Automatic Monitoring of Converged Services on a Telco 2.0 Environment based on QoS Constraints[1]

Monitoreo automático de servicios convergentes en un entorno Telco 2.0 basado en parámetros de QoS[2]

Diego Felipe Adrada Gómez[3]
Esteban David Salazar López[4]
Julián Andrés Rojas Meléndez[5]
Juan Carlos Corrales Muñoz[6]

doi:10.11144/Javeriana.iyu19-2.amoc

How to cite this article:
D.F. Adrada Gómez, E.D. Salazar López, J.A. Rojas Meléndez, and J.C. Corrales Muñoz, "Automatic Monitoring of Converged Services on a Telco 2.0 Environment based on QoS Constraints", Ing. Unv., vol. 19, no. 2, pp. 369-389, 2015. doi:10.11144/Javeriana.iyu19-2.amoc

[1] Reception date: December 7th, 2013. Acceptance date: February 9th, 2015. This article is derived form a research project called Telcomp2.0, with register code 1102-521-28338 CT458-2011, developed by the research group Grupo de Ingeniería Telemática, Cauca University, Popayán, Colombia.

[2] Fecha de recepción: 7 de diciembre de 2013. Fecha de aceptación: 9 de febrero de 2015. Este artículo se deriva de un proyecto de investigación denominado *Telcomp2.0*, con código de registro 1103-521-28338 CT458-2011, desarrollado por el grupo de investigación Grupo de Ingeniería Telemática, de la Universidad del Cauca, Popayán, Colombia.

[3] Estudiante del Departamento de Telemática de la Universidad del Cauca, Popayán, Colombia. Miembro del Grupo de investigación de Ingeniería Telemática. E-mail: diegoadrada@unicauca.edu.co

[4] Estudiante del Departamento de Telemática de la Universidad del Cauca, Popayán, Colombia. Miembro del Grupo de investigación de Ingeniería Telemática. E-mail: edsalazar@unicauca.edu.co

[5] Estudiante de Maestría en Ingeniería Telemática. Docente del Departamento de Telemática de la Universidad del Cauca, Popayán, Colombia. Miembro del Grupo de investigación de Ingeniería Telemática. E-mail: jarojas@unicauca.edu.co

[6] Doctor en Ciencias de la Computación. Docente del Departamento de Telemática de la Universidad del Cauca, Popayán, Colombia. Miembro del Grupo de investigación de Ingeniería Telemática. E-mail: jcorral@unicauca.edu.co

Abstract

The notion of converged service is based on the integration of traditional telecommunications features and Web 2.0 services. Today this concept is given special attention by the telecommunications service providers as a mechanism that allows them to expand their service portfolio and have greater market dynamism. However, these services present new challenges from the management point of view, which makes necessary to have ongoing monitoring mechanisms that allow detecting Quality of Service (QoS) constraint violations and thus, ensure proper operation. This paper introduces a monitoring scheme for converged services that is based on QoS constraints, and facilitates converged service fault-handling and operational management. It also provides an implementation through a sample converged service, which demonstrates its performance and applicability to this type of services.

Keywords

converged service; IT; communications technology; monitoring; quality of service

Resumen

La noción de servicio convergente se basa en la integración de las funcionalidades tradicionales de telecomunicaciones y los servicios de la web 2.0. Actualmente, este concepto recibe especial atención por parte de los proveedores de servicios de telecomunicaciones como un mecanismo que les permita ampliar su portafolio de servicio y, así, tener un mayor dinamismo en el mercado. Sin embargo, estos servicios presentan nuevos desafíos desde el punto de vista de su gestión y hacen necesarios mecanismos de monitoreo en tiempo de ejecución, que permitan la detección de violaciones en sus parámetros de calidad de servicio (QoS) y, así, garantizar su correcto funcionamiento. Este artículo presenta un esquema de monitoreo de servicios convergentes basado en parámetros de QoS, el cual facilita la detección de fallos y una apropiada gestión. También se proporciona una implementación del esquema a través de un servicio convergente de ejemplo que permite demostrar su rendimiento y su aplicabilidad a este tipo de servicios.

Palabras clave

validad de servicio; tecnología de la información; tecnología de las comunicaciones; monitoreo; servicio convergente

Introduction

Currently there is a trend in the telecommunications industry that has created a scenario in which a new model known as Telco 2.0 [1] has been defined. This model relates the concepts, services, and Web 2.0 technologies with the traditional telecommunications features, allowing operators to expand their service portfolio and have a greater impact on the market by reaching end-users with more complex and personalized services. This new type of services is known as converged services, due to their integration of functionalities from the telecommunications domain (voice, video, and data) with IT services from the Web domain.

For the development and execution of new converged services it is necessary to have robust platforms that allow to carry out the integration of different technologies and communications protocols characteristic of services from both, the telecommunications and Web domains. With this purpose, from the academy and the industry, different approaches have been proposed such as the SIP Servlets [2] specification, the Ericsson Converged Service Studio [3], the Alcatel-Lucent uReach CSF (Converged Services Framework) [4], among others. However, an open-source alternative that stands out is the JAIN SLEE [5] specification, which proposes a standard and robust environment for the implementation of converged services, meeting the rigorous performance requirements typical of telecom services (high availability, low latency, asynchronous behavior, etc.) and allowing its interoperability with different Web technologies and communications.

The growing complexity of the requirements in the development and deployment of converged services reveals a major challenge in the appropriate management that should be given to the computational systems that support this kind of services, giving rise to new models based on autonomic computing [6]. This paradigm was proposed by IBM and aims to automate the system management functions for them to be able to dynamically adapt to situations that constrain its proper functioning. Within these management functions is the

monitoring of components, modules, and services which are part of a system. Through monitoring it is possible to observe and register in a detailed way the system behavior, so it could be determined if all its components are working properly or if a failure has occurred.

