2016
SEPTIEMBRE

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

Priscila Cedillo Orellana

Phd Thesis

Monitorización de calidad de servicios cloud mediante modelos en tiempo de ejecución

TESIS DOCTORAL DEPOSITADA EN CUMPLIMIENTO PARCIAL DE LOS REQUISITOS PARA EL GRADO DE DOCTOR EN INFORMÁTICA

Directores: Dr. Emilio Insfrán - Dra. Silvia Abrahão

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

Tesis Doctoral depositada en cumplimiento parcial de los requisitos para el Título de Doctor por la Universitat Politècnica de València

Monitorización de calidad de servicios cloud mediante modelos en tiempo de ejecución

Irene Priscila Cedillo Orellana

Directores:

Dr. Emilio Insfrán

Dra. Silvia Abrahão

Valencia, Noviembre de 2016

Diseño de la Portada: Fabián Vélez

Título:	Monitorización de calidad de servicios cloud mediante modelos en tiempo de ejecución
Presentado por:	Irene Priscila Cedillo Orellana (priscila.cedillo@ucuenca.edu.ec)
Directores:	Dr. Emilio Insfrán (einsfran@dsic.upv.es)
	Dra. Silvia Abrahão (sabrahao@dsic.upv.es)
Institución:	Universitat Politècnica de València (UPV)
Departamento:	Sistemas Informáticos y Computación (DSIC)
Programa de doctorado:	Doctorado en Informática
Fecha de envío:	Valencia, Noviembre de 2016
Defensa:	Valencia, Enero de 2017
Revisores externos:	
	Dra. Nelly Bencomo, Aston University, UK Dra. Marcela Género, Universidad de Castilla-La Mancha, España Dr. Luis Olsina, Universidad Nacional de La Pampa, Argentina.
Tribunal:	
	Dr. Juan María Hernández Núñez, Universidad de Extremadura, España Dr. Félix García Rubio, Universidad de Castilla-La Mancha, España Dra. Nelly Bencomo, Aston University, UK

A mis pequeñas July y Paula

Agradecimientos

Al finalizar este trabajo, quizás aquel que marca un hito muy importante en mi vida, quiero agradecer a todas aquellas personas que de una u otra forma, han aportado para que pueda recorrerlo a pesar de todos los inconvenientes:

A mis directores de tesis, Silvia y Emilio, por sus enseñanzas, enriquecedoras en todo sentido.

A Freddy, mi compañero de vida y la persona que ha estado junto a mí en todo momento.

A mi pequeña Juliana, por existir y estar presente siempre.

A Paula por ser más que mi hermana.

A mi madre amada por estar ahí, como siempre, cuando más la necesito.

A mi padre, mi amigo y la fuerza que me impulsa a ser mejor.

A mis amigos de laboratorio, que han sido mis compañeros de tristezas y alegrías (Abel, Adrián, Alex, Fernando, Javier González, Javier Jiménez, José Antonio, Kamil, Patricia, Raphael, Sonia, Vicente)

A mis mejores amigos valencianos: Rosa y Javier, por su amistad y cariño.

A Javier Jiménez por todo su apoyo en la implementación de la infraestructura y por su amistad.

A todas las personas que han participado en las sesiones experimentales.

A mis amigos que han estado siempre preocupados por mí, durante mi ausencia (Adriana, Paola, Lorena, Gabriela, Zobeida, Fabián)

A mis amigos, con los que he coincidido gracias a este proyecto (Ángel, Kinda, María, Dana, Ingrid, Tiago, Nela, Andrés)

A Lorena, Fabián y Richard por su apoyo constante.

A la Universidad de Cuenca y a mi país, por la confianza depositada en mí y por darme esta oportunidad que ha enriquecido mi vida en todo aspecto.

Y por último a todas las personas que de una u otra manera han contribuido a hacer posible este trabajo.

Resumen

La computación en la nube ha traído consigo muchas ventajas derivadas de sus características particulares (auto-servicio bajo demanda, acceso amplio a la red, elasticidad, modelo de multitenencia, pago por uso, entre otros) proporcionando a sus usuarios varios beneficios pero también nuevos retos en el aprovisionamiento de servicios de hardware y software. Entre los desafíos más significativos está el aprovisionamiento adecuado y de alta calidad de los servicios que el proveedor ofrece a sus clientes. Dado el amplio número de proveedores de plataformas en la nube, se hace indispensable que éstos ofrezcan servicios de calidad, a fin de satisfacer las expectativas de sus clientes.

Las características de calidad de los servicios (*Quality of Services*, QoS) que son pactadas entre clientes y proveedores se ven reflejadas en los acuerdos de nivel de servicios (*Service Level Agreement*, SLA), que también describen las penalizaciones que se deben aplicar en caso de incumplimientos del SLA. De ahí, la necesidad de conocer el estado actual de los servicios para comprobar si los términos del SLA se cumplen.

Los métodos y herramientas de monitorización juegan un papel crucial en este contexto ya que proporcionan información sobre la utilización de los servicios y su nivel de calidad. A pesar de que existen muchas soluciones de monitorización de propósito general, éstas no se adaptan adecuadamente debido a que es habitual necesitar monitorizar aspectos específicos de los servicios con respecto a los recursos concretos de la plataforma. Por otra parte, existen soluciones de monitorización específicas para la nube, pero estas presentan limitaciones importantes a la hora de especificar necesidades de monitorización particulares ya que están centradas en monitorizar atributos de calidad de bajo nivel (uso de CPU, memoria en disco, etc) o son muy rígidas para poder modificar los requisitos en tiempo de ejecución. Además, algunas soluciones existentes están limitadas a ciertas plataformas cloud y otras no permiten explotar datos de monitorización provistos por otras herramientas ya que funcionan de manera aislada. De esta forma, surge la necesidad de un método de monitorización de servicios cloud que permita mitigar los problemas descritos y que además, explote datos obtenidos por soluciones de terceros, ya sean herramientas especializadas en monitorizar ciertos atributos de calidad, agentes, datos proporcionados por las plataformas cloud, entre otros.

La hipótesis de esta tesis es que la utilización de modelos en tiempo de ejecución (models@run.time), una técnica que se enmarca en la Ingeniería Dirigida por Modelos, puede constituir una solución apropiada ya que estos

modelos permitirán cambiar dinámicamente los requisitos de monitorización sin la necesidad de cambiar la infraestructura de monitorización. Por tanto, el principal objetivo de esta tesis doctoral es la definición y validación empírica de un método de monitorización de servicios cloud (Cloud MoS@RT) que explote los modelos en tiempo de ejecución para hacer frente a los desafíos de monitorización de servicios cloud previamente mencionados.

Además, se ha diseñado e implementado una infraestructura de monitorización que soporta el método propuesto. Para ello, se han definido: i) los componentes y artefactos que permitan la configuración de los requisitos de calidad a ser monitorizados así como la generación del modelo en tiempo de ejecución y ii) un middleware que explota este modelo para monitorizar la calidad de los servicios y generar informes de evaluación de calidad e incumplimientos del SLA. La infraestructura de monitorización ha sido diseñada mediante la utilización de estándares y lenguajes de dominio específico que permiten: i) la especificación adecuada de los requisitos de monitorización; ii) la especificación de características, atributos y métricas independientes de plataforma para evaluar dichos requisitos; iii) la especificación de métricas específicas de plataforma (Microsoft Azure, Google App Engine, Amazon AWS) así como varios escenarios de recolección de datos que integran distintas fuentes posibles de recolección de información proporcionando interoperabilidad y flexibilidad a la solución propuesta.

El método y la infraestructura de monitorización han sido validados empíricamente mediante un cuasi-experimento y tres replicaciones del mismo en España, Paraguay y Ecuador. El objetivo del cuasi-experimento realizado en total por 142 participantes ha sido el evaluar la efectividad, eficiencia, facilidad de uso percibida, utilidad percibida e intención de uso de los participantes aplicando Cloud MoS@RT como solución de monitorización. El análisis estadístico ha indicado que los participantes han considerado el método fácil de usar, útil y que emplearían el método de ser necesario en un futuro.

Finalmente, cabe destacar que esta tesis doctoral presenta una solución de monitorización innovadora, ya que al emplear modelos en tiempo de ejecución, proporciona un alto grado de flexibilidad en la especificación de requisitos de monitorización de servicios cloud, evitando la re-implementación de la infraestructura, todo esto debido a que cualquier modificación en los requisitos de monitorización se incluyen en el modelo en tiempo de ejecución. Por otro lado, la solución es interoperable debido a que permite la captura e integración de datos de monitorización desde distintos escenarios y fuentes de datos.

Abstract

Cloud computing has brought many benefits to its users that are derived from its specific characteristics (e.g., on demand self-service, broad network access, elasticity, multi-tenant model, pay-for-use). However, it also creates new challenges as regards the provisioning of software and hardware services. One of the most significant challenges is the appropriate provisioning of high quality services that cloud providers should offer to their customers. Given the large number of cloud service providers, it is essential to supply customers with services that satisfy their needs and the expected level of quality.

The characteristics of the Quality of Services (QoS) agreed upon between customers and providers are contained in Service Level Agreements (SLAs), which also describes the penalties to be applied when the SLA is violated. It is, therefore, necessary to monitor the status of services to the fulfillment of SLAs.

Monitoring methods and tools plays a crucial role in this context, as they provide information about the status of the running services and their quality level. Although there are many general purpose monitoring solutions, it is not possible to use them properly for the monitoring of specific resources in cloud environments. Moreover, there are specific solutions for cloud services with which to determine compliance with the SLA and the overall behavior and quality of cloud services. However, these solutions have significant limitations when monitoring specific individual needs because they are focused on monitoring low-level quality attributes (e.g., CPU or disk usage), and because of their rigidity as regards dealing with the modifications of monitoring requirements at runtime. In addition, other solutions are limited to certain cloud platforms or do not exploit data monitoring provided by third parties. There is thus a need for a method with which to monitor cloud services that can help to mitigate these problems, and also exploit data obtained by third-party solutions (e.g., specialized tools with which to monitor certain quality attributes, agents, data provided by platform libraries).

The hypothesis of this thesis is that the use of models at runtime, which is a Model Driven Engineering (MDE) technique, can provide an appropriate solution to this problem owing to its reflection mechanisms that decouple the model specification which contain the monitoring requirements from the monitoring infrastructure. The principal objective of this thesis is, therefore, the definition and empirical validation of Cloud MoS@RT, a method for monitoring services deployed in the cloud. This method exploits models at runtime in order

to meet the challenges of monitoring services in the cloud that are not available in other solutions of this type.

We have also designed and implemented a monitoring infrastructure that supports the proposed method. The infrastructure comprises a set of components and artifacts that allow the configuration and effective monitoring of quality requirements and the generation of the model at runtime, in addition to a middleware which exploits this model for the monitoring of the quality of cloud services and the generation of reports concerning service quality and SLA fulfillment. The monitoring infrastructure was designed by using standards and domain-specific languages that allow the following: i) the specification of monitoring requirements, ii) the specification of quality characteristics, attributes and platform-independent metrics with which to evaluate the stated requirements, and iii) the specification of platform-dependent metrics for different cloud platforms (e.g., Microsoft Azure, Google App Engine, Amazon AWS), along with several data extraction scenarios and mechanisms that integrate monitoring data from different sources, thus providing the solution with interoperability and flexibility.

The method and the monitoring infrastructure have been empirically evaluated through a quasi-experiment and three replications with 142 participants from Spain, Paraguay, and Ecuador. The goal of the experiment was to evaluate the perceived efficacy of participants when using Cloud MoS@RT to configure the monitoring of cloud services deployed on the Microsoft Azure platform. The results indicates that the participants considered the method easy to use and useful, and they also expressed their intention to use this method in the future.

Finally, it is important to emphasize that this thesis contributes to an innovative monitoring solution, owing to its high flexibility as regards specifying monitoring requirements provided by the model at runtime. The solution is also highly interoperable as it allows the extraction and integration of monitoring data from various scenarios and data sources.

Resum

La computació en el núvol (cloud computing) ha comportat molts avantatges derivades de les seues característiques particulars (autoservei baix demanda, accés ampli a la xarxa, elasticitat, model de multitenencia, pagament per l'ús, entre d'altres) proporcionant als seus usuaris diversos beneficis però també nous reptes en l'aprovisionament de serveis de maquinari i programari. Entre els desafiaments més significatius està l'aprovisionament adequat i d'alta qualitat dels serveis que el proveïdor oferix als seus clients. Donat l'ampli nombre de proveïdors de plataformes en el núvol, es fa indispensable que estos oferisquen serveis de qualitat, a fi de satisfer les expectatives dels seus clients.

Les característiques de qualitat dels serveis (Quality of Services, QoS) que són pactats entre clients i proveïdors es veuen reflecticts en els acords de nivell de servei (Service Level Agreenment, SLA) , que també descriuen les penaliaciones que s'han d'aplicar en cas d'incompliments del SLA. D'ací, la necessitat de conéixer l'estat actual dels serveris amb la fí comprovar si els termes del SLA es complixen.

Els mètodes i ferramentes de monitorització juguen un paper crucial en este context ja que proporcionen informació sobre la utilització dels serveis i el seu nivell de qualitat. A pesar que hi ha moltes solucions de monitorització de propòsit general, estes no s'adapten adequadament pel fet que és habitual necessitar monitoritzar aspectes específics dels serveis respecte als recursos concrets de la plataforma. D'altra banda, hi ha solucions de monitorització específiques per al núvol, però estes presenten limitacions importants a l'hora d'especificar necessitats de monitorització particulars ja que estan centrades a monitoritzar atributs de qualitat de baix nivell (ús de CPU i memòria en disc) o a la seua rigidesa per a modificar els requisits en temps d'execució. A més, algunes solucions existents estan limitades a certes plataformes cloud i altres no permeten explotar dades de monitorització proveïts per altres ferramentes ja que funcionen de manera aïllada. D'esta manera, sorgix la necessitat d'un mètode de monitorització de servicis cloud que permeta mitigar els problemes descrits i que a més, explote les dades obtinguts per solucions de tercers, ja siguen ferramentes especialitzades a monitoritzar certs atributs de qualitat, agents, o les dades proporcionats per les plataformes cloud, entre d'altres.

La hipòtesi d'esta tesi és que la utilització de models en temps d'execució (models@run.time), una tècnica que s'emmarca en l'Enginyeria Dirigida per Models, pot constituir una solució apropiada ja que estos models permetran canviar dinàmicament els requisits de monitorització sense la necessitat de

canviar la infraestructura de monitorització. Per tant, el principal objectiu d'esta tesi doctoral és la definició i validació empírica d'un mètode de monitorització de servicis cloud (Cloud MoS@RT) que explota els models en temps d'execució per a fer front als desafiaments de monitorització de serves cloud prèviament mencionats.

A més, s'ha dissenyat i implementat una infraestructura de monitorització que suporta el mètode proposat. Per implementar aquesta plataforma, s'han definit i) els components i artefactes que permeten la configuració dels requisits de qualitat a ser monitoritzats així com la generació del model en temps d'execució i ii) un middleware que explota este model per a monitoritzar la qualitat dels serveis i generar informes d'avaluació de qualitat i incompliments del SLA. La infraestructura de monitorització ha sigut dissenyada per mitjà de la utilització d'estàndards i llenguatges de domini específic que permeten: i) l'especificació adequada dels requisits de monitorització, ii) l'especificació de característiques, atributs i mètriques independents de plataforma per a avaluar d'aquestos requisits; iii) l'especificació de mètriques específiques de plataforma (Microsoft Azure, Google App Engine, Amazon AWS) així com alguns escenaris de recol·lecció de dades que integren distintes fonts possibles de recol·lecció d'informació proporcionant interoperabilitat i flexibilitat a la solució proposada.

El mètode i la infraestructura de monitorització han sigut validats empíricament per mitjà d'un quasi-experiment i tres replicacions del mateix a Espanya, Paraguai i L'Equador. L'objectiu del quasi-experiment realitzat per 142 participants ha sigut l'avaluar l'efectivitat, eficiència, facilitat d'ús percebuda, utilitat percebuda i intenció d'ús dels participants aplicant Cloud MoS@RT com a solució de monitorització. L'anàlisi estadística indica que els participants consideren el mètode com fàcil d'usar, útil i emprarien el mètode de ser necessari en un futur.

Finalment, cal destacar que aquesta tesi doctoral presenta una solució de monitorització innovadora, ja que a l'emprar models en temps d'execució, proporciona un alt grau de flexibilitat en l'especificació de requisits de monitorització de serveis cloud, evitant la reimplementació de la infraestructura, tot açò pel fet que qualsevol modificació en els requisits de monitorització s'inclouen en el model en temps d'execució. D'altra banda, la solució és interoperable pel fet que permet la captura i integració de dades de monitorització des de distints escenaris i fonts de dades.

Palabras Clave

Palabras Clave: Servicios Cloud, Software como Servicio, Monitorización, Calidad de Servicio, Modelo de Calidad, Acuerdos de Nivel de Servicio, Ingeniería Dirigida por Modelos, Modelos en Tiempo de Ejecución.

Key Words: Cloud Services, Software as a Service, Monitoring, Quality of Service, Quality Model, Service Level Agreements, Model Driven Engineering, Models@run.time

Paraules Clau: Servicis Cloud, Programari com un Servici, Monitorització Qualitat de Servici, Model de Qualitat, Acords de Nivell de Servici, Enginyeria Dirigida per Models, Models en Temps d'Execució.

Contenido

Índice de Figuras

Índice de Tablas

Acrónimos

API	Application Program Interface
AWS	Amazon Web Services
CPU	Central Processing Unit
DSDM	Desarrollo de Software Dirigido por Modelos
DSM	Domain Specific Modeling
EMF	Eclipse Modelling Foundation
GQM	Goal-Question-Metric
IaaS	Infrastructure as a Service
IEC	International Electrotechnical Commission.
ISO	International Organization for Standardization
IT	Information Technology
M@RT	Models@run.time/Modelo en Tiempo de Ejecución
M2T	Model to Text / Modelo a Texto
M2M	Model to Model / Modelo a Modelo
NIST	National Institute of Standards and Technology
NRF / RNF	Non-functional requirements / Requisitos no funcionales
OMG	Object Management Group
ORC	Open Reference Case
PaaS	Platform as a Service
PQM	Product Quality Management
QoS	Quality of Service / Calidad de Servicio
RF	Requisitos Funcionales

RTD	Round-trip Delay Time
RTT	Round-trip Time
SaaS	Software as a Service
SLA	Service Level Agreement
SMI	Service Measurement Index / Índice de Medición de Servicios
SOA	Service Oriented Architecture / Arquitectura Orientada a Servicios.
SPEM	Software & Systems Process Engineering Metamodel
SQuaRE	System and Software Quality Requirements and Evaluation
TGG	Gramáticas de Grafos Triple
TI	Tecnologías de la Información
WSLA	Web Service Level Agreement Language
XDR	External Data Representation / Representación Externa de Datos
XMI	XML Metadata Interchange
XML	Extensible Markup Language / Lenguaje de Marcas Extensible

Capítulo 1
Introducción

El presente capítulo contextualiza el trabajo de investigación realizado en esta tesis. En primer lugar, se discute las particularidades de la monitorización de la calidad de servicios en la nube. A continuación, se plantea el problema a resolver, se establecen los objetivos y, por último, se describe la metodología de investigación empleada.

La sección 1.1 introduce la computación en la nube y sus características esenciales.

La sección 1.2 plantea la necesidad de contar con técnicas de monitorización de la calidad de servicios y herramientas de soporte específicas para la nube. Esta sección también aborda los problemas existentes y plantea un problema latente que será el que se solventa mediante este trabajo.

La sección 1.3 introduce la Ingeniería de Software Dirigida por Modelos y los modelos en tiempo de ejecución como tecnología para abordar la solución a los problemas identificados.

