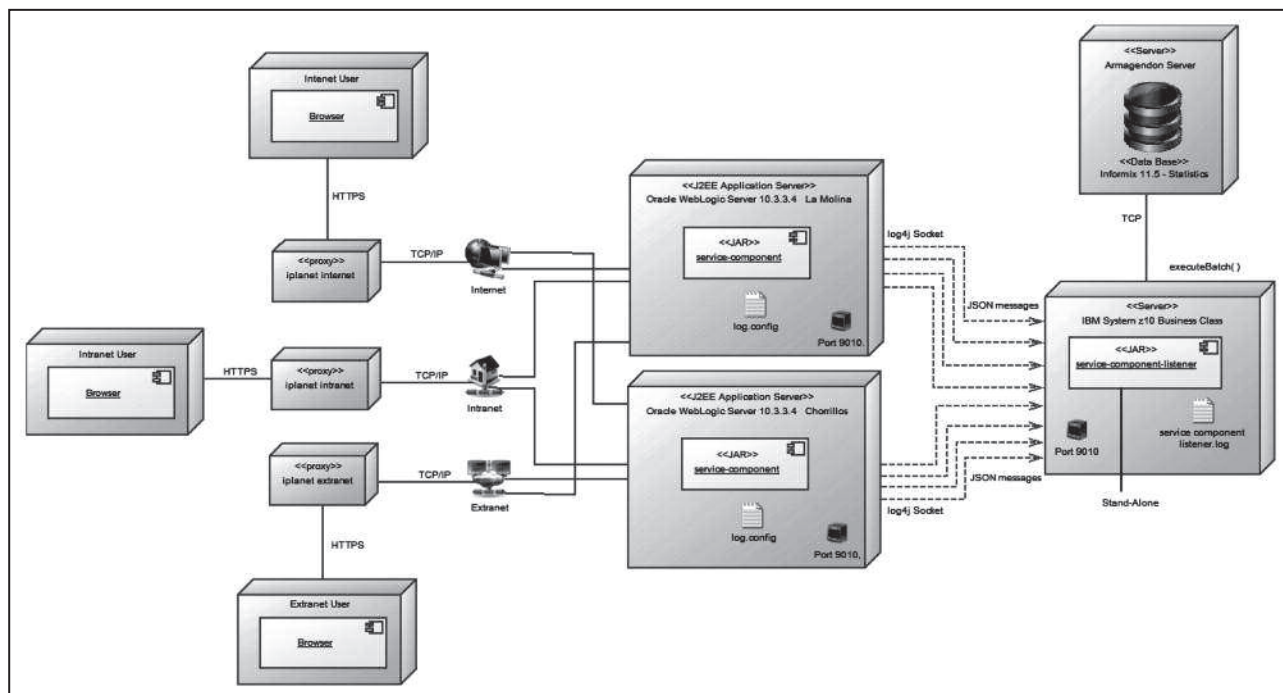


Figure 10. Diagram of technology architecture.

Source: Own elaboration with Power Designer®

Table 1. Monitoring processing times (shows the four first and last)

Test Log4j (599 samples)			Simple Socket Test (599 samples)		
Register date	End date service	Diff	Register date	End date service	Diff
2013-03-29 15:00:07.328	2013-03-29 15:00:07.259	0.069	2013-03-29 19:34:52.42	2013-03-29 19:34:52.298	0.122
2013-03-29 15:00:12.299	2013-03-29 15:00:12.252	0.047	2013-03-29 19:34:54.882	2013-03-29 19:34:54.809	0.073
2013-03-29 15:00:17.081	2013-03-29 15:00:17.04	0.041	2013-03-29 19:34:55.129	2013-03-29 19:34:55.085	0.044
2013-03-29 15:00:19.73	2013-03-29 15:00:19.671	0.059	2013-03-29 19:34:59.597	2013-03-29 19:34:59.523	0.074
.....
.....
.....
2013-03-29 15:25:14.536	2013-03-29 15:25:14.476	0.06	2013-03-29 20:01:04.508	2013-03-29 20:01:04.442	0.066
2013-03-29 15:25:15.319	2013-03-29 15:25:15.276	0.043	2013-03-29 20:01:09.101	2013-03-29 20:01:09.063	0.038
2013-03-29 15:25:18.953	2013-03-29 15:25:18.902	0.051	2013-03-29 20:01:13.971	2013-03-29 20:01:13.912	0.059
2013-03-29 15:25:21.987	2013-03-29 15:25:21.925	0.062	2013-03-29 20:01:17.065	2013-03-29 20:01:17.023	0.042