Converged services and the different areas that comprise this concept (creation, composition, and execution) receive today special attention from the industry and academia, representing a major research focus on the computation and communications domain. Specifically, within the converged service execution area, efforts are oriented towards the development of platforms that support and meet the different requirements of these services, while ensuring proper operation. One of the main challenges in this area is to provide these platforms with effective and accurate monitoring capabilities, which allow detecting possible failures and undesired behaviors that may occur at runtime. In this way the required measures can be taken to ensure proper service operation and compliance with the QoS constraints established between providers and end-users.

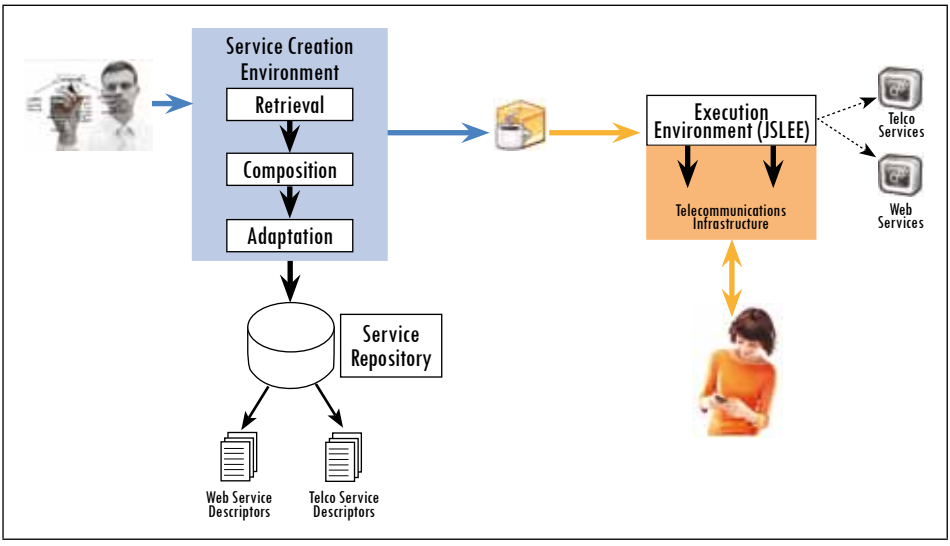
In this context, service monitoring becomes a key aspect for an appropriate management of converged services. As converged services are composed of different functionalities from the Web and telecommunications domains, it is vital to be aware of the actual status of those functionalities at all times during its execution, since a malfunctioning component may compromise the correct performance of the converged service. Therefore, it is crucial for service providers to implement service monitoring mechanisms that allow them to detect malfunctioning components at runtime, and help them to prevent QoS constraint violations and major failures in converged services.

In this paper, it is proposed and developed a runtime monitoring scheme for converged services -built on a JAIN SLEE environment- that considers services from both the Web and telecommunications domains and is based on QoS constraints. The rest of the paper is arranged as follows. Section 1 describes related technologies and analyses different approaches that address service monitoring. Section 2 presents an analysis and evaluation of QoS constraints and their applicability to converged service monitoring. Section 3 gives a detailed description of the proposed monitoring scheme. Section 4 presents a case study through which an evaluation of the proposed monitoring scheme is carried out. Finally, conclusions and future work are presented in the last section.

1. Related Work

The present work is framed within the Project TelComp2.0: *Retrieval and Composition of Complex Components for the Creation of Telco 2.0 Services* [7], funded by Colciencias and developed by the Telematics Engineering Group of the University of Cauca. The TelComp2.0 project proposes the generation of a platform aimed to support the process of creation, composition, and execution of new converged services, providing developers with tools that allow them to articulate atomic services (Web/Telco) over a unified environment for the definition of new value-added functionalities. Figure 1 presents a high level overview of the TelComp2.0 platform and its process to create new converged services.