La sección 1.4 describe el problema a ser resuelto a lo largo de este trabajo de investigación.

La sección 1.5 describe los objetivos del método de monitorización de servicios cloud y la infraestructura que lo soporta, los mismos que forman parte del trabajo a desarrollar, así como la hipótesis que se contrastará.

La sección 1.6 describe el contexto en el que se ha desarrollado el trabajo de investigación que constituye esta tesis doctoral.

La sección 1.7 describe la metodología de investigación utilizada. Finalmente, la sección 1.8 describe la estructura de este documento.

1.1. La computación en la nube

La computación en nube representa un cambio fundamental en la forma en la que los servicios de tecnología de la información (IT) son provistos: a través de una red (usualmente Internet) como utilidades que pueden ser rápidamente aprovisionadas y liberadas bajo demanda.

Este modelo ha atraído la atención de muchas organizaciones en los últimos años debido a las oportunidades que ofrece (reducción de costes, facilidad de uso y conveniencia, elasticidad de recursos, etc.). En mayo de 2008 ya se estimaron las ventajas de costes que provee la computación en la nube, las que van de tres a cinco veces para aplicaciones de organizaciones y más de cinco veces para aplicaciones del cliente final (Lynch, 2008). Los costes se reducen debido a que los clientes no tienen que hacer frente a la infraestructura usada. Debido al modelo de pago por uso, el coste es variable e inferior al incurrido con el uso de tecnologías tradicionales. También se observan la reducción de costes asociados a la compra de nuevas herramientas y renovación de licencias y la reducción de costes de personal.

En un principio, la computación en la nube ha sido más adoptada en empresas grandes pero también se ha extendido a pequeñas y medianas empresas (PYMES) debido a la mínima inversión en infraestructura y al reducido riesgo de experimentar con esta tecnología. Gupta *et al.* (2011) presenta una comparación de varios estudios empíricos sobre la adopción y uso de la computación en la nube por parte de las PYMES. El estudio revela un creciente interés por parte de estas empresas ya que perciben el cloud como una alternativa para acceder a servicios tecnológicos que permitan optimizar su negocio y lograr una mejora competitiva en el mercado.

La actitud positiva hacia la importancia e influencia de la computación en la nube resultó en previsiones optimistas relacionadas a este mercado. Algunas organizaciones reportaron que la ventaja de costes asociada con el modelo de la computación en la nube hace que ésta sea más atractiva en términos de retornos de inversión económica (Hugos *et al.*, 2011) .

El término de computación en la nube ha sido definido de muchas maneras por la academia, la industria, las organizaciones de tecnología y organizaciones de estandarización.

Según el *National Institute of Standards and Technology* - NIST (Mell *et al.*, 2011) la computación en la nube es:

"Un modelo para habilitar el acceso ubicuo, bajo demanda por red a un conjunto compartido de recursos computacionales configurables (p. ej., redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios.

Por otro lado, Gartner (Van der Meulen *et al.*, 2008) define la computación en la nube como un estilo de computación en el cual capacidades escalables que utilizan tecnologías de Internet, son provistas masivamente "como un servicio"

a múltiples clientes externos. Gens, (2008) define a la computación en la nube como un modelo de desarrollo emergente de IT, despliegue y entrega de productos, servicios y soluciones sobre Internet.

En el ámbito académico, una de las definiciones más aceptadas es la proporcionada por la Universidad de Berkeley (Armbrust *et al.*, 2009) que considera la computación en la nube tanto a las aplicaciones provistas como un servicio en Internet como al hardware y sistemas de software en centros de datos que proveen esos servicios. Los servicios por sí mismos han sido referidos como *Software as a Service (SaaS)*. El hardware del centro de datos y el software es lo que llamaremos una nube. Cuando una nube está disponible a través de pago por uso al público en general, entonces a ésta se la llama *red pública*. Se usa el término *red privada* para referirse a los centros de datos internos de una organización, que no se hacen disponibles para el público en general.

La computación en la nube presenta características propias y esenciales, que hacen de éste, un modelo de distribución atractivo para organizaciones, las cuales buscan el aprovisionamiento de software y hardware con reducción de inversión inicial, mínimo esfuerzo de mantenimiento, capacidad de crecimiento según sus necesidades y acceso universal desde cualquier ubicación geográfica.

Teniendo en cuenta estos requisitos, el NIST (Mell *et al.*, 2011) ha incorporado a la definición de computación en la nube, cinco características esenciales tales como: (i) el auto-servicio bajo-demanda, (ii) un acceso universal a la red, (iii) la puesta común de recursos que permite el uso del modelo multitenencia en el que los recursos computacionales del proveedor puede servir a múltiples consumidores, (iv) características de elasticidad para proveer servicios de manera elástica y automáticamente escalar hacia arriba o hacia abajo los servicios provistos, dando la idea de una cantidad ilimitada de recursos y (v) servicio medido, con capacidad de medición para controlar automáticamente el aprovisionamiento, almacenamiento, procesamiento, etc.

De igual manera, la computación en la nube, permite acceder a tres modelos de servicio que proveen diferentes niveles de acceso a los recursos que ésta ofrece. Estos modelos de servicio son: (i) Software como Servicio (SaaS), (ii) Plataforma como Servicio (PaaS) e (iii) Infraestructura como Servicio (IaaS). Cada uno de ellos se enfoca a ofrecer servicios ya sean de software, así como también de plataforma e infraestructura, respectivamente, permitiendo al usuario disfrutar de las ventajas antes señaladas. Al ser este modelo de distribución de servicios una nueva tendencia, muchas organizaciones están actualmente adoptándolo. En este trabajo, nos centraremos en el modelo de servicio SaaS.

1.2. Calidad de servicios en la nube

Las características de la computación en la nube, hacen de ella un modelo conveniente de distribución de hardware, plataforma y software, el cual ofrece muchas ventajas a sus clientes, quienes esperan que las aplicaciones y servicios en la nube sean entregados oportunamente, con altos niveles de calidad de servicio, confiabilidad, disponibilidad y latencia; comparables e incluso mejores que cuando mantenían todos los recursos en plataformas tradicionales con configuraciones de hardware nativos (Bauer *et al.*, 2013).

Las infraestructuras de computación en la nube introducen nuevos riesgos debido a temas como computación virtualizada, memoria, almacenamiento y recursos de red que un proveedor de IaaS provee y en donde son desplegadas las instancias de las aplicaciones (Bauer *et al.*, 2013). Además, los clientes buscan servicios en la nube apropiados y que cumplan con sus expectativas de calidad de servicio. Sin embargo, si estas expectativas no son alcanzadas, los clientes dejarán de usar estos servicios (Varela *et al.*, 2012).

Existe una alta competencia de proveedores de servicios en la nube, basada en los precios a los que ofertan los recursos. Sin embargo, los proveedores deben considerar otro diferenciador más allá del precio de oferta: la calidad de sus servicios. Por ejemplo, si los niveles de rendimiento no cumplen las expectativas o se vuelven impredecibles, los clientes rechazarán el servicio o evitarán su adopción. Por otro lado, si se reúnen o se exceden las expectativas, la reputación de un proveedor en la nube incrementará favorablemente y por tanto sus servicios tendrán una mayor acogida y por ende utilización (Varela *et al.*, 2012). De ahí que, los proveedores de servicios deberán realizar fuertes inversiones económicas para mantenerse en el negocio, ya que cualquier mejora en la calidad de servicio será percibida y valorada por el cliente final.

Además, existe un interés tanto de parte de la industria como de la academia, en abordar los retos asociados a la calidad de los servicios en la nube. De ahí, la habilidad para especificar y mantener la calidad de servicio es un importante tema a ser tratado por clientes y proveedores de servicio (Abdelmaboud, *et al.,* 2015).

La especificación de la calidad de un servicio se establece entre clientes y proveedores a través de los Acuerdos de Nivel de Servicio (*Service Level Agreenment*, SLA). Los SLA son contratos que definen las mínimas garantías que un proveedor de servicios ofrece a sus clientes (Baset, 2012). Además, existen penalizaciones en caso de que la calidad, expresada a través de los requisitos no funcionales (RNF), no sea provista como se acordó en el SLA. Debido a la necesidad de conocer el nivel de cumplimiento de los SLA, así como también el

nivel de calidad del servicio ofrecido por los proveedores, se hace imprescindible contar con métodos y herramientas que permitan conocer el estado actual de provisión de los servicios. Soluciones de este tipo apoyarán tanto al proveedor, quien conocerá el estado de sus servicios y el estado de cumplimiento de los SLA, lo que aportará a la mejora de los atributos de calidad de los servicios ofrecidos; así como también será de utilidad a los clientes quienes podrán tomar decisiones con respecto a la adopción de los servicios que consumirán y así también a solicitar compensación en caso de incumplimientos en los SLA.

En consecuencia, y dada la importancia de una gestión y monitorización adecuada de los SLA, tanto la academia como la industria han puesto sus empeños en este tema, según un informe presentado por Kyriazis (2013), con los resultados de investigaciones realizadas en la Unión Europea. El informe menciona que una de las principales preocupaciones de los usuarios se refiere a la validación y a la supervisión de la calidad de los servicios provistos, en donde los usuarios requieren niveles mayores de transparencia, a través de información de monitorización de SLA de manera precisa y oportuna.

Además, este informe enfatiza la necesidad de herramientas de soporte y mecanismos para apoyar a las distintas fases del ciclo de vida del SLA, entre ellas, la ejecución de la monitorización del servicio, que se analiza en este informe, en base a los resultados de varios proyectos de investigación, orientados a proporcionar una solución en cuanto al manejo y monitorización del SLA como mecanismo para determinar la calidad en la provisión de los servicios despegados en la nube.

Por otra parte, en el trabajo de Muller *et al.* (2014) se habla de la dificultad en la detección de violaciones en el SLA y de la manera en la que esta labor ha sido puesta en manos del cliente, mostrando la necesidad de tener técnicas para supervisar el cumplimiento de los SLA.

Existen varias propuestas que también abordan la necesidad de la detección de las violaciones del SLA, unas centradas en la fase de ejecución del servicio, una vez que el servicio se ha desplegado en la nube (Comuzzi *et al.,* 2003; Keller *et al.*, 2002; Michlmayr *et al.*, 2009), otras en tiempo de pruebas (Di Penta *et al.*, 2007; Palacios *et al.*, 2010), y otras que se centran en proveer una explicación de violaciones del SLA (Mahbub *et al.*, 2007; Spanoudakis *et al.,* 2006).

Sin embargo, algunas de estas soluciones son para servicios web y su uso en los servicios cloud no ha sido probado todavía. Varias soluciones soportan SLAs definidos de manera ad-doc, sin una conceptualización de los elementos de un SLA y la relación entre ellos. Otras soluciones son poco flexibles ya que acoplan

la configuración de la monitorización con una especificación de SLA determinada. Otras soluciones se centran en monitorizar atributos de calidad de bajo nivel (uso de CPU y memoria, etc.). Otras soluciones no cuentan con una arquitectura bien definida lo que dificulta la extensión de las soluciones de monitorización.

1.3. Técnicas de monitorización

Dada la necesidad de conocer la calidad de los servicios provistos, se hace imprescindible contar con métodos y herramientas que permitan llevar a cabo actividades de monitorización de servicios en la nube con el fin de tomar acciones preventivas o correctivas ante fallos o verificar el cumplimiento del SLA. La monitorización es un proceso que permite identificar de manera completa y precisa la causa de un problema, a través de la captura de información para así determinar el estado de un sistema o servicio y mostrarlo utilizando mecanismos de visualización adecuados (p. ej., dashboards).

Entre los métodos y herramientas de monitorización, existen herramientas de propósito general y otras específicas para la monitorización de servicios en la nube. Tanto aquellas de propósito general, que sirven para monitorizar ciertos aspectos de calidad comunes, como aquellas específicas para servicios en la nube, se puede observar la necesidad de que cuenten con características de calidad deseables, de las cuales Fatema *et al.* (2014) identifican: la escalabilidad, portabilidad, no-intrusión, robustez, posibilidad de permitir multitenencia, interoperabilidad, personalización, extensibilidad, monitorización de recursos compartidos, usabilidad, accesibilidad y facilidad de almacenamiento; características que se describirán con mayor detalle en el Capítulo 2 de este trabajo, y a las que se hace necesario añadir la flexibilidad para permitir añadir, modificar y eliminar nuevos requisitos de monitorización.

Mantener todas estas características de calidad es un desafío para los creadores de soluciones de monitorización, quienes día a día realizan mejoras a sus métodos y herramientas, así como también para proveedores cloud quienes han ido paulatinamente agregando herramientas de monitorización a sus plataformas.

Actualmente, algunos proveedores de servicios en la nube, utilizan herramientas de monitorización específicas para sus plataformas. Por ejemplo, Amazon Cloud Watch (Amazon Cloud Services, 2016) que monitoriza los servicios provistos por Amazon Web Services (AWS); Azure Watch (Watch, 2016) que monitoriza las instancias de Windows Azure, las bases de datos, sitios web, aplicaciones web, etc. Sin embargo, dichas herramientas están sujetas a

características de calidad fijas y están además estrechamente ligadas a la plataforma, lo que limita su uso a otras plataformas.

En contraste, existen también herramientas de monitorización tales como Nimsoft (2011) que está disponible para monitorizar servicios de varias plataformas tales como: Amazon EC2, S3 Web Services, Rackspace Cloud, Microsoft Azure, Google App Engine, Google Apps y Salesforce CRM; otra herramienta que permite monitorizar servicios de varias plataformas es Monitis (2014) que puede ser usada para Amazon EC2/AWS y Rackspace Cloud.

Sin embargo, la mayoría de estas herramientas son limitadas a la monitorización de ciertos requisitos no funcionales o no son fácilmente configurables ya que están muy acopladas a la infraestructura de monitorización. En consecuencia, las soluciones existentes presentan ciertas limitaciones de interoperabilidad, flexibilidad o adaptación a múltiples plataformas.

También existen soluciones de monitorización de propósito general. Según (Fatema *et al.*, 2014) estas soluciones presentan limitaciones en cuanto a escalabilidad (p. ej. Nagios, 2007), robustez (p. ej. Bugtracking system, 2015; Nagios, 2007), características multitenencia (p. ej. Nagios, 2007; Cacti, 2012), falta de habilidad para monitorizar el consumo de recursos por servicio (p. ej. Kiwi, 2013), monitorización de calidad de servicio (p. ej. Bugtracking system, 2015; Kiwi, 2013).

Por otra parte, muchas herramientas de propósito general anteriores a la computación en la nube, continúan disponibles y podrían ser adoptadas para monitorizar servicios en varios niveles de abstracción. Esas herramientas en su mayoría utilizan un modelo cliente/servidor que permite la instalación de agentes en cada sistema a ser monitorizado (Fatema *et al.*, 2014). Sin embargo, estas herramientas no están preparadas para ambientes altamente heterogéneos. Existen algunas soluciones específicas para la nube que solucionan estos problemas (Amazon Cloud Watch, 2011; MODAClouds, 2015; Monitis, 2014; Nimsoft, 2011; entre otras), aunque éstas también presentan ciertas limitaciones en cuanto a sus prestaciones (Fatema *et al.*, 2014).

Como las herramientas de propósito general, muchas herramientas específicas para la nube, utilizan agentes, los cuales envían alertas en caso de necesitarse atención, para lo cual se debe permitir la instalación y ejecución de agentes SaaS. Muchas herramientas permiten también la monitorización a diferentes niveles ya sea SaaS, PaaS o IaaS; de hecho, existen herramientas que se especializan únicamente en ciertos niveles de acceso a los recursos (Fatema *et al.*, 2014).

Finalmente, se puede concluir que, a pesar de los esfuerzos tanto de la academia como de la industria, existen aún problemas que no han sido abordados en profundidad y que hacen que los métodos y herramientas de monitorización existentes no reúnan ciertas características necesarias y deseables para este tipo de soluciones (p. ej., flexibilidad, interoperabilidad, escalabilidad). De ahí la necesidad de cubrir dichos aspectos con una solución que proporcione:

- Flexibilidad en la especificación de requisitos no funcionales, para que estos requisitos se puedan añadir y/o cambiar fácilmente en tiempo de ejecución.
- Soporte a la monitorización de características de calidad de más alto nivel (fiabilidad, seguridad, rendimiento, etc.)
- Interoperabilidad y escalabilidad de la infraestructura de monitorización mediante su comunicación con otras soluciones de monitorización.
- Extensibilidad de la infraestructura de monitorización para que pueda adaptarse adecuadamente en respuesta a nuevos cambios.

1.4. Planteamiento del problema

Con la llegada de la computación en la nube y su adopción en muchas organizaciones (Giannakouris *et al.*, 2014), se ha incrementado la cantidad de proveedores de recursos ya sea de infraestructura, plataforma o software en la nube, lo que ha hecho que los clientes de este tipo de servicios, deban seleccionar cuidadosamente al proveedor que cumpla sus expectativas y requisitos tanto funcionales como no funcionales.

Para fijar las cláusulas de aprovisionamiento de servicios entre clientes y proveedores se han establecido acuerdos de nivel de servicio que contienen información relacionada a los requisitos funcionales y no funcionales ofrecidos por el proveedor de servicios y las penalizaciones en el caso de su incumplimiento.

De esta manera, se hace imprescindible contar con herramientas que permitan la evaluación de la calidad del servicio durante su provisión, por una parte para ofrecer tanto al cliente como al proveedor información sobre el cumplimiento de los SLA y la calidad del servicio, y por otra parte para ayudar al proveedor a tomar acciones preventivas y correctivas en cuanto al funcionamiento de sus servicios. Por último, también es importante poder brindar al cliente una visión de cómo los servicios están siendo aprovisionados para que de esa manera ellos tengan una idea clara del valor que los servicios contratados aportan a su organización.

Así, los requisitos no funcionales (RNF), los mismos que constituyen la calidad acordada entre el cliente y el proveedor, cuyos términos son incluidos en el SLA, deben ser cumplidos a cabalidad con el fin de garantizar la satisfacción del cliente.

Sin embargo, estos RNF, pueden cambiar a lo largo de la provisión del servicio, ya sea debido a renegociaciones de los SLA entre cliente y proveedor, por cambios en las ofertas de los proveedores o también por la necesidad de evaluar diferentes atributos de calidad (p. ej. disponibilidad, elasticidad, seguridad, rendimiento); de ahí es muy importante contar con sistemas de monitorización de servicios flexibles, que sean fácilmente adaptables a esos cambios de requisitos y que a su vez estos cambios no tengan un impacto importante que implique modificaciones sustanciales en la implementación de la infraestructura de monitorización.

Además, la computación en la nube y específicamente el software como un servicio (SaaS), trae nuevos desafíos en cuanto a la monitorización de sus recursos, esto debido a las características propias de los servicios de software en la nube: elasticidad, escalabilidad, concurrencia, tiempo de respuesta, pago por uso, entre otros, que hace que sea necesario contar con sistemas de monitorización que sean específicos para este tipo de software (Fatema *et al.*, 2014).

Otro de los problemas existentes actualmente, es que los SLA son difíciles de ser automatizados, ya que éstos son muchas veces especificados inclusive en lenguaje natural, lo que ha complicado mucho su tratamiento automático y verificación; para ello, se han creado algunos lenguajes de especificación de SLA como es el caso del *Web Service Level Agreement (WSLA)* propuesto por Ludwig *et al. (*2003) o el *Web Service Agreement* (Andrieux et al., 2004), entre otros. Estos lenguajes tratan de especificar bajo un estándar a los SLA, de tal manera que puedan ser utilizados, por ejemplo, en el caso de la monitorización. Sin embargo, muchas veces, los sistemas de monitorización son programados de forma estática y abarcan cierto número de requisitos no funcionales que no pueden ser modificados fácilmente, ya que no describen cláusulas del SLA sino más bien ofrecen la monitorización de ciertas propiedades de bajo nivel de los servicios en ejecución.