$$\bar{x}_1 = \frac{\sum_{j=1}^n x_j}{n} = \frac{\sum_{j=1}^{599} x_j}{599} = \frac{31.382}{599} = 0.05239$$

$$\sigma_1 = \sqrt{\frac{1}{n-1} \sum_{j=1}^n (x_j - \bar{x}_1)^2}$$

$$\sigma_1 = \sqrt{\frac{1}{599-1} \sum_{j=1}^{599} (x_j - 0.05239)^2} = 0.011981735$$

$$\bar{x}_2 = \frac{\sum_{m=1}^{599} x_m}{599} = \frac{30.04}{599} = 0.05014$$

$$\sigma_2 = \sqrt{\frac{1}{n-1} \sum_{m=1}^n (x_m - \bar{x}_2)^2}$$

$$\sigma_2 = \sqrt{\frac{1}{599-1} \sum_{m=1}^{599} (x_m - 0.05014)^2} = 0.011092188$$

Source: Own elaboration.

In the **Diff** column provided in table 1 is expressed in seconds the result of the subtraction of the register date and the end date service. This provides the processing time since the network sends the plot to its inclusion in the Informix® database, and socket both log4j simple. In both cases it gives an average of approximately 0.05 seconds and a standard deviation of about 0.011.

The figure 11 appreciates a diagram of linear dispersion sample for log4j. It is shown that the values are close to 0.05 seconds and the vast

majority below 0.07 which is the maximum tolerated by the public entity of this study.

HYPOTHESIS TEST

Based on the data obtained from the samples of the previous point, it proceeds to calculate success rates as shown in Table 2. Additionally, account and considered a successful time for those less than 0.07 seconds.

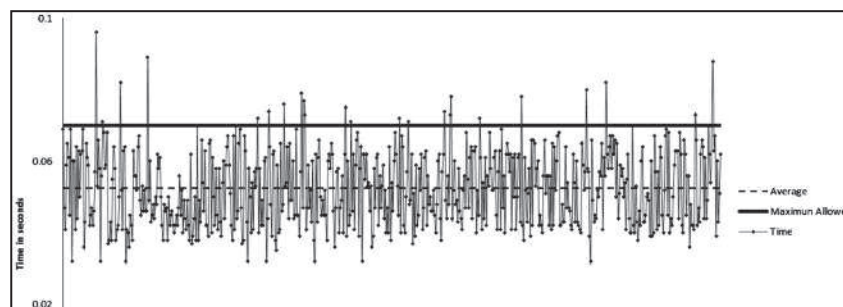


Figure 11. Diagram of the times for the samples of Log4j.

Source: Own elaboration with Microsoft Excel®

Table 2. Times of monitoring processing

	Technology	
	log4j	socket simple
Less than or equal times to 0.07 seconds	576	575
Total number of measured times	599	599
Success percentage	96.16026%	95.99332%

Source: Own elaboration.

With the values of tables 1 and 2, and assuming that this process behaves under a normal distribution statistical calculations are made for achieving determine if one technology is better than the other.

Step 1: It is defined the sample of **log4j** as p_1 and the one of the **socket simple** as p_2 .

Step 2: The assertion of a higher speed of **log4j** is expressed as $p_1 > p_2$.

Step 3: If $p_1 > p_2$ is false, then $p_1 \leq p_2$. The hypothesis are defined as:

$$H_0 : p_1 = p_2 \quad (\text{Null hypothesis})$$

$$H_1 : p_1 > p_2 \quad (\text{Alternative hypothesis})$$

Step 4: The significance level is established as $\alpha = 0.05$.