Figure 1. TelComp2.0 high-level overview



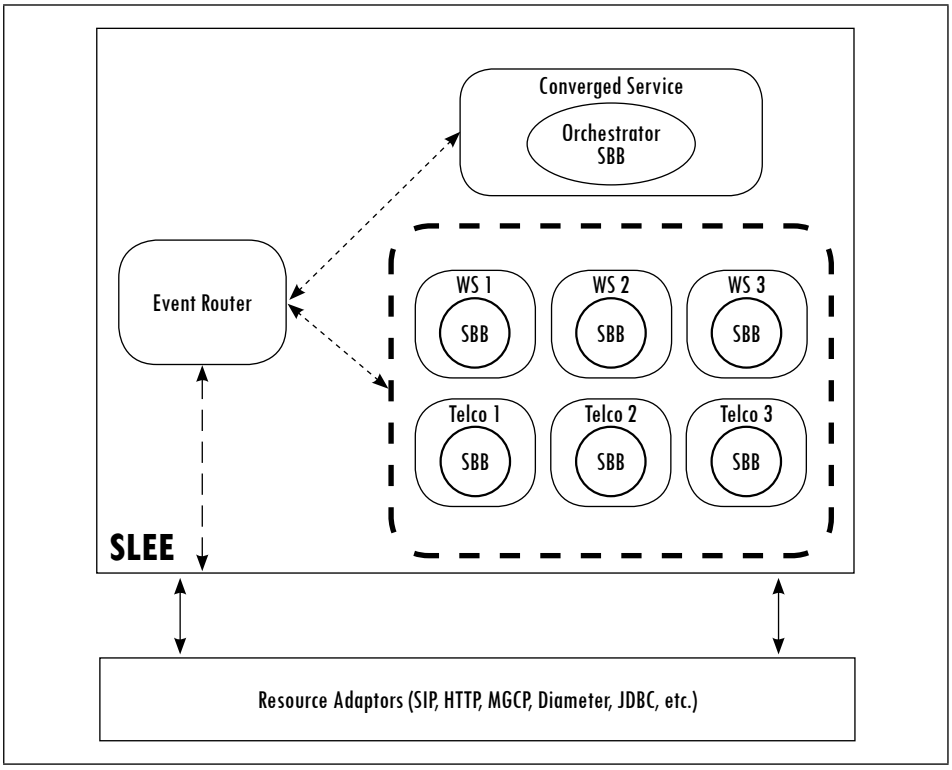
Source: authors' own elaboration

TelComp2.0 execution environment is based on the JAIN SLEE specification, since this technology defines a standard environment for executing service logic and specifies the manner in which portable and high quality telecommunications services can be built, operated, and executed.

Today, three approaches when it comes to services can be clearly identified: the first is the Web approach with the new social network services (Facebook, Twitter, LinkedIn, etc.); the second is from telecommunications operators with traditional services (voice call, SMS, video, etc.); and the third are converged services which are understood as the coordination of a set of services from different

vendors and sources, and are perceived by the end-user as a single service [8]. Taking this into account, TelComp2.0 defined a general structure for converged services over a JAIN SLEE execution environment as shown in Figure 2. The execution logic of a converged service is managed by an orchestrator SBB which communicates with a set of atomic services through firing and receiving events. These events may be provided by resource adapters or defined by each atomic service as custom events developed for specific tasks.

Figure 2. Converged service general structure

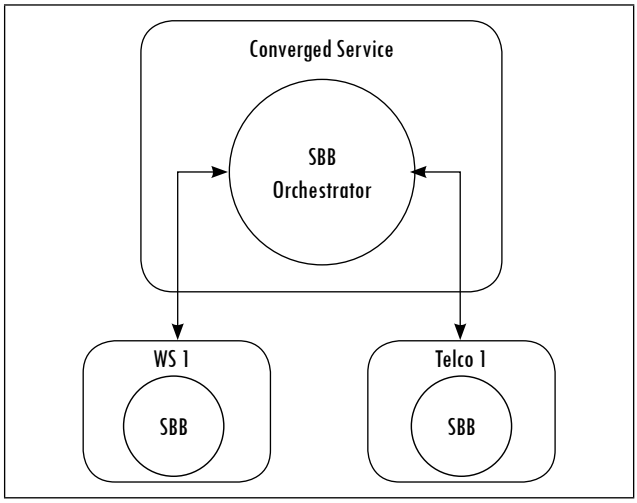


Source: authors' own elaboration

As an example, Figure 3 shows a basic converged service, composed by one Web service and one telecommunications service. As mentioned before, the orchestrator SBB is responsible for managing the control and data flow of the converged service. In other words, it manages the order of execution and the data exchange between atomic services. Once the execution of the converged service starts, the orchestrator SBB proceeds to invoke one or more atomic services, depend-

ing on the execution pattern defined by the service designer. Being driven by asynchronous events, JAIN SLEE supports concurrent execution of one or more processes at the same time. Such invocations are made through events which contain the required data for the proper operation of the atomic services. Each atomic service receives a triggering event from the orchestrator SBB and fires back a notifying event containing the results of its execution and the resulting data if any. This is a generic process implemented by all converged services in TelComp 2.0 environment, regardless of the number of component atomic services or interaction patterns. Such framework comprises a starting point for defining and implementing a monitoring scheme.

Figure 3. Implementation structure of a basic converged service



Source: authors' own elaboration

Platforms like TelComp2.0 allow an agile design, creation, and execution of converged services; however, these platforms do not provide monitoring mechanisms that help managing such services. Currently, a significant number of proposals address this matter, mainly focusing on the Web domain. Also, there is a strong trend towards the definition of composite service monitoring policies, performed at runtime, and based on QoS constraints. Below we present a description of several different approaches that address service monitoring and comprise a conceptual base for this work.

The work made by [9] presents a study of different monitoring models, in order to propose a solution that allows to monitor non-functional QoS

requirements of Web services through the use of natural language analysis in Service Level Agreements (SLA). [10] presents the analysis of a set of tools for Web service monitoring in terms of SLA and develop a model based on the quality standard ISO/IEC 25010 [11]. This model seeks to provide information that may contribute to select an adequate monitoring tool for a specific situation through the analysis of the features of each tool. Haiteng *et al*. [12] propose a monitoring architecture that seeks to clearly separate the business logic of the Web services from the monitoring associated functionalities. For this purpose, a *Monitor Broker* is introduced within the traditional Web services architecture to collect and manage information about QoS values at runtime. Goel and Shyamasundar [13] design and develop a runtime monitoring architecture where the Web service engine and the monitoring tools work concurrently, accessing the entire message exchange, without affecting the normal operation of the application being monitored. The functional properties to be monitored are defined through a SLA. Moser *et al*. [14] present a monitoring system named *VieDAME* focused on BPEL processes and considering QoS constraints. This system intercepts Simple Object Access Protocol (SOAP) messages within the BPEL process at runtime, allowing being aware of the composite application behavior with minimum impact over the service performance. Another proposal is the one presented by [15], which introduces a system to monitor and adapt SOA-based (Service Oriented Architecture) applications at runtime, named *SALMon*. This system monitors composite Web applications to detect SLA violations. Sun *et al*. [16] propose a framework for Web service monitoring based on configurable policies to define which characteristics of the Web service must be monitored. The monitoring tasks are implemented using Aspect-Oriented Programming (AOP) techniques to capture service information at runtime. Artaiam and Senivongse [17] design and develop a Web service monitoring tool as an extension of Java system application server. It is also proposed a QoS model that includes several parameters to improve QoS monitoring tasks on server side. Raimondi and Emmerich [18] present a monitoring methodology for non-functional Web service specifications such as latency and reliability. For monitoring purposes, a set of automated agents are implemented, which help to detect specific constraint violations at runtime. Finally, Ben Halima *et al*. [19] propose a framework for monitoring and analyzing Web services based on QoS constraints by intercepting SOAP messages. An algorithm is implemented to detect malfunctioning services and degradation on QoS parameters, generating health indicators for the monitored Web services.