Además, como se ha discutido anteriormente, muchos sistemas de monitorización, como es el caso de Azure Watch ("*Windows Azure Diagnostic Monitoring and Autoscaling*," 2016) y Amazon Cloud Watch (Amazon Web Services, 2016), son dependientes de la plataforma y están estrechamente vinculados a ella, razón por la que si se obtienen recursos de diferentes

plataformas, se deberían usar sistemas de monitorización diferentes por la falta de interoperabilidad entre cada una de ellas, lo que conlleva un esfuerzo extra. Ese es el caso también de herramientas que permiten monitorizar puntualmente ciertos atributos de calidad y se especializan en ello, lo que hace difícil su adopción.

Todos estos problemas, son latentes cuando no existe una herramienta de monitorización, que permita capturar información de varias fuentes de datos y las integre para poder, en base a esta información, presentar informes a los distintos stakeholders.

Teniendo en cuenta todo lo anteriormente citado, surge la necesidad de un método de monitorización con una infraestructura de soporte cuyo diseño sea independiente de plataforma, que permita una monitorización flexible, fácilmente integrable a distintas plataformas cloud y que además permita la interoperabilidad con otras herramientas de monitorización, de tal manera que los datos emanados por terceros en herramientas de monitorización especializadas, puedan ser tratados de forma homogénea. De esta manera, se podrá disponer de una solución de monitorización de servicios cloud lo suficientemente flexible, abierta, y a un alto nivel de abstracción que se ajuste a las cambiantes necesidades de clientes y proveedores.

1.5. Solución Propuesta: Ingeniería dirigida por modelos y modelos en tiempo de ejecución

Para abordar las limitaciones de las aproximaciones de monitorización existentes, creemos oportuno crear un método y una infraestructura que soporte las características antes mencionadas, utilizando técnicas que permitan en tiempo de ejecución, añadir requisitos no funcionales a monitorizar y cambiar las configuraciones que sean necesarias, para adaptar el sistema a nuevas necesidades de monitorización y que, de ser necesario, puedan integrar datos provenientes de diferentes soluciones, logrando una infraestructura de monitorización con un alto grado de flexibilidad e interoperabilidad.

La hipótesis de esta tesis es que la utilización de técnicas de Ingeniería de Software Dirigida por Modelos (*Model Driven Engineering*, MDE) permite abordar los problemas identificados ya que permiten especificar los requisitos de calidad a ser monitorizados y las características de calidad y métricas para medirlos en un alto nivel de abstracción, así como también establecer fácilmente las fuentes de datos obtenidos desde los servicios y otros parámetros de monitorización. Todo ello no solo en tiempo de diseño sino que también en tiempo de ejecución, mediante el uso de *modelos en tiempo de ejecución (models at runtime)* (Blair *et al.,* 2009).

Esta propuesta plantea que, dado que los modelos son en realidad el propio sistema, podemos tenerlos en tiempo de ejecución, y en caso necesario, modificarlos.

Tradicionalmente, la investigación en MDE ha estado principalmente enfocada al uso de modelos en tiempo diseño. La definición de un modelo puede variar dependiendo de su propósito de uso. En este caso, se ha tomado la definición de (Stachowiak, 1973):

"Un modelo se caracteriza por los siguientes elementos: (i) una parte objetiva a la cual se refiere el modelo, (ii) un propósito que define el objetivo del modelo y (iii) una función de abstracción que mapea únicamente las características que tienen un propósito y son relevantes del dominio original".

Usualmente, un modelo se define en tiempo de diseño, pero en los últimos años se han investigado técnicas que permiten utilizar los modelos en tiempo de ejecución, para así permitir que los sistemas sean más dinámicos y puedan cambiar su comportamiento de acuerdo a circunstancias que se presentan en tiempo real y a lo largo de su ejecución.

Teniendo en cuenta esta necesidad, Baresi *et al. (*2010) mencionan que el futuro de la Ingeniería del Software demandará que se enfoque en proveer soporte inteligente para el software en tiempo de ejecución, rompiendo la barrera rígida existente entre el tiempo de desarrollo y el de ejecución; así también mencionan que los modelos necesitan continuar "viviendo" en tiempo de ejecución y evolucionar según el software o las necesidades lo requieran en tiempo de ejecución.

El uso de los modelos de calidad en tiempo de ejecución permitirá desacoplar la especificación de los requisitos de monitorización de la infraestructura responsable por la medición y análisis en tiempo de ejecución de los niveles de calidad de servicio y detección de inconsistencias del SLA.

Por lo tanto, los modelos en tiempo de ejecución representan una técnica apropiada para la implementación de una solución de monitorización que permita la adición, cambio o eliminación de requisitos no funcionales en tiempo de ejecución, sin necesidad de que la infraestructura sea detenida para reconfigurarse. Así también puede permitir la modificación de distintos parámetros de monitorización (p. ej. frecuencia de captura de información) y la especificación de nuevas métricas y fuentes de datos en tiempo de ejecución, evitando cambios en la infraestructura de monitorización que impliquen programación y conocimientos avanzados sobre su implementación.

De esta forma, los cambios podrán ser realizados en el modelo, el mismo que podrá ser consumido una infraestructura de monitorización que actualizará sus mecanismos de monitorización a los nuevos requisitos, aportando un alto grado de flexibilidad a la solución.

Finalmente, es importante resaltar que los modelos en tiempo de ejecución han sido ya utilizados para la monitorización de varios tipos de características de sistemas pertenecientes a diversos dominios (Garlan *et al.*, 2004; Garlan *et al.*, 2001; Garlan *et al.*, 2002), así como también para la monitorización de características no funcionales (Ardagna *et al.*, 2010) e implementaciones de middleware con diversos objetivos (Costa *et al.,* 2006). Sin embargo, hasta donde se conoce, no se han reportado soluciones que utilicen modelos en tiempo de ejecución para la monitorización de servicios cloud y que tengan como propósito la especificación de los RNF, sus métricas y otras configuraciones para proporcionar flexibilidad y que actúen como parte de un middleware que interactúe de manera directa con los servicios en la nube para recopilar la información necesaria y mostrarla a la parte interesada, ya sea ésta el cliente o el proveedor de los servicios.

1.6. Hipótesis y objetivos

La hipótesis de partida de esta tesis doctoral es que el uso de técnicas provenientes de la Ingeniería Dirigida por Modelos (MDE), y en particular, los modelos en tiempo de ejecución constituyen una tecnología muy apropiada para soportar la monitorización de servicios en la nube. Esta hipótesis general se basa en las siguientes hipótesis específicas:

H1. La aplicación de un método de monitorización basado en modelos en tiempo de ejecución permite evaluar la calidad de los servicios de software en la nube, realizar la comprobación del cumplimiento del SLA, y además, monitorizar características adicionales que contribuyen positivamente a una correcta provisión de los servicios cloud y a la actuación de manera preventiva o correctiva por parte de los proveedores.

H2. La definición de una infraestructura de monitorización independiente de plataforma facilita la extensibilidad e implementación de la infraestructura de monitorización en cualquier plataforma cloud.

H3. La utilización de modelos que estén acordes a estándares (p. ej., ISO/IEC 25010) y lenguajes de dominio específicos (p. ej., WSLA)

permiten especificar tanto los requisitos de monitorización como las características y atributos de calidad de una manera estructurada y guiada.

H4. La adopción de modelos en tiempo de ejecución proporciona flexibilidad para la realización de cambios en los requisitos de monitorización. Estos cambios modifican el modelo en tiempo de ejecución, que contiene todos los parámetros de monitorización, y por tanto, no será necesario realizar ningún cambio en la implementación de la infraestructura de monitorización.

H5. El método e infraestructura de monitorización puede ser fácilmente utilizado y es de utilidad para los consumidores y proveedores de servicio.

Para probar las hipótesis anteriores, se define el siguiente objetivo general para esta tesis doctoral: *la definición y evaluación empírica de un método para la monitorización de servicios desplegados en la nube, bajo el modelo SaaS, haciendo uso de modelos en tiempo de ejecución.*

Dicho objetivo general se descompone en los siguientes objetivos específicos:

1) Definir un método de monitorización basado en modelos en tiempo de ejecución que esté soportado por un proceso sistemático y repetible. El proceso describirá las actividades del método, así como también las tareas necesarias para llevar a cabo las actividades. Este objetivo se corresponde con la hipótesis H1.

2) Definir una infraestructura de monitorización cuyo diseño sea extensible, independiente de la plataforma cloud, y que permita realizar la monitorización de servicios en la nube y la detección de incumplimientos del SLA. La infraestructura deberá hacer uso de modelos en tiempo de ejecución para proporcionar la flexibilidad e interoperabilidad deseables, permitir la especificación de los requisitos no funcionales a partir de acuerdos de nivel de servicios (SLA) y estar alineada con estándares de calidad. Este objetivo se corresponde con las hipótesis H2 y H3.

3) Instanciar e implementar la infraestructura de monitorización en una plataforma cloud específica para realizar monitorizaciones de servicios desplegados en la nube y observar la aplicabilidad y facilidad de adaptación de la infraestructura propuesta. La infraestructura estará compuesta por un configurador que permitirá a los stakeholders configurar los requisitos de monitorización y generar el modelo en

tiempo de ejecución y un *middleware* de monitorización que utilizará este modelo para realizar evaluaciones de calidad y detectar incumplimientos del SLA. Este objetivo se corresponde con las hipótesis H2 y H4.

4) Evaluar empíricamente el método de monitorización propuesto mediante experimentos controlados. Este método empírico es idóneo para proporcionar evidencias sobre la facilidad de uso y utilidad de métodos y tecnologías desarrolladas en el ámbito de la Ingeniería del Software. Esta evaluación proveerá una retroalimentación para la mejora de la propuesta. Este objetivo se corresponde con las hipótesis H4 y H5.

1.7. Contexto de investigación

Esta tesis doctoral se ha desarrollado en el contexto del grupo de investigación de Ingeniería del Software y Sistemas de Información (ISSI) del Departamento de Sistemas Informáticos y Computación de la Universitat Politècnica de València (UPV).

Los trabajos que han hecho posible el desarrollo de esta tesis, se engloban en proyectos de I+D financiados con fondos públicos. En particular, esta tesis ha contribuido a los siguientes proyectos y ayudas:

- Proyecto Value@Cloud: *"Desarrollo Incremental de Servicios Cloud Dirigido por Modelos y Orientado al Valor del Cliente"* de la convocatoria de ayudas a proyectos de I+D+i, orientada a los retos de la sociedad del año 2013 financiado por el Ministerio de Economía y Competitividad (España), Participantes: Universitat Politècnica de València y Universidade Nova de Lisboa. IP: Silvia Abrahão, De Enero de 2013 a Diciembre de 2017.
- Proyecto TwinTIDE: "*Towards the Integration of Transectorial IT Design and Evaluation"*. Financiado por la acción COST de la Unión Europea IC0904. IP: Effie Law (University of Leicester, UK y ETH Zurich) y Silvia Abrahão (Responsable Nodo UPV). De Noviembre de 2009 a Noviembre de 2013.
- Proyecto *"Model-Driven Incremental Development of Cloud Services"*, Microsoft Azure Research Award, Microsoft Research. IP: Emilio Insfran. Julio de 2014 a Julio de 2016.

Por otro lado, esta tesis también ha sido realizada gracias al apoyo del programa de becas de la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) de Ecuador y a la Universidad de Cuenca-Ecuador.

1.8. Método de investigación

La investigación en Ingeniería difiere de las Ciencias en lo que se refiere al objeto de estudio como en método. Mientras que las Ciencias se ocupan del estudio de objetos y fenómenos existentes la Ingeniería basa sus estudios en cómo crear nuevos objetos (Marcos 2005). Por lo tanto, la metodología que se propone utilizar a lo largo de esta tesis doctoral, está estructurada siguiendo una extensión del modelo para la transferencia de tecnología propuesto por Gorschek *et al. (*2006) y está basada en las necesidades de la industria. Ésta incluye actividades de evaluación y observación.

Este modelo de investigación y transferencia de tecnología, se basa en ocho actividades relacionadas, donde la búsqueda de soluciones adecuadas se realiza en un proceso iterativo, por medio de la formulación de soluciones candidatas y la correspondiente validación empírica que permite dirigir los esfuerzos hacia una solución realista.

En esta tesis doctoral se han cubierto las 6 primeras actividades de esta metodología, llegando hasta la validación inicial de las soluciones planteadas en la misma, aunque la validación realista y la liberación de la solución hacia la industria se planean abordar como trabajo futuro. Como preparación para explorar trabajos conjuntos con empresas, se han explorado casos de estudio más complejos, como el planteado por el proyecto europeo SLA@SOI denominado Open Reference Case (ORC) (McCarthy, 2011), el cual es planteado como una solución para soportar el proceso de ventas de supermercados. Se han realizado también pruebas de la solución en un caso de estudio real que tiene que ver con el dominio de las subastas en línea. Mostrando así la aplicabilidad del método.

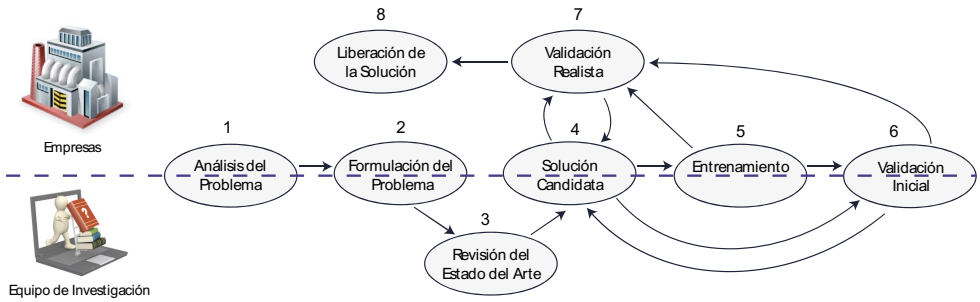


Figura 1-1. Modelo de transferencia tecnológica

1. **Análisis del problema**: Se busca entender el problema que los socios industriales quieren resolver. Este paso se ha llevado a cabo a través de

reuniones informales donde participaron un grupo de investigadores y algunos expertos de las empresas que participan como EPO (Entidades Promotoras Observadora) en el proyecto Value@Cloud. En estas reuniones se han hecho presentaciones y se han discutido los retos, necesidades y problemas específicos que los expertos perciben.

2. **Formulación del problema:** Una vez que el problema está identificado, éste es formulado de manera más precisa y los factores contextuales y las asunciones de trabajo, son especificadas claramente.

3. **Revisión del estado del arte:** Se realiza una revisión crítica de la literatura así como de las soluciones tecnológicas comerciales y de código abierto disponibles para identificar hasta qué punto las metas han sido abordadas y cuáles son los problemas abiertos a resolver con la investigación a desarrollar.

4. **Solución candidata:** Se idean una o más soluciones potenciales. Dichas soluciones serán más tarde depuradas a través de las distintas actividades de refinamiento (actividades 6 y 7).

5. **Entrenamiento:** Se trata de una etapa incremental. En las primeras fases, el entrenamiento se enfoca a que el equipo de investigación proporcione el conocimiento base necesario para que los profesionales del área puedan formarse una opinión de la aplicabilidad de la propuesta. En las fases tardías, el entrenamiento sirve para crear guías y pasos metodológicos detallados para que los profesionales sean capaces de aplicar las soluciones propuestas.

6. **Validación inicial:** Se lleva a cabo una evaluación preliminar de las soluciones, en un entorno de laboratorio. En este caso se llevarán a cabo estudios empíricos con alumnos y profesionales.

7. **Validación realista:** Se llevan a cabo estudios de caso y/o experimentos controlados en entornos industriales, empezando por estudios piloto para después extenderse en usos más amplios. En esta fase se definirán guías prácticas y se desarrollarán prototipos con las herramientas que soporten el método de monitorización de servicios en la nube.

8. **Liberación de la solución:** En este último paso se valoran los resultados obtenidos y las herramientas y el material de entrenamiento son preparados para su uso.

Experimentos

La experimentación es una fase crucial de la evaluación empírica y puede ayudar a determinar si los métodos, herramientas u otras tecnologías utilizados en la monitorización de la calidad de los servicios cloud se ajustan a una teoría particular. La experimentación es apropiada para investigar diferentes aspectos, como confirmar o probar teorías existentes, la evaluación de la precisión de modelos, validación de medidas, etc.

Aunque existen varios métodos empíricos, en esta tesis, seleccionamos los experimentos como método de investigación para proporcionar evidencia sobre la utilidad del método de monitorización propuesto. Un experimento es una investigación empírica que manipula una variable (denominada independiente) del entorno o fenómeno estudiado con el objetivo de medir el efecto que tiene sobre otra variable (denominada dependiente). Existen dos tipos de experimentos: controlados y cuasi. En los experimentos controlados los tratamientos se asignan a los participantes de manera aleatoria mientras que en los cuasi-experimentos esta aleatorización no es posible.

Los diseños que carecen de un control experimental de todas las variables relevantes debido a la falta de aleatorización ya sea en la selección aleatoria de los participantes o en la asignación de los mismos a los grupos experimental y control, y que no necesariamente poseen dos grupos (el experimental y el control), son conocidos como cuasi-experimentos.

En esta tesis, hemos seleccionado los cuasi-experimentos como método de investigación debido a la falta de un grupo de control, ya que no existe actualmente un método o infraestructura de monitorización para servicios cloud que pueda ser considerada estándar o ampliamente aceptada en la industria. Además, como se ha mencionado antes, este trabajo cubre las 6 primeras actividades del método de transferencia tecnológica de Gorschek *et al. (*2006), llegando hasta la validación inicial del método de monitorización propuesto. Esta validación inicial consistirá en realizar una *familia de cuasi- experimentos* con estudiantes y profesionales para evaluar su percepción en cuando a la utilidad del método de monitorización propuesto.

Los cuasi-experimentos que aparecen en esta tesis han sido diseñados siguiendo los pasos del proceso experimental en Ingeniería del Software propuesto por Wohlin *et al. (*2012). La Figura 1-2 muestra las principales actividades involucradas en el proceso experimental así como los artefactos generados en cada una de las actividades.

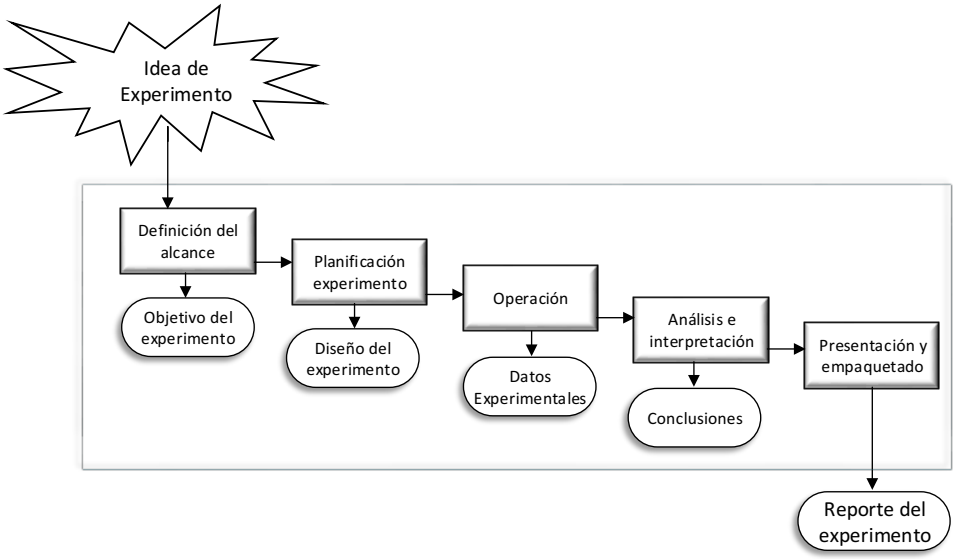


Figura 1-2 Visión global del proceso experimental de Wohlin *et al.* (2012)

El punto de partida de un experimento (ya sea controlado o cuasi) es la idea clara de lo que el experimento va a evaluar. El proceso comienza con la definición del alcance, donde se define el alcance, la meta y los objetivos del experimento en términos del problema. La meta del experimento es formulada a partir del problema a resolver siguiendo el marco de trabajo *Goal –Question-Metric (GQM)* propuesto por Basili *et al. (*1994), el cual sigue el esquema presentado en la Tabla 1-1.