Step 5: Apply the normal distribution and calculate the estimate of the grouped sample \bar{p} .

$$\bar{p} = \frac{x_1 + x_2}{n_1 + n_2} = \frac{576 + 575}{599 + 599} = 0.96$$

$$\text{Where: } \bar{q} = 1 - 0.96 = 0.04$$

Step 6: Calculate the test statistic value expressed by the following equation:

$$z = \frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\sqrt{\frac{\hat{p}\hat{q}}{n} + \frac{\hat{p}\hat{q}}{n}}} = \frac{\left(\frac{576}{599} - \frac{575}{599}\right) - 0}{\sqrt{\frac{(0.96)(0.04)}{599} + \frac{(0.96)(0.04)}{599}}} = \frac{0.00166945}{\sqrt{0.000128214}} = 0.14743683$$

As this is a normal distribution, then you get left area for the z statistic (see table in [9]), being 0.5557.

Therefore, the value P (area to the right):

$$P = 1 - 0.5557 = 0.4443$$

Step 7: Since P is not less than the significance level ($P > \alpha$), then the null hypothesis H_0 is not rejected.

8. CONCLUSIONS AND RECOMMENDATION

- According to the results obtained in the sampling and testing of hypotheses, we conclude that in both cases both **log4j** and **simple socket** means and standard deviations are very similar and with little variation. In addition, by not rejecting the null hypothesis $H_0 : p_1 = p_2$ can not be said to be more efficient **log4j** logging response times than **simple socket**.
- The design using the UML modeling language and the implementation of good practices in software engineering allow successfully develop and implement non-intrusive component changes in the monitored applications just modifying XML files.
- As discussed throughout this article, it is concluded that the log4j implementation is the best choice because using fewer lines of code for sending frames; you configure the IP address and destination port to an external file the enterprise applications, making these parameters are subsequently modified without altering the source code. That is superior in terms of quality.
- It is highly recommendable to install the software component deployed as a library and distributed within the application server instead of including them in the "classpath" of each web application. As a result, the component is achieved centralize allowing quick and efficient updates. For this research were performed configurations and facilities suggested by Patrick R [11] in the case of Oracle Web Logic® (see pages 286, 287 and 288 for details). For other distributions should consult their own documentation.

9. BIBLIOGRAPHY

- [1] Carter J. Deshmukh H. Maliva J. (2003). SCWCD Exam Study Kit: Java Web Component Developer Certification. Manning Publications Co. U.S.A.
- [2] Cecil R. (2002). UML for Java Programmers. Prentice Hall, U.S.A.
- [3] Dennis A. Durcikova A. Fitzgerald J. (2012). Business Data Communications & Networking. John Wiley & Sons, Inc. U.S.A.
- [4] Dereck C. (2004). The J2EE Architect's Handbook: How to be a successful technical architect for J2EE applications. DVT Press, Illinois, U.S.A.
- [5] Gupta S. (2005). Pro Apache Log4j: Java application logging using the open source Apache Log4j API. Apress®, U.S.A.
- [6] Gustafson D. (2002). Theory and Problems of Software Engineering. McGraw – Hill, U.S.A.
- [7] Halili E. (2008). Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites. Packt Publishing, United Kingdom.
- [8] Harrop H. Ho C. (2012). Pro Spring 3: A comprehensive reference and practical guide to the Spring Framework. Apress®, U.S.A.
- [9] Hayter A. (2012). Probability and Statistics for Engineers and Scientists. Brooks/Cole, U.S.A.
- [10] IT Governance Institute (2003). Board Briefing on TI Governance. ITGI, U.S.A.
- [11] Patrick R. Nyberg G. Aston P. (2010). Professional Oracle Web Logic Server. Wiley Publishing. Inc, U.S.A.
- [12] Pawllak R. Retailié J. Seinturier L. (2005). Foundations of AOP for J2EE Development. APress®, U.S.A.
- [13] Sierra K. Bates B. (2008). SCJP Sun® Certified Programmer for Java 6: Study Guide (Exam 310-065). McGraw-Hill, U.S.A.
- [14] Sommerville I. (2007). Software Engineering 8. Addison-Wesley, U.S.A.
- [15] Zakas N. (2012). Professional JavaScript® for Web Developers. John Wiley & Sons, Inc. U.S.A.