Given the nature of the services in the telecommunications domain, which are developed to be robust and with high levels of availability, monitoring tasks focus mainly on analyzing packet traffic and network security, mostly through proprietary tools. Li *et al*. [20] propose an adaptation of composite services that supports a system in order to monitor the network environment, where its main function is to monitor the system, so that it meets the minimum execution times of a composite service established in its SLA. Falcarin and Walter [21] implement a registration framework able to insert monitoring code on JAIN SLEE applications. For this purpose it makes use of an implementation framework called JBoss-AOP. This framework allows intercepting a method at the time it is called, and insert additional code into it seamlessly. Although this work defines a mechanism for runtime monitoring of JAIN SLEE applications, it does not specify a method for handling the monitoring information or which QoS parameters must be monitored, neither an implementation of the mechanism.

The analysis carried out on the different proposals on service monitoring, presents an important gap on the application domain they are focused on, considering the concept of converged service, since most of them center their efforts on defining monitoring mechanisms exclusively on Web services. Thus, architectures and posed schemes cannot be applied to the monitoring of converged services, since both execution platforms and the features of each domain, have different operating characteristics. Also, most of the related works use QoS parameters to perform service monitoring, but there is not a unified approach that specifies which are the most important and suitable parameters that must be taken into account for converged service monitoring. Following the TelComp2.0 initiative, and considering the analysis performed on the related works, this work proposes a non-intrusive and automatic monitoring scheme that considers the converged services complexity and defines a set of QoS parameters to be monitored at runtime.

2. Quality of Service Constraints

This section presents the results of an analysis performed over different service monitoring models and tools related with both the Web and telecommunications domains, so that a set of non-functional QoS parameters that suit best the converged services requirements for monitoring can be selected. For this purpose, an evaluation of different QoS parameters is performed considering their recurrence on the monitoring models analyzed and their suitability for converged service monitoring. The non-functional QoS parameters selected for

evaluation were defined considering the ISO/IEC 25010 [11] standard and the ITU-T M.3400 [22], since these references allow to evaluate a system quality and performance from both, the software and telecommunications network management perspective. Therefore, the selected parameters are the following: *Availability, Response Time, Execution Time, Throughput, Accuracy, Reliability, Resource Usage,* and *Latency*. To carry out the evaluation of the previously mentioned parameters, the following definitions are used:

- *Nt* represents the relation between the number of times that a given QoS parameter is used within the different monitoring models analyzed and the total of models analyzed. It is defined in equation (1):

$$Nt = \frac{\#\ used}{\#\ models\ analyzed} \qquad \text{(Eq. 1)}$$

- *Ce* represents the suitability of a given QoS parameter for converged service monitoring, taking values between 0 and 1 depending on its applicability and ease of measurement for such services.
- *Wpar* represents the weight of a given QoS parameter, providing a relevance measurement for each parameter evaluated in order to select the most suitable and important ones that will be used in the proposed monitoring scheme. It is defined in equation (2):

$$Wpar = \frac{Nt + Ce}{2} \qquad \text{(Eq. 2)}$$

Ten different monitoring models and tools were considered for this evaluation, which address composite service monitoring based on QoS constraints. The previous section provided the description and major contributions of these proposals. Table 1 shows the analysis performed on the different monitoring models and tools to calculate *Nt* for each QoS parameter considered in this evaluation.

Considering the results presented in Table 1, through which *Nt* could be calculated, and establishing a value of *Ce* for each QoS parameter, it is possible to calculate the relevance (*Wpar*) for converged service monitoring of each one of them. Table 2 shows the results.

Table 1. QoS parameter consideration for composite service monitoring

Proposal	QoS parameters							
	Av	RT	ET	Tr	Ac	Re	RU	La
[20]	x	x		x			x	x
[14]	x	x			x			
[15]	x	x	x		x			
[9]	x	x		x				
[12]	x	x			x			
[13]	x	x				x		
[16]		x				x		
[17]	x					x		
[18]				x		x		x
[19]	x	x	x	x				
Total	8	8	2	4	3	4	1	2

Av: Availability; RT: Response Time; ET: Execution Time; Tr: Throughput; Ac: Accuracy; Re: Reliability; RU: Resource Usage; La: Latency.

Source: authors' own elaboration

Table 2. QoS Parameters evaluation results

QoS parameter	Nt (0-1)	Ce (0-1)	Wpar (0-1)
Availability	0.8	0.9	**0.85**
Response Time	0.8	0.9	**0.85**
Execution Time	0.2	0.3	0.25
Throughput	0.4	0.3	0.35
Accuracy	0.3	0.5	0.4
Reliability	0.4	0.8	**0.8**
Resource Usage	0.1	0.3	0.2
Latency	0.1	0.3	0.2

Source: authors' own elaboration

The evaluation results presented in Table 2 allow concluding that the most relevant QoS parameters for converged service monitoring, based on their recurrence on different proposals and their suitability for such services are *Availability, Response Time,* and *Reliability*. This is due to the fact that these three parameters represent critical constraints on the proper operation of converged services. As stated above, converged services are composed of different telecommunications and Web services forming a structure of distributed and independent applications that cooperate to provide a more complex functionality. Therefore, if one of those

components is unavailable, unreliable, or takes too long to provide an adequate response, it could compromise the entire converged service operation. Next, a formal definition of each QoS parameter is provided to proceed to the design and implementation of the converged service monitoring scheme.