Tabla 1-1. Plantilla para la definición de Goal-Question-Metric

Analizar	<Objeto(s) de Estudio> - ¿Qué es lo que se analiza?
Con el propósito de	<Propósito> - ¿Qué intención tiene el estudio?
Con respecto a	<Enfoque de Calidad> - ¿Cuál es el efecto estudiado?
Desde el punto de vista de	<Perspectiva> - ¿Quién se ve afectado?
En el contexto de	<Contexto> - ¿Dónde tiene lugar el estudio, sobre qué artefactos y con qué tipo de participantes?

La siguiente actividad es la *planificación del experimento*, donde se define el diseño del experimento. En ésta, se define en profundidad el contexto, que incluye el perfil de los sujetos, el entorno en el que tendrá lugar el experimento y en nuestro caso particular qué es lo que se va a monitorizar. También se especifican las hipótesis de una manera formal, esto incluye tanto las hipótesis nulas como las alternativas y las variables dependientes e independientes. Además se seleccionará un diseño experimental y la instrumentación. En esta

fase es además, un tema fundamental la evaluación de la validez, es decir cómo de válidos son los resultados encontrados. La evaluación de la validez no se debe dejar para el final y que desde el inicio es importante analizar las posibles amenazas que se pueden producir e intentar mitigarlas en la medida de lo posible.

Luego tenemos la *operación* del experimento, donde se van a preparar, ejecutar y validar los datos recogidos. Durante esta actividad se llevan a cabo tres tareas. La primera es la preparación, en donde se motiva a los participantes para así obtener resultados fiables. La ejecución en donde se lleva a cabo el experimento de acuerdo al diseño definido en la fase de planificación y es el momento en el que se recoge la información del experimento. Finalmente la validación de los datos, en donde una vez que se han recogido los datos, el experimentador debe comprobar que son razonables y han sido recogidos de forma adecuada.

En el *análisis e interpretación* se utilizan los datos ya validados en la etapa anterior y se emplean estadísticos descriptivos para así entender los datos de manera informal. Luego se analiza si el conjunto de datos a considerar debe ser reducido y tras analizar si hay variables redundantes, se lleva a cabo la comprobación de las hipótesis. Por último, la actividad de *presentación* y *empaquetado* está relacionada con la preparación de la documentación, que puede ser por un lado un artículo de investigación que permita la difusión de los resultados, o mediante un paquete de laboratorio para así llevar a cabo las réplicas del experimento que sean necesarias.

1.9. Estructura de la tesis

En este capítulo se ha presentado la motivación, el planteamiento del problema, las hipótesis que serán contrastadas, los objetivos de la tesis, además del contexto de la investigación y la metodología que será empleada durante el proceso de investigación. El proceso de investigación seguido en esta tesis se describe en varios capítulos que se corresponden con las actividades del modelo de transferencia de tecnología propuesto por Gorschek et al. (2006). La Figura 1-3 muestra la estructura general de la tesis donde se puede observar las distintas fases, flujo de actividades, y relación con los capítulos.

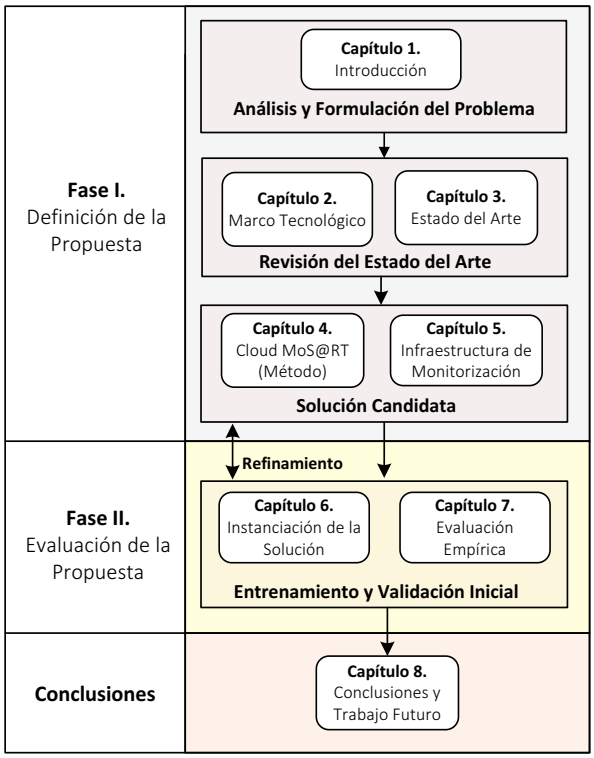


Figura 1-3. Estructura general de la tesis

La primera y segunda actividad del método de investigación están relacionadas con el *análisis y formulación del problema* que han sido presentados en el presente capítulo (Capítulo 1). La tercera actividad está relacionada con la *revisión del estado del arte* que será abordada mediante un análisis del marco tecnológico sobre el cual se plantea la contribución de esta tesis (Capítulo 2) y del análisis de las aproximaciones existentes para la monitorización de servicios en la nube (Capítulo 3). La cuarta actividad está relacionada con la *solución candidata*, que se corresponde con el planteamiento del método Cloud MoS@RT (Capítulo 4) y la infraestructura de monitorización que lo soporta (Capítulo 5). La quinta y sexta actividades, relacionadas con el *entrenamiento* y *validación inicial,* se corresponden con la instanciación del método e infraestructura de monitorización a una plataforma cloud específica para su uso posterior en un caso de estudio (Capítulo 6) y una familia de cuasi-experimentos (Capítulo 7). Estas evaluaciones son cruciales para refinar el método e infraestructura de monitorización propuestos a partir de la evidencia empírica obtenida sobre su utilidad.

Asimismo, a continuación, se describe el contenido de cada capítulo de forma más detallada:

- Capítulo 2: Marco tecnológico

  En este capítulo se abordan los ejes tecnológicos sobre los cuales se desarrolla esta tesis: computación en la nube, calidad en ambientes de distribución de servicios con modelo *Software as a Service (SaaS)*, acuerdos de nivel de servicio (SLA), monitorización de servicios, Ingeniería de Software Dirigida por Modelos y los modelos en tiempo de ejecución.

- Capítulo 3: Estado del arte

  En este capítulo se discuten las propuestas existentes de monitorización de servicios en plataformas distribuidas en general, en arquitecturas orientadas a servicios y en plataformas de computación en la nube. Este capítulo también discute varios estándares existentes relacionados con la computación en la nube, los SLA y el aseguramiento de calidad.

- Capítulo 4: Cloud MoS@RT, un método para la monitorización de servicios en la nube

  En este capítulo se presenta la contribución metodológica de este trabajo de tesis. En particular, se introduce un proceso de monitorización, sus actividades principales y tareas, así como una descripción de los artefactos de entrada y salida de cada una de las actividades del proceso.

- Capítulo 5: Infraestructura de monitorización

  En este capítulo se detalla la infraestructura de monitorización independiente de plataforma, sus componentes principales y una descripción detallada de los artefactos que permitirán soportar el proceso de monitorización y detección de incumplimientos del SLA.

- Capítulo 6: Instanciación del método Cloud MoS@RT a una plataforma específica

  En este capítulo se presenta la instanciación de la infraestructura de monitorización a la plataforma Microsoft Azure. Primero se presenta un configurador de la monitorización y después un middleware de monitorización, presentando su estructura, diseño, conceptos relevantes a la hora de su implementación y su funcionamiento. Por último, se describe los resultados de un caso de estudio que ilustra la aplicación del método de

monitorización, configurador y middleware propuestos en la monitorización de servicios en la nube.

- Capítulo 7: Evaluación empírica del método Cloud MoS@RT

  En este capítulo se presenta la evaluación empírica del método Cloud MoS@RT mediante una familia de cuatro experimentos, cuyo objetivo es la evaluación del método basado en la percepción de un grupo de usuarios.

- Capítulo 8: Conclusiones y trabajos futuros

  En este capítulo se presentan las conclusiones basadas en las contribuciones y los objetivos alcanzados a lo largo de este trabajo de investigación. Así mismo, se discuten las líneas de investigación actuales y futuras, junto con las publicaciones que se originaron a partir de este trabajo de investigación.

- Capítulo 9: Conclusions and further work

  Este capítulo presenta la traducción a lengua inglesa del capítulo 8, incluido en esta memoria con el fin de cumplir con los requisitos para la obtención del Doctorado Internacional.

- Apéndice A: Materiales del caso de estudio

  Este apéndice resume los materiales utilizados para llevar a cabo las tareas del estudio realizado en el Capítulo 6, así como también detalles relevantes de la implementación del configurador y middleware de monitorización.

- Apéndice B: Material Experimental

  Este apéndice resume los materiales experimentales empleados en los experimentos descritos en el Capítulo 7, incluyendo los formularios utilizados, captura de datos, cuestionarios y artefactos involucrados en las tareas experimentales y cuestionarios de evaluación.

Capítulo 2
Marco Tecnológico

Este capítulo presenta el marco tecnológico en el cual se ha desarrollado este trabajo de investigación, los temas principales que se abordan y que abarcan nuestra solución son: La computación en la nube, sus modelos de servicio y de despliegue. Por otro lado están las características de calidad con que los servicios en la nube deben contar durante la provisión del servicio. Además los acuerdos de nivel de servicio (SLA) en los que se especifican los requisitos funcionales (RF) y no funcionales (RNF) que ofrecerán los servicios contratados. También una introducción a los tipos de sistemas de monitorización que permiten la verificación de los RNF. Finalmente abordaremos la Ingeniería Dirigida por Modelos y los modelos en tiempo de ejecución y sus particularidades. Por tanto, la estructura del capítulo es la siguiente:

La sección 2.1 introduce los temas relacionados con la computación en la nube, modelos de servicio y modelos de despliegue.

La sección 2.2 define y detalla los requisitos no funcionales propios de ambientes de computación en la nube. Que diferencian a este tipo de computación de la computación tradicional. Modelos de calidad y su especificación para la evaluación de servicios de software en la nube.

La sección 2.3 aborda los acuerdos de nivel de servicio, las partes involucradas y el contenido principal de estos acuerdos.

La sección 2.4 introduce al enfoque de desarrollo dirigido por modelos y a los conceptos y generalidades, características y aplicaciones de los modelos en tiempo de ejecución.

La sección 2.5 presenta los sistemas de monitorización más comunes y los tipos de soluciones de monitorización que existen.

2.1. La computación en la nube

La computación en la nube es un modelo para habilitar el acceso ubicuo, bajo demanda, por red a un conjunto compartido de recursos de computación configurables (por ejemplo, redes, servicios, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y publicados con un esfuerzo mínimo de manejo o interacción del proveedor de servicios (Mell *et al.*, 2011).

La computación en la nube presenta cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue.

2.1.1. Características esenciales

La computación en la nube posee características propias y esenciales, que hacen de este, un modelo de distribución atractivo para organizaciones, las cuales buscan el aprovisionamiento de software y hardware con reducción de inversión inicial, mínimo esfuerzo de mantenimiento, capacidad de crecimiento según sus necesidades y acceso universal desde cualquier ubicación geográfica. Teniendo en cuenta estos requisitos, el NIST (Mell *et al.*, 2011) ha incorporado a la definición de computación en la nube, cinco características esenciales, a las cuales hemos agregado el concepto de multitenencia:

1. **Auto-Servicio Bajo-Demanda:** Esta característica plantea que un cliente puede aprovisionarse de un servicio, según lo necesite sin requerir de ninguna interacción humana por parte del servidor.
2. **Acceso Amplio a la Red:** Se refiere a la capacidad de acceder a los servicios provistos a través de mecanismos estándar que promuevan el uso heterogéneo a las diferentes plataformas (por ejemplo, a través de teléfonos móviles, tabletas, ordenadores portátiles y estaciones de trabajo).
3. **Puesta en Común de Recursos:** El proveedor de recursos de computación sirve a múltiples consumidores usando un modelo multitenencia, con diferentes recursos físicos y virtuales dinámicamente asignados y reasignados de acuerdo a la demanda del cliente.
4. **Elasticidad:** Se refiere a la capacidad de aprovisionar y publicar elásticamente y en algunos casos automáticamente, escalar rápidamente hacia arriba o hacia abajo los recursos de acuerdo a la demanda. Esto da la idea de una cantidad ilimitada de recursos.
5. **Servicio a Medida:** Los sistemas en la nube automáticamente controlan y optimizan el uso de los recursos a cierto nivel de abstracción, ofreciendo un servicio apropiado según su tipo (por ejemplo: almacenamiento, procesamiento, ancho de banda y cuentas activas de clientes). El uso de recursos puede ser monitorizado, controlado y reportado, proveyendo transparencia tanto para el proveedor como para el cliente.
6. **Multitenencia:** Las soluciones alojadas en ambientes cloud requieren el soporte para múltiples organizaciones así como también múltiples usuarios dentro de una misma organización. En un ambiente cloud, este concepto es referido como multitenencia. La multitenencia asegura el aislamiento entre servicios y datos a ser usados dichos servicios. La

multitenencia es importante para todo tipo de software alojado en la nube. Para cierto tipo de aplicaciones, debería ser suficiente proveer aislamiento de datos mientras que para otros podría requerirse almacenar datos sensibles en nubes privadas (Chauhan *et al.*, 2012).

2.1.2. Modelos de servicio

La computación en la nube provee servicios a tres niveles: Software como Servicio (SaaS), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS). El NIST (Mell *et al.*, 2011) define a los modelos de servicio de la siguiente manera:

- **Software como Servicio (SaaS):** Es la capacidad de proveer al cliente aplicaciones ejecutándose sobre una infraestructura en la nube. Las aplicaciones son accesibles desde diferentes dispositivos del cliente a través de interfaces ligeras, tales como: un navegador web o una interface de un programa. El cliente no maneja o controla la infraestructura, redes, servidores o sistemas operativos e incluso capacidades individuales de las aplicaciones, con la posible excepción de configuraciones específicas de la aplicación.
- **Plataforma como Servicio (PaaS):** Es la capacidad de proveer al cliente el control para desplegar aplicaciones sobre la nube en una plataforma específica, sin embargo no permite el manejo o control de la infraestructura en la nube, redes, sistemas operativos o almacenamiento. Además permite la configuración del ambiente de despliegue para las aplicaciones.
- **Infraestructura como Servicio (IaaS):** Es la capacidad provista al cliente para contar con: procesamiento, almacenamiento, redes y otros recursos fundamentales de computación, donde el cliente está habilitado para desplegar y ejecutar software arbitrariamente, lo cual puede incluir sistemas operativos, almacenamiento y aplicaciones, además de limitar el control de seleccionar componentes de red (firewalls).

2.1.3. Modelos de despliegue

Los modelos de despliegue tienen relación a la manera en la cual los recursos son provistos y la manera en que estos son desplegados para ser usados por sus clientes, el NIST (Mell *et al.*, 2011) clasifica los modelos de despliegue en:

- **Nube Privada:** La infraestructura de la nube está provista para uso exclusivo de una sola organización comprendida por múltiples usuarios (por ejemplo: unidades de negocio). Esta podría ser propiedad y manejada y operada por la organización.

- **Nube Comunitaria:** La infraestructura en la nube está provista para uso exclusivo por una comunidad de clientes de organizaciones que tienen preocupaciones comunes (por ejemplo: misión, requisitos de seguridad, políticas y consideraciones de cumplimiento). Esta puede ser manejada y operada por una o más organizaciones de dicha comunidad, una tercera parte o una combinación de ellas.
- **Red Pública:** La infraestructura de la nube es provista para uso abierto por el público en general. Esta puede ser propiedad, manejada y operada por un negocio, academia u organización gubernamental o una combinación de ellas. Esto existe sobre la premisa del proveedor en la nube.
- **Red Híbrida:** La infraestructura de la nube es una composición de dos o más infraestructuras distintas (privadas, comunitarias o públicas) que son puestas juntas por tecnologías estándares o propietarias que habilitan datos y portabilidad de aplicaciones (por ejemplo: balanceo entre las redes).

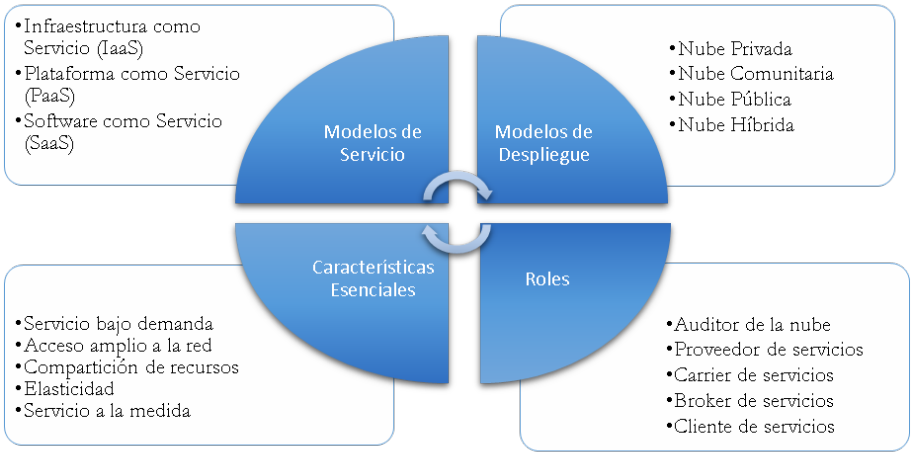


Figura 2-1. Generalidades de la Computación en la Nube

2.1.4. Estándares para la computación en la nube

Existen estándares que rigen la computación en la nube y las plataformas distribuidas, en lo referente a la especificación de un formato de virtualización abierto, vocabulario, arquitectura de referencia, interoperabilidad, acuerdos de nivel de servicio, entre otros (ISO, 2016) .

Teniendo como base el estándar ISO/IEC 17788:2014, en el cual se provee una visión sobre las definiciones y términos de la nube. Siendo esta recomendación aplicable a todos los tipos de organizaciones. Dentro de este

estándar se definen términos similares a los ya mencionados e introducidos por el NIST, más ciertos términos que son propios de las tecnologías cloud.

Como ejemplo podríamos citar conceptos relacionados a la disponibilidad, confidencialidad, seguridad de la información, integridad, entre otros. Términos que serán abordados a lo largo de este trabajo y que dentro de este estándar, han sido recopilados incluso desde diversas fuentes. Más información y a manera de glosario la podemos encontrar en la página en donde se describe el estándar en su totalidad ("ISO/IEC 17788:2014. Information technology -- Cloud computing -- Overview and vocabulary," 2014)

2.1.5. Plataformas de computación en la nube

Dentro de la adopción de nubes públicas más comunes, se ha visto que en los últimos años se han popularizado varios productos comerciales que ofrecen los servicios de computación en la nube. Los proveedores cloud continúan añadiendo servicios y sus costos caen, lo que ha contribuido a su masificación y adopción.