- *Availability:* It is defined as the proportion of time a service is operating properly over a given interval. Its mathematical definition is given on equation (3).
- *Response Time:* Indicates the time elapsed between sending a request to a service and receiving its response. It is commonly expressed in milliseconds. Its mathematical definition is given on equation (4).
- *Reliability:* Is defined as the probability of a service to carry out its purpose correctly over a given number of requests. Its mathematical definition is given on equation (5).

Table 3 presents the mathematical definition of each QoS parameter to be used on the implementation of the proposed monitoring scheme.

Table 3. Mathematical definition of selected QoS parameters

Parameter	Definition	Notation
Availability	$Av(S) = 1 - \frac{down\ time}{up\ time}$ (3)	The downtime and uptime are measured in minutes.
Response Time	$Rt(S) = t_f(S) - t_b(S)$ (4)	Where $t_b(S)$ represents the time when the request is issued and $t_f(S)$ represents the time when the response is received.
Reliability	$Re(S) = \frac{\#\,success\ requests}{\#\,total\ request}$ (5)	

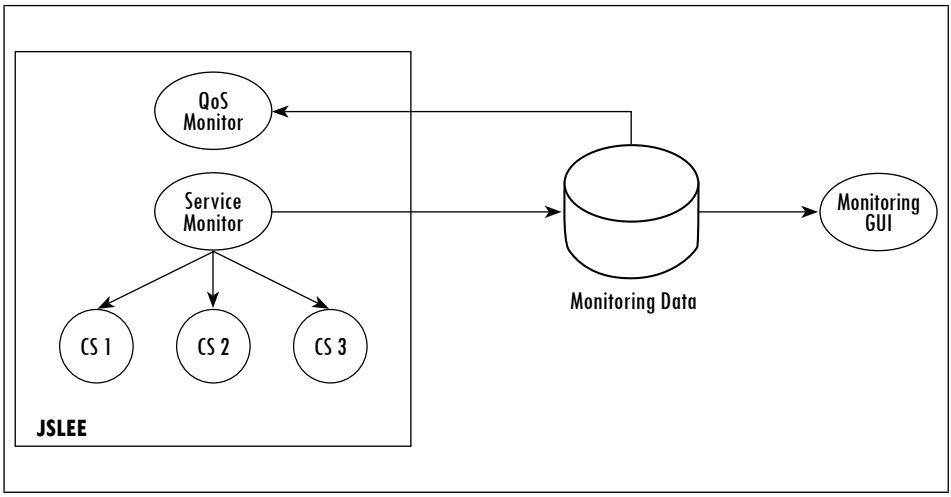
Source: authors' own elaboration

3. Proposed Monitoring Scheme

The proposed monitoring scheme is designed and implemented under the JAIN SLEE specification and is deployed along with the converged services that are going to be monitored. This scheme follows a proactive approach aiming at detecting possible QoS constraints violations by constantly monitoring and analyzing each component of the converged services currently deployed in order

to avoid major failures at runtime. Two main modules are defined for this: *Service Monitor* and *QoS Monitor*; each one of them is implemented as JAIN SLEE applications. Also, a Web application is implemented to provide a graphical view of the monitoring process. Figure 4 shows an architectural view of the converged service monitoring scheme.

Figure 4. Architectural view of the converged service monitoring scheme



Source: authors' own elaboration

A detailed description of each module is provided below:

- *Service Monitor:* It is responsible for detecting new converged service deployments on the execution environment and for gathering all the necessary monitoring data for further analysis. Detecting a new converged service deployment is possible by using the *Service Started Event* provided by the JAIN SLEE specification, which is triggered each time a new service is activated. Once a new converged service deployment is detected, this module proceeds to identify every component service, Web or telecommunications, by analyzing its orchestrator SBB class through Java Reflection techniques. Then, it executes periodical test requests to each component service, storing the results in an external database.
- *QoS Monitor:* This module is responsible for analyzing the monitoring data stored in the database and calculating the QoS parameters values to detect possible violations. This task is performed for every component service composing

currently deployed converged services, on a periodical basis. Once a violation is detected, it triggers a JAIN SLEE *Alarm Event* containing information about the malfunctioning component service and which converged services depend on the malfunctioning component.

- *Monitoring GUI:* As stated before, this module is an external Web application, implemented using Node.js, which allows observing on real time the monitoring information and QoS parameters values being retrieved by the *Service Monitor* and *QoS Monitor* modules. This application aims to provide a simple interface for managing the converged services and helping system managers to detect QoS violations, and take corrective measures to prevent major failures or undesired service behaviors.
- *Monitoring Data:* This module is responsible for storing all the data gathered during the monitoring process. It is a non-relational database built over the database engine MongoDB. Table 4 illustrates the data model implemented on this database.

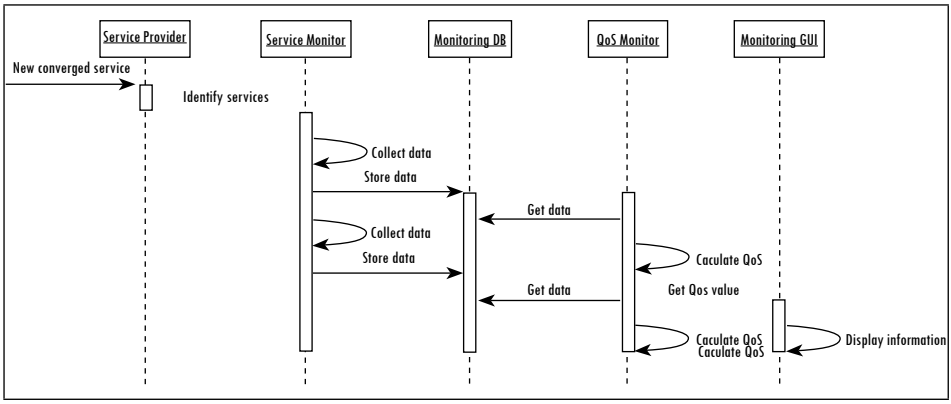
Table 4. Monitoring data model description

Name	Data Type	Description
wsdl_id	Int	Identification of the atomic service that is being monitored.
wsdl	String	WSDL address or invocation address whether is a Web or telecommunications service that is being monitored.
responseTime	Int	Elapsed time for receiving a response from a request sent to the atomic service. Is measured in milliseconds.
availability	int	Availability of the monitored atomic service. 1 means that the service is available, 0 means that the service is unavailable at the time.
convergentService	String[]	Converged services names to which is related the converged service that is being monitored.

Source: authors' own elaboration

Figure 5 presents the proposed monitoring scheme behavior through a sequence diagram that illustrates the different interactions of the modules that comprise the scheme, beginning with the deployment of a new converged service and following with the different tasks carried out to monitor it.

Figure 5. Sequence diagram of the monitoring scheme functionality

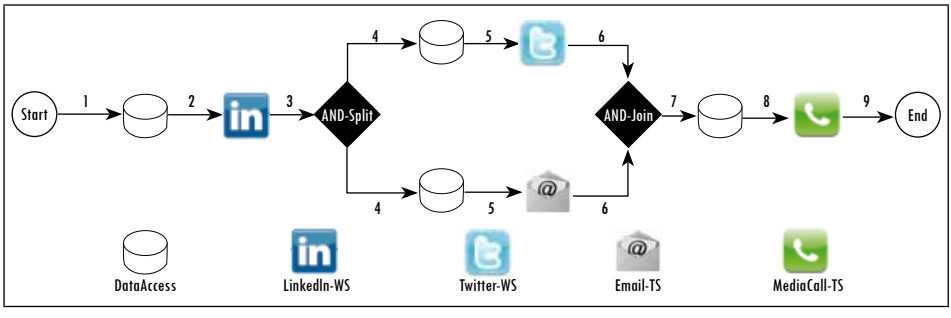


Source: authors' own elaboration

4. Evaluation

The evaluation of the proposed monitoring scheme was made using a converged service named *LinkedIn Job Notifier* which was built using the TelComp2.0 platform. This service searches for job offers associated to the user LinkedIn profile and notifies him/her through his/her Twitter account, email, and through a voice call service that takes the job offer and, using a Text-to-Speech engine, converts it to an audio message. Figure 6 presents the execution flow of each component service used in this converged service. *LinkedIn Job Notifier* provides a complex functionality that integrates Web 2.0 services (LinkedIn or Twitter) and telecommunications features (email, voice call). Specifically, it is composed by two Web services (*LinkedIn-WS, Twitter-WS*), two Telco services (*Email-TS, MediaCall-TS*), and a data access service (*DataAccess*) that retrieves the user information at runtime.

Figure 6. LinkedIn job notifier execution flow



Source: authors' own elaboration

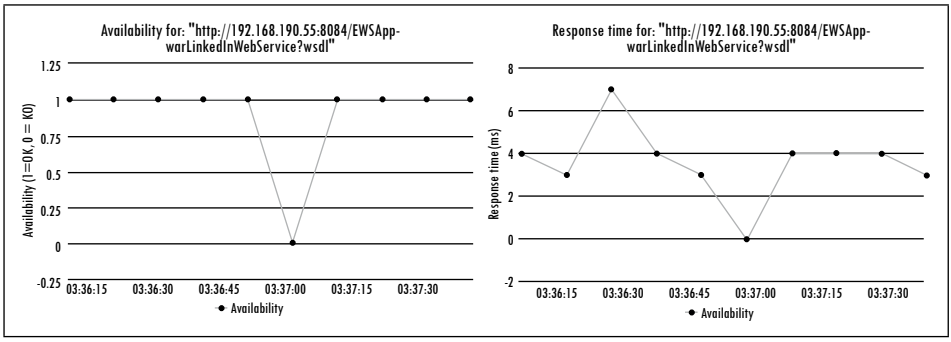
The service is triggered by the user from its mobile device. Once the service receives the initial request, it identifies the user and gets the LinkedIn user information through the *DataAccess* service (step 1). Later, it invokes the *LinkedIn-WS* to retrieve a new job offer (step 2). Following with the execution flow, the service invokes the *DataAccess* service to get user the Twitter Id and email address (step 4), and sends the job offer information through the *Twitter-WS* and *Email-TS* (step 5). Finally, it invokes the *DataAccess* service to get the user phone number (step 7) and sends the job offer as an audio message through the *MediaCall-TS* (step 8).