Plataformas tales como Amazon Web Services, Microsoft Azure, VMware Cloud Air entre otras, como se muestra en la Figura 2-2, se han ido adoptando más a menudo por organizaciones como una solución atractiva, altamente confiable y con atributos de calidad deseables. Un estudio profundo ha sido realizado en RightScale 2016 State of the Cloud Report (Weins, 2016), en el cual se muestra las tendencias de adopción de las diferentes soluciones en las nubes y sus modelos de despliegue. En este reporte, se muestra también que a inicios de éste año (2016), se ha incrementado también el uso de nubes privadas, lo que se muestra en la Figura 2-3

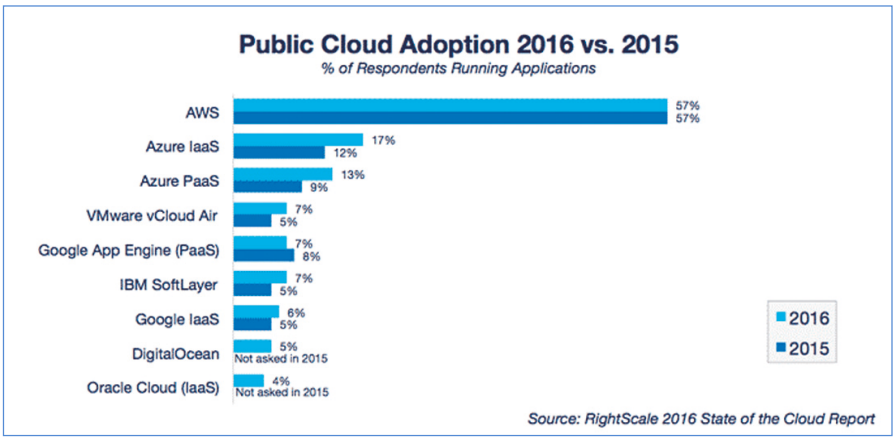


Figura 2-2. Adopción de Plataformas de Computación en la Nube Públicas

Además, el fuerte crecimiento del uso de nubes privadas, combinadas con la ubicuidad de nubes públicas, significa que la mayoría de organizaciones están abriéndose a ambientes híbridos.

Por otro lado, con el incremento de la madurez de los usuarios y proveedores, se está observando una reducción en problemas de seguridad en la nube (Weins, 2016).

Finalmente, cabe destacar que entre los proveedores cloud, AWS mantiene un liderazgo, pero Azure continúa avanzando en la incursión de su tecnología e incrementando el número de usuarios que hacen uso de esta solución.

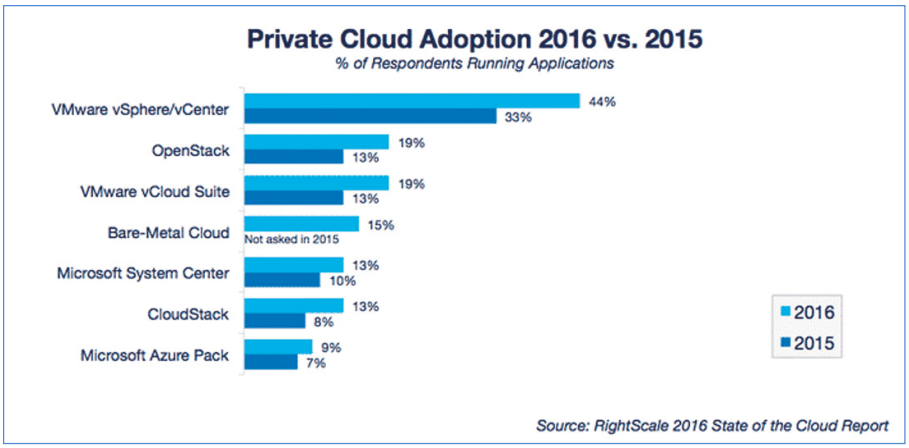


Figura 2-3. Adopción de Ambientes para Computación en la Nube Privados

A continuación se muestran algunas de las más populares plataformas, con la finalidad de dar un vistazo a las características más popularmente ofertadas de estas soluciones.

2.1.5.1. Microsoft Azure

Microsoft Azure (2014) es una plataforma ofrecida como servicio, antes denominada Windows Azure y *Azure Services Platform*, ésta es propiedad de Microsoft. Su versión beta fue anunciada en la conferencia de desarrolladores profesionales de Microsoft en el año 2008 y posteriormente, para el año 2010, pasó a ser un producto comercial. Esta plataforma tiene diferentes servicios para aplicaciones, que van desde el alojamiento de aplicaciones en sus centros de procesamiento para que sean ejecutados sobre su infraestructura, hasta servicios de comunicación segura. De ahí, Microsoft Azure es una colección cada vez mayor de servicios integrados en la nube (análisis, proceso, base de datos, móviles, redes, almacenamiento y Web).

Azure además utiliza un sistema operativo especializado, un clúster localizado en los servidores de datos de Microsoft que se encargan de manejar los recursos almacenados y procesamiento para proveer los recursos para las aplicaciones que se ejecutan sobre su plataforma.

Azure es capaz de soportar distintos lenguajes de programación, herramientas y servicios, incluyendo los pertenecientes a terceros así como un ecosistema creado por Microsoft basado en .Net. De ahí, Windows Azure, es el sistema operativo de la nube de Microsoft. Éste proporciona un entorno gestionado para la ejecución y despliegue de aplicaciones y servicios en la nube. Azure proporciona a los desarrolladores un entorno de computación bajo demanda y almacenamiento alojado en los centros de datos de Microsoft para aplicaciones en la Web (Estoy en la Nube, 2016) .

2.1.5.2. Amazon Web Services (AWS)

Amazon Web Services (AWS) ofrece un amplio conjunto de servicios globales de computación, almacenamiento, bases de datos, análisis, aplicaciones e implementaciones que ayudan a las organizaciones a avanzar con más rapidez, reducir costos de TI y escalar aplicaciones. Estos servicios tienen la confianza de las mayores compañías y las empresas emergentes más innovadoras para respaldar una amplia variedad de cargas de trabajo, como las aplicaciones web y móviles, IoT, el desarrollo de juegos, el almacenamiento y procesamiento de datos, el almacenamiento en general, el archivado y muchas otras (Amazon, 2016).

Entre los servicios más destacados de AWS están S3 y EC2. El primero permite alojar las aplicaciones y el segundo ofrece lo que podría llamarse un servidor virtual completo ejecutándose en una plataforma de Amazon.

AWS está disponible en 190 países, 12 regiones geográficas y más de 50 puntos de presencia locales. Lo que la hace una opción global muy competitiva.

2.1.5.3. Google Cloud Platform

Google Cloud Platform (Google, 2016) ofrece un amplio espectro de productos y servicios cloud para computación, almacenamiento, red, *big data*, aprendizaje de máquina, operaciones y más.

Así, Google ha invertido los últimos 15 años en desplegar una de las plataformas de comunicaciones más potentes y desde hace relativamente poco tiempo (4 años) ha dado acceso a terceros para que puedan desplegar sus propias aplicaciones, compartiendo de esta manera hardware con *Gmail*, *YouTube* y *Google Search*. Entre los productos más destacados de esta plataforma están: *Compute*

Engine: producto IaaS que permite gestionar máquinas virtuales, dando flexibilidad y alta disponibilidad con elementos como redes privadas, balanceadores de carga entre otros. *App Engine:* que es un producto PaaS de Google que permite enfocarse principalmente en el desarrollo de funcionalidades con una alta escalabilidad y disponibilidad. Con esta herramienta no hay que hacer actualizaciones de seguridad o preocuparse por caídas del sistema, permitiendo una amplia gama de lenguajes de programación para el despliegue de software en su infraestructura. *Cloud Storage* permite un almacenamiento masivo de datos en la nube replicado en varios lugares del mundo para dar alta disponibilidad y redundancia de información. *Big Query*: que permite *Big Data* para hacer consultas interactivas sobre almacenes muy grandes de información. Finalmente, *Cloud SQL* que es un clúster MySQL montado y funcionando que permite lanzar proyectos en tiempos cortos.

2.2. Calidad de servicios de software en la nube

La calidad de software tiene como objetivo asegurar la conformidad con los requisitos de software. Esto significa que un producto de software necesita atender tanto a especificaciones implícitas como explícitas, con las cuales un producto es ofrecido. De ahí que la meta es asegurar que un producto satisfaga todas las expectativas del cliente (Kan, 2003) .

La calidad de un producto de software es uno de los criterios de elección más importantes. En este contexto, se puede decir que la elección del software y los servicios en la nube están dirigida por los factores de calidad que estos servicios presentan; de ahí, que el éxito depende del cumplimiento de sus requisitos funcionales y no funcionales (Lew *et al.*, 2008). Los requisitos no funcionales y sus atributos de calidad relacionados, ayudan a definir la calidad en un producto de software.

Los clientes esperan que las aplicaciones y servicios desplegados en infraestructuras de computación en la nube, sean entregadas con altas características de calidad de servicio, confiabilidad, disponibilidad y latencia, muy similar a como si estuviesen desplegadas en configuraciones tradicionales nativas. La computación en la nube introduce nuevos retos de calidad que se producen debido a la computación virtualizada y a la manera en la que se proveen los recursos de hardware tales como la memoria, almacenamiento y recursos de red (Bauer *et al.*, 2013).

Los servicios de software en la nube requieren ciertos requisitos no funcionales específicos de este modelo de aplicaciones. Tales requisitos han sido

identificadas por varios autores como es el caso de Lee *et al. (*2009), entre ellos están la reusabilidad, manejo de datos por parte del proveedor de los servicios, personalización de los servicios, disponibilidad, escalabilidad, pago por uso, etc. A partir de la identificación de estos requisitos no funcionales, se han definido modelos de calidad que permitan una fácil clasificación de estos requisitos y su descomposición en características, sub-características, atributos y métricas. En las siguientes sub-secciones se establecerán conceptos relacionados a los modelos de calidad, que serán de utilidad para la comprensión de este trabajo.

2.2.1. Modelos de calidad

Los modelos de calidad han constituido un tópico de investigación por algunas décadas y un gran número de modelos de calidad han sido propuestos. Los primeros modelos de calidad datan de los años 70's cuando varios autores tales como McCall *et al. (*1977) describieron las características de calidad y su descomposición. Estos autores usan una descomposición jerárquica del concepto de calidad en características tales como mantenibilidad o confiabilidad. Luego, aparecieron variaciones de esos modelos, siendo una de las más populares FURPS propuesta por Joseph, (1988), la cual descompone la calidad en: funcionalidad, usabilidad, confiabilidad, rendimiento y mantenibilidad. Además de esta descomposición jerárquica, la idea principal de estos modelos es que se pueda llegar a especificar la calidad a un nivel en la que ésta pueda ser medida y de ahí evaluada.

Esta clase de modelos de calidad se han convertido en las bases del estándar internacional ISO/IEC 9126, (2001) en su primera versión de 1991. El estándar define una descomposición en características de calidad y sugiere un pequeño número de métricas para la evaluación. A continuación, se presentó el sucesor del estándar ISO/IEC 9126, el ISO/IEC 2510, (2011); el cual presentó varias mejoras, incluyendo un modelo de referencia para la medición. A continuación se muestra un resumen del Estándar ISO 25000 (SQuaRE).

2.2.2. Estándar ISO 25000 (SQuaRE)

La meta perseguida en la creación de esta norma es dar un paso hacia un conjunto de estándares organizados de manera más lógica, enriquecidos con nuevas aportaciones y unificados con respecto a normas anteriores para ser capaces de cubrir los dos principales procesos: especificación de requisitos de calidad del software y evaluación de la calidad del software, soportados por un proceso de medición. SQuaRE se centra exclusivamente en el producto software estableciendo criterios para su especificación, su medición y su evaluación. Es

decir, básicamente se trata de una unificación y revisión de los estándares ISO/IEC 9126 e ISO/IEC 14598.

La comprensión de la especificación y la evaluación de la calidad de software y sistemas de cómputo, es un factor que asegura valor a los stakeholders. Esto puede ser conseguido definiendo las características de calidad deseadas y necesarias asociadas con los stakeholders, sus metas y objetivos para el sistema. Esto incluye las características de calidad relacionadas al sistema de software y los datos, como también, el impacto que el sistema tiene en los stakeholders.

El SQuaRE define seis características de calidad y describe un modelo de proceso de evaluación de producto software. SQuaRE por tanto revisa el ISO/IEC 9126-1:2001, e incorpora las mismas características de calidad de software con algunas enmiendas.

- El ámbito de los modelos de calidad ha sido extendido para incluir sistemas de computación y calidad en uso desde la perspectiva del sistema.
- El contexto cubierto ha sido añadido como una característica de calidad en uso, con sub-características de completitud de contexto y flexibilidad.
- La seguridad ha sido añadida como una característica, además de una sub-característica de funcionalidad, con sub-características de confidencialidad, integridad, no repudio, contabilidad y autenticidad.
- Compatibilidad (incluyendo interoperabilidad y co-existencia) ha sido añadida como una característica.
- Las siguientes sub-características han sido añadidas: completitud funcional, capacidad, protección de errores de usuario, accesibilidad, disponibilidad, modularidad y reusabilidad.
- Las sub-características de conformidad han sido removidas, como conformidad con leyes y regulaciones.

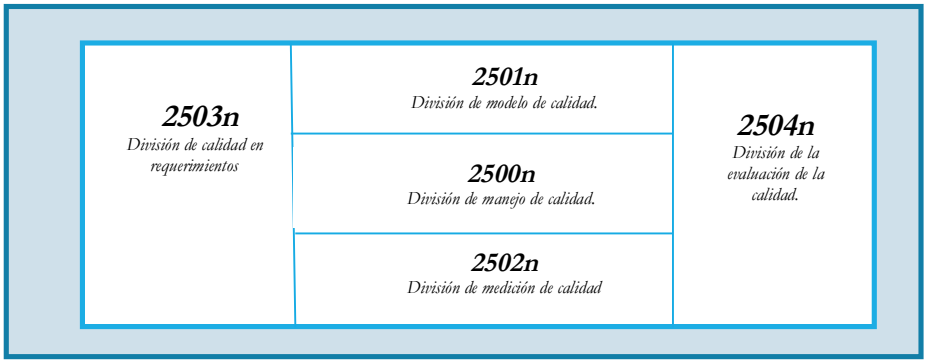


Figura 2-4. Organización de las series SQuaRE

- Los modelos de calidad interna y externa han sido combinados como el modelo de calidad de producto.
- Cuando es apropiado, las definiciones genéricas han sido adoptadas, además usando las definiciones específicas de software.
- Se han dado nombres más precisos a algunas características y sub-características.

2.2.2.1. Divisiones del Estándar SQuaRE

- ISO/IEC 2500n – División de Manejo de la calidad. Los Estándares internacionales que forman esta división definen todos los modelos comunes, términos y definiciones además referidos a todos los otros estándares internacionales del SQuaRE. La división además provee requisitos y guías para una función de soporte que es responsable del manejo de los requisitos, especificación y evaluación de la calidad de producto del software.
- ISO / IEC 2501n – División del modelo de calidad. Los Estándares Internacionales que forman esta división presentan modelos de calidad detallados para sistemas de cómputo y productos de software, calidad en uso y datos. Una guía práctica del uso de los modelos de calidad es además provista.
- ISO / IEC 2502n – División de Medida de la Calidad. Los estándares internacionales que forman esta división incluyen un modelo de referencia para la medir la calidad del producto, definiciones matemáticas o medidas de calidad y guías prácticas para su aplicación. Presenta métricas aplicadas a la calidad interna del software, calidad externa de software y calidad en uso.
- ISO / IEC 2503n – División de calidad de requisitos. El estándar que forma esta división ayuda a especificar los requisitos de calidad. Estos requisitos de calidad pueden ser usados en el proceso de elicitación de requisitos de calidad, para un producto de software que puede ser desarrollado o como entrada de un proceso de evaluación. El proceso de definición de requisitos es mapeado a procesos técnicos definidos en ISO / IEC 15288 – Tecnología de la información – Manejo del ciclo de vida – Proceso del sistema de ciclo de vida.
- ISO / IEC 2504n – División de la calidad de la evaluación. Los estándares que forman esta división proveen requisitos, recomendaciones y guías para la evaluación del producto de software ya sea ejecutada por evaluadores independientes, adquirientes o desarrolladores. Además, presenta el soporte para documentar una medida como un módulo de evaluación.

- ISO/IEC 25050-25099 – División de extensión del SQuaRE. Estos Estándares internacionales, actualmente incluyen requisitos para la calidad de software que servirán para reportes de usabilidad.

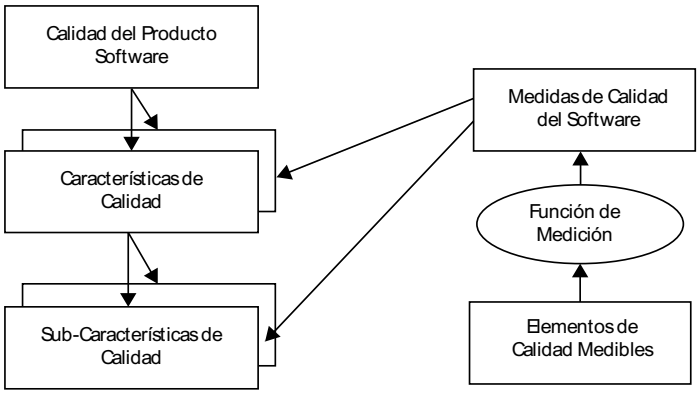


Figura 2-5. Modelo de medición de la calidad del producto software según SQuaRE.

2.2.2.2. Características y Atributos de Calidad de SQuaRE

El modelo de calidad representa la piedra angular en torno a la cual se pueden encontrar las características de calidad que se van a tener en cuenta al momento de evaluar las propiedades del software.

Hay que tener en cuenta que el estándar ISO/IEC 25010 (2011), plantea la descomposición en características y sub-características, sin embargo dependiendo del producto de software que se quiera evaluar, hará falta definir atributos y métricas específicos, que permitan realizar las mediciones.

La calidad de producto software será el grado en el cual se satisfacen los requisitos de calidad del usuario aportando así un valor. El modelo de calidad definido por el estándar ISO/IEC 25010 (2011) se encuentra compuesto por ocho características que se muestran en la Figura 2.6 y se describen a continuación:

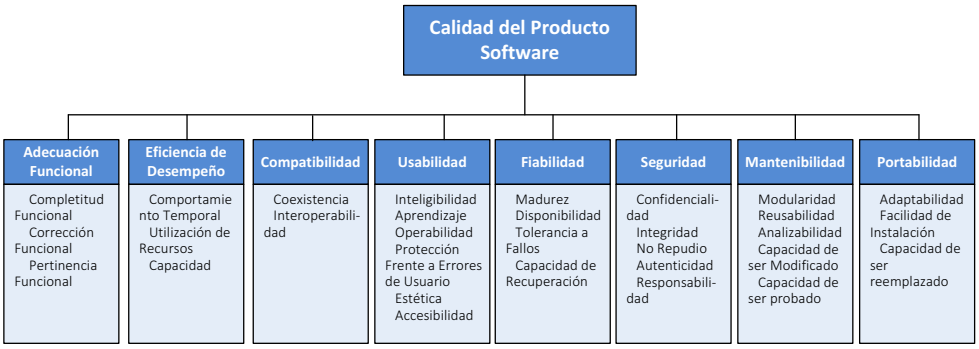


Figura 2-6. Modelo de calidad del producto definido por la ISO/IEC 25010

Adecuación Funcional

Es la capacidad del producto para proporcionar funciones que satisfacen las necesidades declaradas e implícitas. Esta característica a su vez se divide en las siguientes sub-características:

- **Completitud Funcional:** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- **Corrección Funcional:** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- **Pertinencia Funcional:** Capacidad del producto para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

Eficiencia de Desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes sub-características:

- **Comportamiento temporal:** Los tiempos de respuesta y procesamiento y los ratios de *throughput* de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (*benchmarking*) establecido.
- **Utilización de recursos**: Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Capacidad:** Grado en que los límites máximos de un parámetro de un producto software cumple con los requisitos.

Compatibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información, o compartir el mismo entorno de hardware o software. Esta característica se subdivide en:

- **Coexistencia:** Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
- **Interoperabilidad:** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Esta característica se subdivide a su vez en las siguientes sub-características:

- **Madurez:** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
- **Disponibilidad:** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- **Tolerancia a fallos:** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- **Capacidad de recuperación:** Capacidad del producto software para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallo.

Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no tengan acceso a ella. Esta característica se subdivide a su vez en las siguientes sub-características:

- **Confidencialidad**: Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- **Integridad:** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.
- **No repudio:** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.

- **Responsabilidad:** Capacidad de rastrear de forma inequívoca las acciones de una entidad.
- **Autenticidad:** Capacidad de demostrar la identidad de un sujeto o un recurso.

Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Esta característica se subdivide a su vez en las siguientes sub-características:

- **Modularidad:** Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- **Reusabilidad:** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- **Analizabilidad:** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **Capacidad para ser modificado:** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
- **Capacidad para ser probado:** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes sub-características:

- **Adaptabilidad:** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
- **Capacidad para ser instalado:** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
- **Capacidad para ser reemplazado:** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

2.3. Acuerdos de nivel de servicio (SLA)

Los acuerdos de nivel de servicio (SLA) son una pieza clave en el manejo de la calidad de servicio tanto para el proveedor como para el cliente. Existen varias definiciones de SLA, entre las que encontramos:

La norma ISO 20000-1 (2001) define un SLA como: *"un documento acordado entre el proveedor del servicio y su cliente que identifica servicios y objetivos del servicio.*

Emeakaroha *et al.* (2010) dice que "un SLA es *un contrato firmado entre el cliente y el proveedor de servicios. Este establece los términos del servicio incluyendo los requisitos no funcionales (RNF) del servicio especificado como la calidad de servicio (QoS), obligaciones, facturación del servicio y penalidades en caso de violaciones del acuerdo".*

Cada servicio está típicamente acompañado por un SLA, el cual define las mínimas garantías que un proveedor ofrece a sus clientes. La falta de estandarización en servicios basados en la nube, implica una falta de claridad en los SLA ofrecidos por diferentes proveedores (Baset, 2012).

Por otro lado, se ha incrementado el interés tanto en las comunidades industriales como académicas y existen muchos retos con respecto a la calidad de los servicios (Panzieri *et al.*, 2013). De ahí, la habilidad para especificar la calidad de servicio es un tema importante para clientes y proveedores de servicios. Sin embargo, existe una falta de confianza entre clientes y proveedores y dudas sobre cómo tienen que asegurar que los proveedores de servicio ofrecen la calidad de servicio esperada (Nallur *et al.*, 2013).

Un acuerdo de nivel de servicio se descompone en partes fácilmente diferenciadas, las mismas que se detallan a continuación:

2.3.1. Partes generales de un SLA

En términos generales, un SLA típico de un proveedor de servicios en la nube, tiene los siguientes componentes (Baset, 2012):

- **Garantías de servicio:** especifica las métricas que un proveedor se empeña en reunir durante un período de tiempo. Las fallas en esas métricas resultarán en un crédito para el cliente. Por ejemplo: La disponibilidad será del 99.9%, el tiempo de respuesta será menos que 50 ms, la recuperación ante desastres y la falta de tiempo de resolución.
- **Período de tiempo para la garantía de servicio:** describe la duración en la que una garantía de servicio debe ser reunida. El período de tiempo puede ser de un mes o un tiempo producido desde que el último reclamo fue realizado.

- **Granularidad de la garantía de servicio:** describe la escala de recursos sobre la cual un proveedor especifica una garantía de servicio. Por ejemplo, la granularidad puede ser por servicio, por centro de datos, por instancia o por transacción. Similar al período de tiempo, la garantía de servicio puede ser rigurosa si la granularidad del servicio es a un nivel demasiado fino. La granularidad de la garantía de servicio puede además ser calculada como una función de agregación de los recursos considerados, tales como instancias o transacciones. Por ejemplo, el tiempo de actividad *(uptime)* de todas las transacciones ejecutándose debe ser mayor que el 99.95%. Sin embargo, tal garantía implica que algunas instancias en la computación de SLA pueden tener un nivel más bajo. Como consecuencia, esto deja al proveedor un margen de maniobra para manejar sus servicios.
- **Exclusiones de la garantía de servicios:** son las instancias que son excluidas de los cálculos de las garantías de servicio. Estas exclusiones típicamente incluyen abuso del sistema por un cliente o cualquier tiempo de parada asociado con un mantenimiento programado.
- **Crédito de servicio:** es la cantidad de crédito aplicada al cliente para pagos futuros si los términos de garantía no son reunidos. La cantidad puede ser un crédito completo o parcial al cliente por el servicio afectado.
- **Medición de la violación de servicio y reportes:** describe cómo y quién medirá y reportará las violaciones de las garantías de servicio, respectivamente.

2.3.2. Partes de SLA en estándares existentes

Existen estándares para la especificación de SLA para servicios web, estos mismos estándares pueden ser adaptados con ciertos ajustes, a la computación en la nube, dadas sus características tecnológicas; estos estándares fijan secciones, con el fin de abarcar todas las posibles partes que contienen estos documentos. Tal es el caso del Web Service Level Agreement Language (WSLA), éste es un estándar para la monitorización de SLA, propuesto por IBM (Keller *et al.*, 2003), basado en XML y definido como un esquema XML. Este permite a los actores especificar las métricas de rendimiento asociadas con una aplicación de servicios web, objetivos a alcanzar y acciones que se tomarán en caso de que éste rendimiento no sea alcanzado. El WSLA establece tres secciones principales: Una sección que describe las partes, una sección que contiene una o más definiciones de servicio y la sección que define las obligaciones de las partes (Ludwig *et al.*, 2003).

- **Partes:** el WSLA describe las partes envueltas en el manejo de los servicios web. Esta incluye las partes signatarias del acuerdo como también las partes de soporte que son incluidas en el SLA para actuar por parte del proveedor de los servicios o del cliente, estas partes de soporte no pueden ser responsabilizadas de los incumplimientos del SLA.
- **Definiciones del Servicio:** Describe los servicios que se aplicarán al WSLA. Las definiciones del servicio representan el entendimiento común de las partes del contrato de la estructura del servicio, en términos de operaciones y los parámetros del servicio y métricas que son las bases del SLA. Estas además incluyen la especificación de la medición de las métricas del servicio.
- **Obligaciones:** Definen el nivel de servicio que es garantizado con respecto a los parámetros del SLA definidos en la sección de definición de servicios. Las promesas para ejecutar acciones bajo ciertas condiciones son además representadas en esta sección.

Otro de los estándares de especificación del SLA es el WS-Agreement (Andrieux et al., 2004), el cual es una especificación de Global Grid Forum de 2005. Define así también un lenguaje basado en XML para acuerdos como también un protocolo para publicar las capacidades de los proveedores de servicios, crear acuerdos entre clientes y proveedores y monitorizar el cumplimiento de los acuerdos. Al igual que el WSLA, el WS-Agreement provee un esquema XML que define la estructura total del documento del acuerdo. Además, la especificación WS-Agreement define un protocolo para negociar y establecer acuerdos dinámicamente basados en servicios web (un conjunto de definiciones WSDL). Un SLA en formato WS-Agreements contiene las siguientes partes (Bianco *et al.*, 2008):

> **Identificación del Acuerdo:** Un ID obligatorio seguido por un nombre opcional.
> **Contexto del Acuerdo:** El cual incluye la identificación del as diferentes partes, varios metadatos sobre el acuerdo (tales como una fecha de expiración y una plantilla ID en caso de que el acuerdo sea creado siguiente una plantilla). Y los atributos definidos por el usuario.
> **Los términos del acuerdo:** los cuales son divididos en términos del servicio y términos de garantía. Éstos pueden ser además extendidos.
> **Términos del servicio:** los cuales son divididos en términos de servicio y términos de garantía. Éstos también pueden ser extendidos.
> **Términos de servicio que identifican y describen los servicios que son sujeto del acuerdo:** Los elementos actuales en la descripción del servicio deben ser personalizados basados en necesidades específicas del dominio. Los términos del servicio además comprenden las propiedades

medibles del servicio y exponen las propiedades que pueden ser usadas para expresar los objetivos de nivel del servicio. Estos son equivalentes a los parámetros WSLA.

Los términos de garantía que especifican los niveles de calidad de servicio que las partes acuerdan: Ejemplo de estos términos puede ser la disponibilidad de un servicio y el tiempo de respuesta promedio ante una petición.

Como se puede observar en estos dos ejemplos de especificación de servicios, las partes acuerdan implícitamente los términos del acuerdo, servicio y garantía que ofrecen al proveer sus servicios, utilizando los SLA.

Además, existen estándares que especifican los conceptos, métricas y requisitos para conformar los SLAs. Tales estándares están actualmente siendo desarrollados, según lo informa la página de la ISO (ISO/IEC 19086-1:2016 – SLA, overview y conceptos; ISO/IEC 19086-2:2016, Métricas; ISO/IEC 19086-3:2016, requisitos centrales para su formulación). Como trabajo futuro, se espera una vez lanzados en sus versiones finales poder analizarlos y de ser necesario, realizar mejoras a la solución que se propone en los siguientes capítulos (ISO, 2016).

2.4. Ingeniería de software dirigida por modelos

La Ingeniería de Software Dirigida por Modelos es una metodología de desarrollo, la cual centra sus esfuerzos en crear y explotar modelos de dominio (esto es, representaciones abstractas de conocimiento y actividades que gobiernan un dominio particular). El enfoque dirigido por modelos se espera que incremente la productividad maximizando la compatibilidad entre individuos y equipos de trabajo sobre el sistema (vía una estandarización de la terminología y las mejores prácticas usadas en el dominio de la aplicación) (Cordova, 2012).

Un paradigma de modelado para MDE se considera efectivo, si sus modelos tienen sentido desde el punto de vista de un usuario que es familiar con el dominio, y si ellos pueden servir como una base para implementar sistemas. Los modelos son desarrollados a través de una comunicación extensa entre los gerentes de producto, diseñadores, desarrolladores y usuarios del dominio de la aplicación.

Elevar el nivel de abstracción en la descripción de los problemas y sus soluciones ha sido una de las preocupaciones más comunes dentro de la evolución de la industria del software. El modelado, es una herramienta clave en

todo proceso científico o de ingeniería (Brambilla *et al.,* 2012). En cualquier rama de la ingeniería pueden crearse modelos que abstraigan los detalles de interés de su dominio y permitan el análisis de las soluciones a dicho problema. En el campo de la Ingeniería del Software, el uso de modelos no es nuevo, han sido utilizados durante mucho tiempo para documentar tanto la estructura interna del software como la estructura de los sistemas, sus interfaces olas estructuras de datos subyacentes (Gonzalez-Huerta, 2014).

En el *Desarrollo de Software Dirigido por Modelos* (DSDM), se aborda el desarrollo de sistemas de software mediante el refinamiento o transformación sucesiva de modelos. El sistema se modela en términos del dominio del problema. Cada nuevo modelo, desciende en el nivel de abstracción, añadiendo nuevos detalles de la plataforma destino a los modelos definidos en niveles de abstracción superior, hasta llegar al código del sistema en desarrollo. De esta manera, los modelos pasan de ser elementos de documentación, a ser parte del software, esto permite incrementar la velocidad de desarrollo y la calidad del software obtenido (Stahl *et al.*, 2006).

En la Ingeniería del Software tradicional, el proceso de desarrollo está dividido en algunas actividades o fases alcanzadas desde la especificación de los requisitos sobre la construcción del despliegue y mantenimiento. Esas fases de desarrollo de software y ejecución son estrictamente separadas una de otra, y modificar o ejecutar este software es hecha mediante un re-despliegue o cambio de los componentes del software. Sin embargo, en años recientes la distinción entre desarrollo de software y ejecución se ha difuminado, esto debido a aplicaciones modernas que han tenido que auto-adaptarse de acuerdo a los cambios de requisitos y ejecución mientras están siendo ejecutados (Baresi *et al.*, 2010). Además, ejecutar cambios a nivel de modelo mejora la sincronización entre los artefactos de diseño y la implementación del sistema de software (Giese *et al.*, 2006). Para realizar tal sincronización entre el sistema ejecutándose y sus modelos, el sistema en ejecución necesita una auto-representación de sí mismo basado en modelos que están causalmente conectados (Blair *et al.,* 2009).

2.5. Modelos en tiempo de ejecución

Basados en la anteriormente mencionada utilización de modelos, un modelo en tiempo de ejecución es definido como una abstracción de un sistema ejecutándose que puede ser manipulado en tiempo de ejecución para un propósito en particular (Bencomo *et al.*, 2013). Una definición alternativa es provista por Blair *et al. (*2009), en donde define un modelo en tiempo de ejecución como una auto-representación causalmente conectada de un sistema

asociado que enfatiza en la estructura, comportamiento o metas del sistema desde una perspectiva del espacio del problema a ser contemplado.

Un análisis de las dinámicas y ejecutabilidad de los modelos en tiempo de ejecución ha sido abordada por Breton *et al. (*2001). Los autores diferencian tres partes de los modelos dinámicos.

- **Parte de definición:** es la parte estática de una modelo, definida en tiempo de ejecución.
- **Parte de situación:** incluye todos los elementos que describen el estado dinámico de un modelo durante su ejecución, y finalmente la
- **Parte de ejecución:** especifica las transiciones del modelo desde un estado a otro, en otras palabras la lógica de ejecución.

Éste fue un punto de partida para el trabajo de Lehmann *et al. (*2010), con la diferencia de que el trabajo de Breton *et al. (*2001) se centra en modelos ejecutables y que no es aplicable del todo a los modelos en tiempo de ejecución. De ahí que Lehmann *et al. (*2010) identifican las propiedades típicas de un modelo en tiempo de ejecución que son:

- Una parte prescriptiva del modelo especificando cómo el modelo debería ser.
- Una parte descriptiva del modelo especificando cómo el modelo ese.
- Las modificaciones válidas sobre el modelo de las partes descriptivas, ejecutable en tiempo de ejecución.
- Las modificaciones válidas de las partes descriptivas, ejecutables en tiempo de ejecución.
- Las modificaciones válidas de las partes prescriptivas, ejecutables en tiempo de ejecución.
- La conexión causal en forma de un flujo de información entre el modelo y el sistema.

2.5.1. Objetivos de los modelos en tiempo de ejecución

En esta sección, se observan los objetivos perseguidos por un sistema el cual utiliza modelos en tiempo de ejecución según la clasificación propuesta por Szvetits *et al. (*2013). Ellos divisan siete clases de objetivos de los modelos en tiempo de ejecución:

- **Adaptación:** Cambian el sistema de acuerdo a los cambios de ambientes y requisitos.

- **Monitorización, simulación y predicción:** Monitoriza el sistema usando modelos los cuales ayudan a trazar el comportamiento de la aplicación. Simula cambios a nivel de modelo y analiza sus consecuencias. Predice propiedades del sistema tales como rendimiento por análisis a nivel de modelo.
- **Abstracción:** Interactúa con el sistema usando modelos que son cercanos al espacio del problema.
- **Independencia con la plataforma:** Provee vistas independientes dela plataforma sobre el sistema bajo observación.
- **Consistencia y conformidad:** Evita contradicciones entre diferentes partes del software y/o artefactos de desarrollo. Asegura la conformidad hacia otros modelos o restricciones de integridad en general.
- **Chequeo de políticas y cumplimiento:** Adhiere a políticas (en tiempo real) y regulaciones de seguridad.
- **Manejo de errores:** Habilita la depuración basada en modelos, localización de fallos, trazabilidad y auto-corrección.

2.5.1.1. Adaptación

Uno de los principales objetivos de usar modelos en tiempo de ejecución es la fácil adaptación a ambientes que cambian continuamente de requisitos. Cuando se adapta un sistema, un reto importante es la manera de adaptación correcta con un sistema compuesto. En (Szvetits *et al.*, 2013), se han expuesto cinco escenarios principales direccionados por la adaptación a través de los modelos en tiempo de ejecución:

- **Adaptación de Interfaces de Usuario:** La adaptación de interfaces trata sobre la interacción de interfaces con el usuario dinámica, manejando el sistema desde distintas perspectivas. Los modelos en tiempo de ejecución sirven para realizar tales escenarios describiendo posibles flujos de interacción, restricciones y partes del sistema a ser personalizados por el usuario.
- **Cambio de Requisitos y Contextos**: Los cambios de requisitos y contextos direccionan las modificaciones necesarias para la evolución de los requisitos y cambios en ambientes operacionales. Los principales cambios caen en hallazgos fuera de configuraciones alternativas del sistema y análisis de configuraciones encontrando beneficios comparados con el estado del sistema actual.
- **Cumplimiento de la Calidad de Servicio:** En el escenario del cumplimiento de la calidad del servicio, la satisfacción de las propiedades no funcionales es direccionada, por ejemplo, un cierto nivel de rendimiento o

confiablidad. Esto es especialmente requerido cuando se trata con ambientes operativos volátiles y con restricciones de tiempo. Usando modelos en tiempo de ejecución, posibles estados del sistema pueden ser predichos cuando un cambio del ambiente ocurre y tales estados sirven como una base para analizar si las condiciones requeridas pueden reunirse si cambia ese ambiente.

- **Análisis del impacto de cambio:** El análisis de impacto del cambio direcciona la detección de partes del sistema afectadas en caso de cambios de requisitos o ambientales, como también las consecuencias de transiciones a estados futuros. Las representaciones en tiempo de ejecución de modelos de diseño pueden ser usadas para identificar componentes conectados para detectar componentes afectados, especialmente cuando se usan modelos arquitectónicos e información sobre composiciones de servicios.

Como resumen, la adaptación es principalmente usada para resolver los problemas de predicciones imprecisas y cambios en los ambientes de ejecución.

2.5.1.2. Monitorización, simulación y predicción

Además de la adaptación, los modelos en tiempo de ejecución pueden ser usados para monitorizar un sistema, simular evoluciones futuras analizando impactos sobre el nivel del modelo y predecir las propiedades del sistema como rendimiento y confiabilidad en caso de reconfiguración del sistema. Los modelos habilitan la monitorización, simulación y predicción a un nivel más alto de abstracción (cercano al espacio del problema), direccionando los problemas de bajo nivel de abstracción. La monitorización está además relacionada al problema de chequear o cumplir reglas o restricciones, si los sistemas han sido creados con ésta finalidad. Más específicamente, la monitorización, simulación y predicción están cercanamente relacionadas a la adaptación ya que la mayoría de técnicas de adaptación requiere monitorizar el sistema en ejecución. Además, la simulación y predicción son usadas como entrada para adaptaciones, por ejemplo, en el análisis y planeación de la técnica del bucle de control autonómico.

2.5.1.3. Abstracción e Independencia de la Plataforma

Un objetivo importante de los modelos de software es levantar el nivel de abstracción, independientemente de si son usados en tiempo de diseño o en tiempo de ejecución. Con el advenimiento del desarrollo dirigido por modelos, los modelos independientes de la plataforma han ganado una atención considerable.

2.5.1.4. Chequeo y Aplicación de la Consistencia y Conformidad.

La consistencia asegura que no existan contradicciones entre las diferentes partes de un sistema de software y los artefactos de desarrollo relacionados con el sistema. La conformidad asegura que un sistema de software reúna un estándar específico. Los requisitos de consistencia y conformidad viene desde diferentes clases de recursos, tales como consistencia o conformidad a los artefactos de diseño, combinaciones de características válidas, otros modelos (por ejemplo un modelo de sincronización) o políticas de integridad. Los modelos en tiempo de ejecución pueden ayudar a chequear o asegurar los requisitos de consistencia o conformidad, ya que ellos hacen disponible la información del modelo en tiempo de ejecución, ya sea acerca de las reglas de consistencia y conformidad o los artefactos del sistema a ser chequeados.