4.1. Functional Evaluation

For the functional evaluation of the proposed monitoring scheme, the *Service Monitor* and *QoS Monitor* modules are deployed on the Mobicents JAIN SLEE [23] execution environment, which will be standing by for new converged service deployments. Later, the *LinkedIn Job Notifier* service is deployed immediately activating the *Service Monitor* module, which analyses the orchestrator SBB class of the converged service for identifying the different services that compose it. Once identified, the *Service Monitor* module triggers requests to each composing service and stores the results of each request in the database, measuring its availability as 1 if the service is available or 0 if it is not; its reliability as 1 if the service provides a response or 0 if it does not; and its response time in milliseconds (ms) in case of receiving a response and if no response is provided, it is set as 0ms. The *Service Monitor* module performs this process every 10 seconds, storing the values obtained each time in the database.

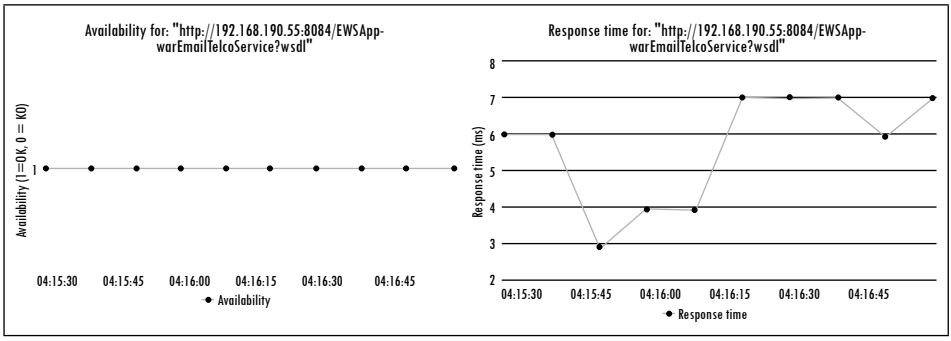
At the same time, the *QoS Monitor* module begins to access the database information to calculate the QoS values for each composing service and trigger a JAIN SLEE *Alarm Event* in case that a QoS constraint is violated. The minimum acceptable values for each QoS parameter could be configured according to the needs of the service provider and end-user. For this evaluation, the minimum accepted value for the availability of the composing services was set to 0.9, for reliability it was set to 0.9, and for response time it was set to 13 ms. On the *Monitorig GUI* application it is possible to observe in real time the monitoring results of each service. Figures 7 and 8 provide an example view of the availability and response time of a Web service (LinkedIn-WS) and a telecommunications service (Email-TS), as shown on the *Monitoring GUI* application.

Figure 7. LinkedIn-WS real time measures for availability and response time



Source: authors' own elaboration

Figure 8. Email-TS real time measures for availability and response time



Source: authors' own elaboration

As shown above, with the proposed monitoring scheme it is possible to constantly be aware of the component services behavior and to measure its QoS parameters status. The complete monitoring process performed on the *LinkedIn Job Notifier* converged service, was carried out for 30 minutes and its results are presented in Table 5.

Table 5. Monitoring results for LinkedIn job notifier converged service

Component Service	Availability	Response Time (ms)	Reliability
LinkedIn-WS	0.928	15	0.928
Twitter-WS	0.983	10	0.983
Email-TS	1	7	1
MediaCall-TS	1	12	1

Source: authors' own elaboration

The results presented in Table 5 show that all services complied with the established QoS constraints, with the exception of the response time of *LinkedIn-WS*, the average time of which was 15 ms and the acceptable response time was set to 13 ms. Because of this the *QoS Monitor* module triggered an alarm indicating that this service did not comply with the QoS constraints and the name of the converged service it composes. Figure 9 shows the resulting alarm as seen on the Mobicents JAIN SLEE management console.

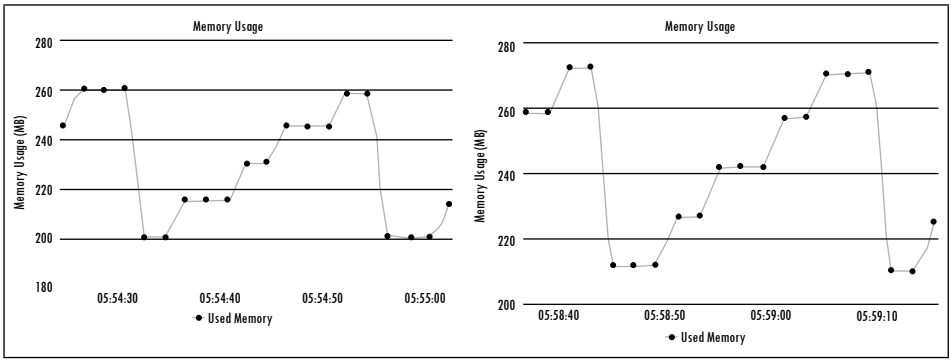
Figure 9. Alarm notification for QoS violation of LinkedIn-WS

Source: authors' own elaboration

4.2. Performance Evaluation

For the performance analysis of the proposed monitoring scheme, a Web tool named Jolokia [24] was used. This tool allows remote access to the information provided by the server Java MBeans, defined by the Java Management Extensions (JMX) specification, through a HTTP/JSON middleware, and retrieves behavioral information of the system. Specifically, for this analysis, the MBean that provides memory usage information of the server was accessed. First, the memory usage of the server with the basic component services and the *LinkedIn Job Notificator* service deployed for an interval of 10 seconds was measured. Then, the memory usage of the execution environment adding the *Service Monitor* and *Qos Monitor* modules during the same interval was measured. Figure 10 presents the results of those measurements.

Figure 10. Memory usage of the derver with (right) and without (left) the monitoring modules



Source: authors' own elaboration

The performance analysis results show that with the monitoring modules deployed and being executed, there is an average increase in the memory usage of the server of 9.4 Mb. This allows concluding that the proposed monitoring scheme has a minimum impact on the memory consumption of the converged service execution environment, thus having a negligible impact on the converged service execution performance.