2.5.1.5. Comprobación y Aplicación de Políticas

La comprobación y aplicación de políticas abarcan los modelos usados en tiempo de ejecución para hacer frente a políticas, tales como restricciones en tiempo real, control de acceso y regulaciones de seguridad u otras reglas de cumplimiento. Varios modelos en tiempo de ejecución ayudan a satisfacer tales políticas: Modelos de seguridad describen propiedades de seguridad de los sistemas, modelos de políticas de control de acceso basados en roles y modelos en tiempo real soportan el modelado de restricciones de tiempo a ser preservados en tiempo de ejecución.

2.5.1.6. Manejo de Errores

Los modelos en tiempo de ejecución pueden ser usados para un manejo eficiente de control de errores en forma de depuración, localización de fallos, trazabilidad, auto-corrección y limitación de casos de prueba. De ahí, los problemas como "localización de errores", "predicciones imprecisas" y "ambientes cambiantes / heterogéneos". Modelos de máquina de estados y modelos de flujo de trabajo son usables para muchos de esos escenarios por su naturaleza expresiva en ejecución de flujos de datos, los cuales pueden fácilmente ser chequeados contra trazas de ejecución dadas, obtenidos por eventos emitidos o registros de ejecución.

2.5.2. Técnicas de los modelos en tiempo de ejecución

Las técnicas identificadas por Szvetits *et al. (*2013) que utilizan modelos en tiempo de ejecución son:

- Lazo de control autonómico: analizan, ejecutan el sistema y planifican acciones correctivas.

- Introspección: extraen datos desde un sistema en ejecución.
- Conformidad del modelo: asegura la conformidad y consistencia contra los modelos.
- Comparación: chequea diferencias entre dos modelos.
- Transformación de modelos: cambia la representación de los modelos.
- Ejecución de modelos: Ejecuta modelos con semántica operacional.

A continuación se describe cada una de las técnicas de los modelos en tiempo de ejecución con más detalle:

2.5.2.1. Lazo de Control Autonómico

Los lazos de control autonómico son conceptos clave cuando se adaptan sistemas en tiempo de ejecución (Oreizy et al., 1999). La Figura 2.7 ilustra la idea del lazo de control autonómico el cual es medir los parámetros del sistema, analizarlos, planificar acciones correctivas si es necesario y ejecutarlas. Este proceso es conocido como Monitorizar-Analizar-Planificar-Ejecutar (MAPE).

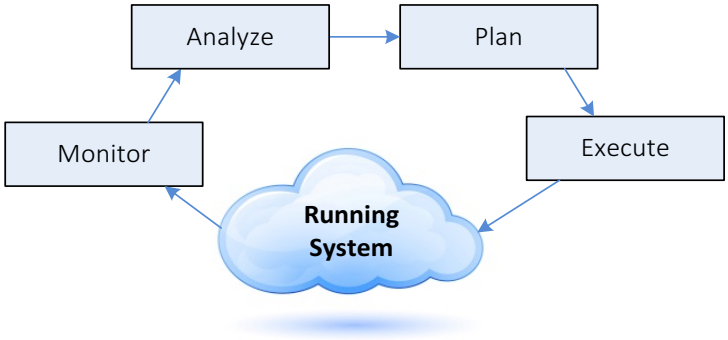


Figura 2-7. Idea detrás del lazo de control autonómico

El componente de planificación del lazo necesita identificar configuraciones alternativas que satisfagan los requisitos contextuales restricciones. Se usan modelos con este lazo de control como representación abstracta del sistema para reconfigurar aplicaciones. Este modelo evoluciona con el sistema y describe entidades y componentes de aplicaciones, ayudando a planificar acciones adecuadas de reconfiguración (Szvetits *et al.*, 2013). Un enfoque similar es introducido por Garlan *et al. (*2004) en donde su marco de trabajo utiliza modelos de arquitectura abstracta para monitorizar un sistema, chequear violaciones de restricciones y disparar una adaptación si es necesario.

2.5.2.2. Introspección

La introspección trata con los datos de un sistema en tiempo de ejecución para aplicar técnicas basadas en modelos y lazos de control sobre el comportamiento analizado (Szvetits *et al.*, 2013). Estos autores han dividido la introspección en:

- **Chequeo de los registros de eventos:** Realiza la extracción de datos escaneando las entradas de los registros de eventos emitidas por el sistema a observar. Tales registros tienen que seguir una estructura para ser procesados externamente y en consecuencia para ser relacionado a las representaciones en tiempo de ejecución. Se asume que los registros de eventos son consistentes con el sistema en ejecución y además no contradice las partes del modelo. En otras palabras se asume que los registros de eventos reflejan una representación correcta del sistema en ejecución. El reto de este método es identificar las partes relevantes del registro de eventos y habilitar la consistencia entre el sistema ejecutándose y el modelo.
- **Instrumentación:** En el caso de la instrumentación, la funcionalidad de monitorización está insertada en el sistema bajo observación. La inserción del código de monitorización requiere que el código fuente esté disponible y es a menudo realizado usando programación orientada a aspectos. La extracción de los datos está relacionada a las representaciones en tiempo de ejecución de los modelos para habilitar el razonamiento a un alto nivel de abstracción.
- **APIs de gestión:** Aquí existe una interface la cual permite la extracción de información de estado en tiempo de ejecución. Un ejemplo de tal interface es la reflexión. Las APIs de lenguajes de programación modernos permiten consultar y modificar la estructura de un programa de ejecución en tiempo de ejecución. Una arquitectura reflectiva consiste de dos capas causalmente contactadas: Una a nivel-base que se haga cargo del aspecto funcional del sistema, mientras que una a nivel-meta se haga cargo de los aspectos no funcionales del sistema y lleve a cabo la evolución sobre una reificación de la información de diseño.

2.5.2.3. Conformidad del Modelo

Direcciona la conformidad y consistencia de los datos y procesos de acuerdo a los modelos. La conformidad del modelo está estrechamente relacionado a los objetivos previamente descritos de consistencia y conformidad, chequeo y aseguramiento de políticas y manejo de errores. La información en tiempo de ejecución es chequeada contra los modelos para trazar violaciones de restricciones y comportamiento del sistema. La conformidad del modelo

requiere mecanismos de introspección en combinación con representaciones de los modelos en tiempo de ejecución.

2.5.2.4. Comparación de Modelos

Trata acerca de la comparación entre dos modelos, especialmente para producir información sobre sus diferencias y necesidad de operaciones para transformar un modelo en otro. Una herramienta que a menudo se referencia para esta transformación es EMF Compare (Eclipse Modelling Foundation), a través de una alternativa como KMF (Kevoree Modelling Framework). La comparación de modelos es principalmente usada para calcular operaciones necesarias para transiciones entre estados de un sistema.

2.5.2.5. Transformación de Modelos

La transformación de modelos es usada para cambiar las representaciones de un modelo fuente a otra forma llamada el modelo objetivo. Tales transformaciones tratan de reducir los errores a través de la automatización mientras aseguran la consistencia entre el modelo fuente y el modelo objeto. Estos son generalmente divididos en transformaciones Modelo a Texto M2T y Modelo a Transformación de Modelo (cuya salida es otro modelo).

2.5.2.6. Ejecución del Modelo

Los autores Szvetits *et al. (*2013) han incluido en su investigación la ejecución del modelo como una técnica separada ya que cambios en un modelo en tiempo de ejecución interpretados directamente implican cambios en el comportamiento del sistema en ejecución. Esta propiedad cae directamente bajo los conceptos de conexión causal. La ejecución de los modelos atiende directamente a la ejecución de modelos que contienen semánticas operacionales. En otras palabras, una representación sirve tanto como el modelo así como el programa los cuales están en el mismo nivel de abstracción. Un ejemplo popular de modelos ejecutables son las máquinas de estados.

2.5.3. Arquitecturas de los modelos en tiempo de ejecución

Procesar modelos en tiempo de ejecución requiere una arquitectura con capacidades de introspectiva para extraer la información relevante en tiempo de ejecución y para relacionar los datos recolectados con los modelos. Esto es muy similar a la reflexión donde el sistema bajo observación está habilitado para acceder a su propia estructura la cual está causalmente conectada a ésta (las modificaciones a su auto-representación son reflejadas en su propio estado y computación). Los modelos en tiempo de ejecución difieren de la reflexión principalmente en términos de abstracción: mientras la reflexión es más

orientada al espacio de la solución. Los modelos en tiempo de ejecución operan a un nivel más alto de abstracción hacia el espacio del problema (Blair *et al.*, 2009). Al nivel de modelo, las técnicas dirigidas por modelo pueden ser utilizadas en tiempo de ejecución, lo cual no es considerado en los enfoques tradicionales de enfoques dirigidos por modelos.

En el trabajo de Szvetits *et al. (*2013) los autores han identificado cinco principales tipos de arquitecturas cuando se procesan modelos en tiempo de ejecución.

- Arquitectura monolítica
- Arquitectura de flujo de datos local
- Arquitectura Middleware "*model-aware"*
- Arquitectura Middleware de comunicación
- Arquitecturas de repositorio

Estas arquitecturas difieren principalmente en términos de acceso al modelo y niveles de dinámicas tales como capacidades de adaptación o simulación.

2.5.3.1 Arquitectura Monolítica

En una arquitectura monolítica, la funcionalidad del sistema completo es capturada en una simple unidad sin separación en múltiples componentes del sistema. En caso de los modelos en tiempo de ejecución, se habla de arquitectura monolítica si un modelo en tiempo de ejecución es localmente accedido o manipulado por el sistema ejecutándose por sí mismo o si es directamente integrado en un componente de software. La pérdida de acoplamiento a los componentes del sistema conlleva a un diseño simple del sistema, pero hace que se pierda adaptabilidad, escalabilidad y reusabilidad. De ahí, esta arquitectura es usada a menudo en combinación con ejecución directa de modelos o interpretación, cuando aspectos dinámicos como adaptación, simulación y depuración pueden ser rechazados.

2.5.3.2. Arquitectura de Flujo de Datos Local

Las arquitecturas de flujo de datos local son usadas para descomponer una tarea local en varias sub-tareas con complejidad inferior, entonces se compensan las desventajas de las arquitecturas monolíticas introducidas por la pérdida de acoplamiento. Los flujos de datos entre las sub-tareas locales pueden ser implementados por mecanismos de comunicación de inter-procesos e in-procesos tales como *sockets* y *threads*, respectivamente.

2.5.3.3. Arquitectura Middleware "model-aware"

Las arquitecturas middleware habilitan la comunicación distribuida y proveen funcionalidad adicional, tal como, mejorar el control monitorización y registro. Una arquitectura de middleware "model-aware" consiste en una entidad central responsable de procesar la información del modelo y proveer acceso a los servicios con un sistema distribuido. Tal arquitectura ayuda a mantener los mecanismos de monitorización y adaptación de manera centralizada, incrementando la reusabilidad mientras se compensa las limitaciones de los límites de las máquinas locales en arquitecturas monolíticas y de flujos de datos.

2.5.3.4. Arquitectura Middleware de Comunicación

Un middleware de comunicación se centra en abstraer la complejidad de una red de comunicaciones entre componentes distribuidos. El uso de servicios es bien establecido en combinación con los modelos en tiempo de ejecución. Muchos enfoques basados en middleware utilizan servicios como métodos de comunicación primaria entre componentes distribuidos.

2.5.3.5. Arquitectura de Repositorio

Los enfoques basados en repositorios usan un almacenamiento central de modelos o datos de modelos relacionados. La ventaja de los enfoques basados en repositorios cae en el acceso controlado a las instancias de modelos como también al almacenamiento centralizado de la información de evolución de modelos. Los repositorios son usados para establecer una arquitectura distribuida y un fácil acceso y manejo de los datos del modelo proveyendo interfaces estandarizadas para la manipulación de los mismos. Las actualizaciones de los repositorios pueden ser ya sean aplicadas por acciones manuales u otros sistemas los cuales tienen acceso de escritura al repositorio.

2.6. Monitorización en la nube

La monitorización en la nube es necesaria para medir continuamente y evaluar la infraestructura o el comportamiento de la aplicación en términos de rendimiento, confiabilidad, uso de recursos, habilidad para cumplir los SLA, seguridad, etc. Además, se puede utilizar también para análisis de los negocios, para mejorar la operación de los sistemas y aplicaciones y otras actividades. Aceto *et al.* (2013) proveen una revisión de la literatura acerca de la monitorización de servicios en la nube e introducen una serie de conceptos en base a la monitorización, los mismos que serán presentados en esta sub-sección con la finalidad de describir conceptos claves que serán utilizados a lo largo de

este trabajo de investigación. Estos autores reportan esos conceptos en base a una taxonomía en la que proponen las principales características de la monitorización de servicios en la nube entre las cuales están la escalabilidad, elasticidad, confiabilidad, disponibilidad, adaptabilidad autonomía, precisión, entre otras.

De acuerdo al trabajo de Cloud Security Alliance, una nube puede ser modelada en siete capas:

- *Instalación*: a esta capa se considera la infraestructura física que comprende los centros de datos que hospedan el equipo de cómputo y red.
- *Red*: a esta capa se considera las redes y rutas ambos en la red y entre la nube y el usuario.
- *Hardware*: a esta capa se considera el componente físico de la computación y la red.
- *Sistema Operativo*: en esta capa se ubican los componentes de software que forman el sistema operativo, tanto del host (el sistema operativo que se ejecuta sobre la máquina física) y el usuario (el sistema operativo que se ejecuta en la máquina virtual).
- *Middleware*: en esta capa se ubica la capa de software entre el sistema operativo y la aplicación del usuario. Esta es típicamente presentada solamente por los sistemas en la nube que ofrecen SaaS y PaaS.
- *Aplicación*: En esta capa se consideran las aplicaciones que se ejecutan por el usuario y el sistema en la nube.
- *Usuario*: a esta capa consideramos al usuario final del sistema en la nube y las aplicaciones que se ejecutan fuera de la nube (por ejemplo, un navegador web)

En el contexto de la monitorización cloud, estas capas pueden ser vistas como el lugar donde se deben ubicar los sistemas de monitorización. De hecho, la capa en la cual se ubica el sistema de monitorización tiene consecuencias directas sobre el fenómeno que quiere ser observado (Aceto *et al.*, 2013).

Además, debido a la complejidad tan alta de los sistemas en la nube, no es posible estar seguro de que cierto fenómeno será observado o no. Por ejemplo, si ponemos exploramos en una aplicación que se ejecuta en la nube, para recolectar información sobre la tasa a la cual esta intercambia información con otras aplicaciones ejecutándose en la misma nube, no necesariamente conoceremos si esa tasa comprende además la tasa de transferencia de la red.

Esto depende si las dos aplicaciones se ejecutan en el mismo host o no, y ésta información no siempre está expuesta por el proveedor. Se pueden observar

problemas similares al evaluar el rendimiento del cómputo: el tiempo requerido para terminar una tarea puede depender del hardware actual que si está ejecutando las instrucciones y sobre la carga de trabajo debido a otros ambientes virtualizados ejecutándose en el mismo servidor virtual (el cual no se expone del todo al consumidor) (Aceto *et al.*, 2013).

2.6.1 Niveles de abstracción

En computación en la nube, se pueden distinguir monitorización en alto y bajo nivel, y los dos son requeridos (Caron *et al.*, 2012). La monitorización de alto nivel está relacionada a la información sobre el estado de la plataforma virtual. Esta información es recolectada en el middleware, aplicación y capas de usuario por proveedores o consumidores a través de plataformas y servicios operados por ellos mismos o terceras partes. En el caso de SaaS, la información de monitorización de alto nivel es generalmente de mayor interés para el cliente que para el proveedor (muy relacionado a la experiencia de calidad de servicio del primero). Por otro lado, la monitorización de bajo nivel está relacionada con la información recolectada por el proveedor y usualmente no expuesta al cliente, y esta es mayormente centrada en el estado de la infraestructura física de la nube (por ejemplo, servidores y almacenamiento, etc.). En el contexto de IaaS, ambos niveles son de interés para clientes y proveedores (Aceto *et al.*, 2013).

2.6.2. Pruebas y métricas

Las pruebas de monitorización están divididas en dos categorías principales: i) basadas en computación y ii) basadas en red. Las pruebas basadas en computación son relacionadas a las actividades de monitorización que permiten ganar conocimiento sobre el estado de inferir el estado de las plataformas reales o virtualizados que ejecutan aplicaciones en la nube. Las pruebas son relacionadas a métricas tales como rendimiento, definido como el número de peticiones por segundo, velocidad del CPU; tiempo del CPU por ejecución, etc. Mientras que las pruebas basadas en red están relacionadas a la monitorización dela capa de red. Este conjunto incluye el tiempo de ida y regreso (RTT), fluctuación, pérdida de paquetes, ancho de banda disponible, capacidad, etc. (Aceto *et al.*, 2013).

2.6.3. Monitorización grid vs. clúster vs. cloud

Las similitudes y superposiciones de propiedades entre la computación en la nube y paradigmas previos han dado lugar a una discusión sobre la definición de computación en la nube y sus características específicas. Aceto *et al. (*2013) han considerado las principales diferencias desde el punto de vista de la monitorización.

Cuando comparada con la computación grid, la monitorización en la nube es más compleja debido a las diferencias en cuanto al modelo de confianza y a la vista de recursos/servicios presentados al usuario. De hecho, el principal objetivo de una grid es la compartición de recursos a través de múltiples organizaciones, implicando criterios simples y limitados de abstracción de recursos los cuales crean una simple relación entre los parámetros de monitorización y estado de recursos físicos. En la otra mano, para la nube, la presencia de múltiples capas y paradigmas de servicios conllevan a una abstracción alta de recursos, resultando en relaciones más opacas entre la capa o servicios observables y los recursos subyacentes. Además la computación en la nube, incluso si las interfaces abstractas ofrecidas al consumidor pueden aparentemente requerir una necesidad de recursos reducidas para monitorización con respecto a grid, en realidad tal necesidad es adjudicada al proveedor del servicio, que tiene que cumplir con promesas o rendimiento esperado y con la optimización de recursos en un escenario altamente dinámico y heterogéneo.

Esta brecha de objetivos y transparencia debe ser llenada cuando se adopta un sistema de computación en la nube desde el campo de computación grid. Finalmente el paradigma de servicios "bajo demanda" supone retos adicionales a sistemas de monitorización no diseñados para un alto número de usuarios y recursos.

Todas esas diferencias son hasta más acentuadas cuando comparamos el paradigma de la nube a los clústeres; en este caso, la arquitectura relativamente rígida, las capacidades limitadas de negociación de servicio y la baja automatización de recursos hacen a los clústeres comparados con una tecnología base para proveedores IaaS y conllevan a requisitos en términos de monitorización que son un conjunto limitado de una nube. De ahí la mayoría de propiedades caracterizadas por sistemas de monitorización en la nube no aplican a clústeres o grids (elasticidad, adaptabilidad, autonomía) o no son vitales para su propósito (facilidad de comprensión, extensibilidad e intrusividad).

2.6.4. Monitorización cloud: propiedades y problemas relacionados

A fin de operar propiamente, un sistema de monitorización distribuido se requiere que tenga diferentes propiedades de cuando son consideradas en el escenario de computación en la nube introducen nuevos retos.