Conclusions and Future Work

This paper presented a new proposal to address the monitoring of converged services considering both, Web and telecommunications domain, and taking into account QoS constraints. The proposed monitoring scheme follows a proactive and non-intrusive approach by constantly monitoring the component services that compose different converged services deployed on the execution environment, allowing detecting QoS constraint violations and generating notifications if so. The scheme also separates the monitoring functions from the application logic converged services by defining specific modules that perform the monitoring tasks without interfering with the natural execution flow of the monitored services.

This approach provides a helpful tool for system managers to be aware of the status of the different components that comprise converged services, helping them to detect undesired behaviors and taking proper action to prevent major failures without having to perform complex diagnostics and extensive server log analysis. Without the monitoring scheme, QoS constraint violations could not be detected until a major failure happens for a converged service, causing a negative impact for service providers and bad user experience.

The evaluation of the monitoring scheme shows promising results by enabling real time monitoring of converged services and having a minimum impact on the memory consumption of the execution environment, therefore having no perceptible effect on their execution performance. As future work, it has been planned to extend this work adding reconfiguration mechanisms which allow to automatically handle and correct the possible failures detected by the monitoring scheme.

Aknowledgements

The authors would like to thank University of Cauca and TelComp2.0 project (Code: 1103-521-28338 CT458-2011) for supporting and financing this work and the MSc. student Julián Andrés Rojas.

References

[1] J.-K. Yoon, "Telco 2.0. A new role and business model", *Communications Magazine, IEEE*, vol. 45, no. 1, pp.10-12, 2007.

[2] JSRs: Java Specification Requests, *JSR 289: SIP Servlet v1.1*, 2008 [online]. Available: http://jcp.org/en/jsr/detail?id=289

[3] Ericsson, *Ericsson converged service studio*, 2013 [online]. Available: http://www.ericsson.com/ourportfolio/telecom-operators/converged-service-studio

[4] uReach Technologies, *Converged services framework*, 2013 [online]. Available: http://www.ureachtech.com/csf.html

[5] JSRs: Java Specification Requests, *JSR 240: JAIN SLEE (JSLEE) v1.1*, 2008 [online]. Available: http://www.jcp.org/en/jsr/detail?id=240

[6] J. O. Kephart and D. M. Chess, "The vision of autonomic computing", *Computer*, vol. 36, no. 1, pp. 41-50, 2003 [online]. Available: http://dx.doi.org/10.1109/MC.2003.1160055

[7] Grupo de Ingeniería Telemática, *TelComp2.0 Project Website*, 2013 [online]. Available: http://190.90.112.7:8080/TelComp-SCE/

[8] I. Braun *et al*., "A monitoring and adaptation architecture for the future internet of services", in *Proc. IADIS International Conference on WWW/Internet*, Rome, Italy, pp. 67-71, 2009.

[9] M. H. Hasan, J. Jaafar and M. F. Hassan, "A review on monitoring vague quality of service (QoS) compliance for web services", in *Proc. Computer Information Science (ICCIS), 2012 International Conference on*, pp. 517-521, 2012.

[10] 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*, pp. 1-12, 2012.

[11] ISO, *ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)*, 2011 [online]. Available: http://www.issco.unige.ch/en/research/projects/ewg96/node13.html

[12] Z. Haiteng, S. Zhiqing and Z. Hong, "Runtime monitoring web services implemented in BPEL", in *Uncertainty Reasoning and Knowledge Engineering (URKE), 2011 International Conference on*, pp. 228-231, 2011.

[13] N. Goel and R. K. Shyamasundar, "Automatic monitoring of SLAs of web services", in *Proc. Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pp. 99-106, 2010.

[14] O. Moser, F. Rosenberg and, S. Dustdar, "Non-intrusive monitoring and service adaptation for WS-BPEL", in *Proc. of the 17th International Conference on World Wide Web*, New York, NY, USA, ACM, pp. 815-824, 2008 [online]. Available: http://doi.acm.org/10.1145/1367497.1367607

[15] M. Oriol *et al*., "Monitoring adaptable soa-systems using salmon", in *MONA+@ServiceWave'08*, pp. 19-28, 2008.

[16] M. Sun, B. Li and P. Zhang, "Monitoring BPEL-Based web service composition using AOP", in *ACIS-ICIS*, pp. 1172-1177, 2009.

[17] N. Artaiam and T. Senivongse, "Enhancing service-side QoS monitoring for web services", in *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on*, pp. 765-770, 2008.

[18] F. Raimondi and J. S. W. Emmerich, "A methodology for on-line monitoring non-functional specifications of web-services", in *Proc. of the First International Workshop on Property Veri□cation for Software Components and Services*, pp. 50-59, 2007.

[19] R. Ben Halima *et al*., "Non-intrusive QoS monitoring and analysis for self-healing web services", in *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, pp. 549-554, 2008.

[20] F. Li, B. Zhang and L. Ge, "Composite service adaptation supported environmental monitoring system", in *Genetic and Evolutionary Computing (ICGEC), 2011 Fifth International Conference on*, pp. 315-318, 2011.

[21] P. Falcarin and L. Walter, "An aspect-oriented approach for dynamic monitoring of a service logic excecution environment", IEC Annual Review of Communications, vol. 59, pp. 237-242, 2006.

[22] International Telecommunication Union, *ITU-T M.3400 Telecommunications Management Network*, 2000.

[23] Mobicents, *Mobicents JAIN SLEE*, 2013.

[24] R. Hub, *Jolokia*, 2013 [online]. Available: http://www.jolokia.org/features/overview.html