Las herramientas de monitorización en la nube, se esperan que tengan ciertas características de la calidad que les permitan cumplir con su propósito de la manera más adecuada tanto para proveedores como para clientes, el trabajo de Aceto *et al. (*2013), el mismo que hemos venido citando a lo largo de este tema,

trata de una investigación sobre las herramientas de computación en la nube y establece una taxonomía de las características más deseadas en este tipo de herramientas. Éstas serán descritas a lo largo de esta sección:

2.6.4.1. Escalabilidad

Un sistema de monitorización es escalable si puede hacer frente a un largo número de exploraciones. Tal propiedad es muy importante en escenarios de computación en la nube debido al largo número de parámetros a ser monitorizados sobre un inmenso número de recursos. Esta importancia es amplificada por la adopción de tecnologías de virtualización, las cuales permiten ubicar muchos recursos virtuales en la cima de un simple recurso físico. Las medidas requeridas para obtener una vista comprensible del estado de la nube conllevan a la generación un amplio volumen de datos que vienen desde múltiples localizaciones distribuidas. De ahí, un sistema de monitorización escalable debe ser habilitado para recolectar eficientemente, transferir y analizar tal volumen de datos sin impedir la operación normal de la nube.

En la literatura tal problema ha sido principalmente direccionado proponiendo arquitecturas en las cuales los datos monitorizados y eventos son propagados al control de la aplicación después de su agregación y filtro, para reducir su volumen: la agregación combina múltiples métricas en una síntesis que es inferida o no directamente; el filtrado evita datos inútiles que puedan ser propagados al control de la aplicación. Existen también propuestas que mejoran la escalabilidad adoptando optimizaciones adicionales, algoritmos eficientes para el desarrollo de agentes e interconexión.

2.6.4.2. Elasticidad

Un sistema de monitorización es elástico si este puede hacer frente a cambios dinámicos de entidades de monitorización, esto es recursos virtuales creados y liberados por expansión y contracción y que estos puedan ser monitorizados correctamente. Esta propiedad además se refiere al dinamismo, e implica escalabilidad al añadir o quitar recursos a ser monitorizados.

Al contrario de los paradigmas previos de computación (p. ej., computación grid), la computación en la nube requiere que sus recursos sean dinámicos, haciendo de la elasticidad una propiedad esencial del sistema de monitorización desde tres principales directrices: variación de la asignación de recursos a usuarios, variación de requisitos de monitorización para el usuario y variación de la presencia de usuarios (escenarios multitenencia). Un reto en proveer la elasticidad está relacionado con el hecho de que esta es una nueva propiedad

fundamental introducida por la monitorización en la nube y no considerada previamente como un requisito para monitorizar sistemas distribuidos genéricos.

2.6.4.3. Adaptabilidad

Un sistema de monitorización es adaptable si este se puede adaptar a varias cargas computacionales y de red sin ser invasivo.

Debido a la complejidad y al dinamismo de los escenarios en la nube, la adaptabilidad es fundamental para un sistema de monitorización para impedir tanto como sea posible un impacto negativo de las actividades de monitorización sobre las operaciones normales de la nube, especialmente cuando mediciones activas son involucradas.

De hecho, la carga de trabajo generada por mediciones activas con la colección, procesamiento transmisión y almacenamiento de datos de monitorización y el manejo del subsistema de monitorización requieren recursos de cómputo y comunicación y de ahí constituyen un costo para la infraestructura en la nube. Entonces, la habilidad de cambiar las monitorizaciones de monitorización de acuerdo a políticas apropiadas es de significativa importancia para reunir las metas de manejo de la nube. Proveer adaptabilidad no es trivial, porque esto requiere de una reacción rápida a cambios, manteniendo un correcto balanceo entre precisión e invasividad.

2.6.4.4. Oportunidad

Un sistema de monitorización es oportuno si los eventos detectados son detectados a tiempo para su uso.

La monitorización es instrumental a actividades relacionadas con metas clave de un consumidor o un proveedor, de ahí es necesario contar con información oportuna para la respuesta adecuada (p. ej., para levantar una alarma, para proveer más recursos, para migrar servicios, para hacer cumplir diferentes políticas).

La oportunidad es independiente de otras propiedades de la monitorización tales como la elasticidad, autonomía y adaptabilidad. Por tanto, esto implica los mismos retos o compensaciones entre requisitos opuestos. Más en detalle, el tiempo entre la ocurrencia de un evento y su notificación puede ser descompuesto en distintas contribuciones: muestra, análisis y retraso de comunicación.

Cada una de ellas supone algunos problemas. El intervalo más pequeño de muestreo, es el retraso entre el tiempo una condición sucede y es capturada. Entonces, para obtener información actualizada una compensación entre

precisión y frecuencia de muestreo es necesaria, considerando además las restricciones de recurso (p. ej. CPU, red, ancho de banda o memoria).

El retraso del análisis supone un problema similar de acuerdo a los eventos complejos (el resultado de un cómputo sobre múltiples parámetros), los cuales requieren considerar además el tiempo para tomar toda la información necesaria, además del tiempo de cómputo.

Finalmente, siendo la nube un sistema distribuido, el retraso de la comunicación puede ser significante porque la información puede tener que viajar a través de múltiples vínculos para alcanzar los nodos de procesamiento y este retraso es hasta más importante cuando se consideran eventos complejos que envuelven información recibida desde fuentes remotas.

2.6.4.5. Autonomía

Un sistema de monitorización autonómico es habilitado para auto-manejar sus recursos distribuidos por medio de la reacción automática a cambios impredecibles, mientras se esconde la complejidad a proveedores y consumidores (Ostermann *et al.*, 2010).

Como las infraestructuras en la nube han sido concebidas para proveer servicio bajo demanda una rápida elasticidad mientras operan continuamente con interrupciones mínimas de servicio, es extremadamente importante para el sistema de monitorización reaccionar a cambios detectados, fallos y degradación de rendimiento sin intervención manual. Soportar autonomía en el sistema de monitorización, no es una tarea trivial, ya que requiere implementar un bucle de control que reciba las entradas desde un alto número de sensores (datos de monitorización) y propagar acciones de control a un largo número de actuadores distribuidos. Esto en cambio implica elasticidad y oportunidad. Además, las capacidades de análisis para esta situación deben ser implementadas, así como la definición de políticas adecuadas para dirigir el comportamiento del sistema de monitorización en respuesta a los eventos detectados es algo necesario.

2.6.4.6. Comprensibilidad, Extensibilidad e Intrusividad

Un sistema de monitorización es comprensible si este soporta diferentes tipos de recursos (físicos y virtuales), algunos tipos de datos de monitorización y multitenencia (Hasselmeyer *et al.*, 2010); es extensible si puede ser fácilmente extendido (por ejemplo a través de plugins o modelos funcionales); es intrusivo si su adopción requiere una modificación significativa de la nube (Katsaros *et al.*, 2011).

Las primeras dos propiedades están estrictamente relacionadas; la última representa la posibilidad de mejorar la primera sin modificar el marco de trabajo de la monitorización. Tener un sistema de monitorización comprensible es útil tanto para desarrolladores (consumidores IaaS y PaaS) y sus respectivos proveedores.

La ventaja del primero está relacionada a la posibilidad de adoptar un API de monitorización, independientemente de qué clase de información de monitorización será usada. Para el último, la ventaja consiste en desplegar y mantener solamente una infraestructura de monitorización única. Además proveer extensibilidad, tales ventajas pueden fácilmente persistir a cambios o adiciones de componentes y mantener baja intrusividad permitiendo minimizar los costes de implementación.

La computación en la nube es un paradigma relativamente nuevo y no existen estándares comunes que hayan sido adoptados por sistemas desplegados. Algunos problemas se dan cuando se considera la comprensibilidad, un primer problema está relacionado al hecho de soportar diferentes arquitecturas, tecnologías y recursos, mientras se preserva el aislamiento entre diferentes huéspedes. Por otro lado, un sistema de monitorización comprensible permite una mejor ejecución de las actividades, las cuales también pueden presentar problemas en debido al dinamismo de los ambientes de computación en la nube y a la alta heterogeneidad de recursos y parámetros considerados a diferentes capas.

2.6.4.7. Resistencia, Confiabilidad y Disponibilidad

Un sistema de monitorización es *resistente* cuando la persistencia de la entrega de servicios es confiable ante cambios, que básicamente significa resistente a un número de fallos de componentes mientras continúa con su operación normal; es *confiable* cuando puede ejecutar una función requerida bajo condiciones establecidas por un período específico de tiempo; y es *disponible* si este provee servicios de acuerdo al diseño del sistema cuando el usuario los requiere (Shirey, 2007).

Como la monitorización es funcional a actividades críticas de la nube tales como facturación, verificación del cumplimiento del SLA y manejo de recursos, los sistemas de monitorización deben ser resistentes, confiables y disponibles para no comprometer tales actividades. Con el uso intenso de tecnologías de virtualización por plataformas en la nube, los hosts monitorizados y servicios pueden migrar desde un ordenador físico a otro, invalidando la monitorización clásica lógica y bajando la confiabilidad del sistema de monitorización. De ahí es necesario brindar ciertas características para la computación en la nube tales

como trazabilidad y manejo de monitorización heterogénea y recursos de monitorización, caracterizando posibles fallos de los sistemas de monitorización y protegiéndolos contra ellos.

2.6.4.8. Precisión

Nosotros consideramos que un sistema de monitorización es preciso cuando las medidas provistas por éste son precisas, esto es cuando son tan cercanas como sean posible a los valores reales a ser medidos.

La precisión es importante para cualquier sistema de monitorización distribuida porque este tiene un alto impacto en las actividades que hacen uso de los valores monitorizados, por ejemplo, cuando el sistema de monitorización es usado para resolver problemas serios de imprecisión en las medidas puede acarrear una incorrecta identificación de la causa del problema).

En el contexto de computación en la nube la precisión se vuelve incluso más importante. En primer lugar, porque los servicios en la nube son sujetos a SLA bien definidos y los proveedores tienen que pagar penalizaciones a sus clientes en caso de las violaciones del SLA, monitorización imprecisa puede acarrear a pérdida de dinero. Segundo, dado que un sistema de monitorización es usado para actividades importantes de la nube, la precisión en la monitorización es necesaria para efectiva y eficientemente ejecutarlas.

El análisis de la literatura propuesto por Aceto *et al.* (2013) ha relevado dos problemas principales relacionados a la precisión de los sistemas de monitorización en escenarios de computación en la nube. El primero y dado que los servicios cloud están relacionados a SLAs bien definidos en donde los proveedores tienen que pagar penalización en caso de violaciones, las imprecisiones en la monitorización pueden conllevar a pérdidas de dinero. El segundo está relacionado a la monitorización en sí, dado que si ésta no es precisa mal se podrían hacer mejoras sobre los servicios.

2.7. Conclusiones

Este capítulo ha introducido los principales conceptos relacionados con éste trabajo de investigación. Se ha abordado la computación en la nube, sus características esenciales, modelos de servicio y despliegue, con la finalidad de contextualizar a qué tecnología se referirá esta tesis, así como también sus particularidades y conceptos.

Por otro lado, se han discutido temas relacionados con la calidad de los servicios cloud, los modelos de calidad existentes y los estándares más utilizados

actualmente, teniendo así en este capítulo, un resumen del estándar ISO 25000, las subdivisiones que éste contiene y la manera en la que se descomponen las características de calidad de manera general.

Luego se ha discutido el rol de los SLA, las partes constitutivas de los mismos y la importancia que éstos tienen al momento de pactar las características no-funcionales entre proveedores y clientes de servicios en la nube.

A continuación, se ha discutido las particularidades de la Ingeniería de Software Dirigida por Modelos, previo a la conceptualización y detalle de los *modelos en tiempo de ejecución*, cuya utilización representa el aporte de éste trabajo. Dentro de los modelos en tiempo de ejecución se especifican sus objetivos, las técnicas que éstos emplean, así como también las arquitecturas de los mismos.

Finalmente, éste capítulo se centra en los conceptos de la monitorización de servicios en la nube, sus niveles de abstracción, pruebas, las características más importantes que una solución de monitorización en la nube debe proveer y una breve explicación de las diferencias entre los sistemas de monitorización de propósito general versus monitorización de servicios en la nube.

Como se puede observar, por tanto, en este capítulo, se tratan los conceptos fundamentales que serán de utilidad a lo largo de este trabajo y que permitirán al lector tener una clara idea de cada una de las tecnologías que intervienen en la solución aquí presentada.

Capítulo 3
Estado del Arte

En este capítulo se analizan los métodos, técnicas y enfoques existentes de evaluación de calidad de servicios en la nube, comprobación del cumplimiento de los SLA y monitorización de la calidad de servicios en la nube. Además, se tienen en cuenta soluciones que utilizan la Ingeniería de Software Dirigida por Modelos así como también modelos en tiempo de ejecución.

En la sección 3.1 se analizan los modelos de calidad existentes para servicios en la nube desplegados con el modelo *Software as a Service*, así como también la manera en la que la calidad es abordada en arquitecturas que comparten características comunes como son las arquitecturas orientadas a servicios y las de computación distribuida.

En la sección 3.2, se abordan los métodos de apoyo al cumplimiento de los acuerdos de nivel de servicio (SLA), que al igual que en la sección anterior, se han analizado tanto para arquitecturas orientadas a servicio en general como para la computación en la nube. Esta sección, por tanto, nos permitirá analizar la forma en la que la comprobación del cumplimiento de los acuerdos de nivel de servicio, está siendo abordada por los diferentes enfoques y propuestas existentes.

En la sección 3.3, se describen los métodos de monitorización de calidad de servicios existentes, tanto aquellos de propósito general en donde se abarcan características comunes a los sistemas de monitorización, como aquellos en los cuales se contemplan características específicas de la nube.

Finalmente, en la sección 3.4 se describen las soluciones de monitorización que utilizan modelos en tiempo de ejecución, y en general se analiza cómo ésta tecnología ha sido aplicada para solucionar problemas similares.

3.1. Modelos de calidad de servicio

La calidad debería ser definida y medida si se quieren conseguir mejoras en los productos software. Muchas veces el término calidad es entendido de una manera ambigua. La confusión puede ser atribuida a varias razones. Primero, la calidad no es una idea simple sino un concepto multidimensional. Las

dimensiones de calidad incluyen la entidad de interés, el punto de vista de la entidad y los atributos de calidad de la entidad en sí (Kan, 2003).

Teniendo en cuenta lo dicho, existen numerosas propuestas que presentan modelos de calidad para distintos tipos de software, éstos son construidos dependiendo de las características intrínsecas del software y/o otras características operacionales al que quieren evaluar. Este es el caso de las propuestas de Olsina *et al.* (2012) en la que los autores proponen un modelo de calidad para aplicaciones web y la propuesta de Zeng *et al.* (2004) que plantea un modelos de calidad para *grid computing*.

Otros ejemplos son los modelos de calidad para mashups, como es el caso del trabajo presentado por Cappiello *et al.* (2011) en el que los autores identifican un conjunto de dimensiones de calidad y métricas para medir la calidad de los componentes de un mashup o el trabajo de Insfran *et al.* (2012) en el cual se propone un modelo de calidad centrado en la usabilidad de un mashup.

Se pueden nombrar también modelos de calidad específicos para la computación en la nube como es el caso del trabajo propuesto por Lee *et al.* (2009) o modelos de calidad para servicios web como los mencionados en la revisión de la literatura realizada por Oriol *et al.* (2014).

Como se puede apreciar, existe mucha investigación en cuanto al desarrollo de modelos de calidad según el tipo de software que se quiera evaluar; además, es importante contar con modelos de calidad apropiados para la evaluación de la calidad; estos modelos de calidad deben considerar las características intrínsecas del software que se esté evaluando, sus atributos y las métricas que se pueden emplear para realizar las mediciones de los atributos.

Los modelos de calidad proporcionan una definición operacional de la calidad mediante la definición de las características que influencian la calidad del software y proveen también las bases para la evaluación del software. Sin embargo, para hacer uso de un modelo de calidad en un dominio específico, se debe extenderlo e incluir las particularidades de cada tipo de software (Radulovic *et al.*, 2011).

Con la finalidad de crear modelos de calidad para diferentes tipos de sistemas de software, se hace uso de estándares de calidad existentes como por ejemplo el ISO/IEC 9126 (2001) que ha sido extendido por Radulovic *et al.* (2011) para especificar un modelo de calidad para tecnologías semánticas; o el estándar ISO/IEC 25010 (2011) que ha sido usado como base para el modelo de calidad para mashups propuesto por Insfran *et al.* (2012).

En los estándares ISO antes mencionados, un modelo de calidad es una descomposición jerárquica de características y subcaracterísticas de calidad. Existen varias críticas que hablan sobre la ambigüedad de la descomposición de las características de calidad, además puntualizan que a menudo ciertas características no son lo suficientemente específicas para poder medirlas directamente. El estándar ISO/IEC 25010 presenta algunas mejoras, incluyendo un modelo de medición de referencia aunque a pesar de ello las críticas siguen siendo válidas debido a que las mediciones aún son insuficientes. Además, un estudio realizado por Wagner (2013) muestra que un 28% de las empresas usan modelos de calidad estándar mientras que el 71% han desarrollado sus propias variantes. De ahí la necesidad de la personalización de los modelos de calidad.

A continuación, se discuten los modelos de calidad y otras estrategias de aseguramiento de calidad que han sido propuestas para arquitecturas orientadas a servicios, arquitecturas de computación distribuidas y la computación en la nube.

3.1.1. Calidad en arquitecturas orientadas a servicios

La calidad de servicio (QoS) es un tema de interés en actividades relacionadas con los servicios web. Los modelos de calidad han sido propuestos como artefactos de ingeniería para proveer un marco de trabajo común para el entendimiento de la calidad, definiendo factores de calidad que pueden ser aplicados al uso de servicios web y en arquitecturas orientadas a servicios en general (Oriol *et al.*, 2014).

En el trabajo presentado por Oriol *et al.* (2014) se han resumido varias propuestas de modelos de calidad, por medio de un mapeo sistemático. En este trabajo, los autores han analizado los diferentes modelos de calidad definidos y de esta manera, han evaluado el estado del arte de los modelos de calidad para arquitecturas orientadas a servicios y servicios web en general, identificando sus fortalezas y debilidades.

Los servicios web traen consigo características de calidad distintas a las presentadas por otro tipo de software, tal es el caso de la facilidad en la instalación, descrita en el estándar ISO 25010, y que no puede aplicarse dentro de este dominio debido a que los servicios web son ejecutados de forma remota. En este mapeo sistemático, los autores han identificado 65 trabajos. Estos trabajos, presentan 47 diferentes propuestas de modelos de calidad (ciertos modelos de calidad son explicados en más de un trabajo) y han podido abstraer los resultados presentados según cada característica de calidad para lo cual se han

basado en el estándar ISO/IEC 25010. En esta revisión se han incluido los modelos propuestos desde el año 2001.

Dentro de los hallazgos, los autores señalan que el primer modelo de calidad diseñado específicamente para servicios web fue presentado en el año 2002. Además, contrario a lo que se esperaba, la mayoría de propuestas no toman en consideración los estándares ISO 9126-1 o ISO/IEC 25010 para el desarrollo del modelo de calidad, con excepciones del modelo de calidad S-Cube (Gehlert *et al.*, 2008) y WSQM (Kim *et al.*, 2012), además de los propuestos por Abramowicz *et al. (*2008), Kritikos *et al. (*2010), Oriol *et al.* (2010). Por otro lado, el 56% de los enfoques no han considerado otros modelos de calidad para servicios web en su definición. El 26% de los enfoques han influenciado la definición de otros modelos de calidad. Si se centran propuestas particulares, los modelos más antiguos de calidad han tenido el impacto más alto.

El modelo de calidad de servicio más usado es el de Ran (2003) que descompone la calidad de los servicios web en características agrupadas por tiempo de ejecución, soporte transaccional, manejo de configuración, costo y seguridad. Las características en tiempo de ejecución son: escalabilidad, capacidad, rendimiento, confiabilidad, disponibilidad, robustez, flexibilidad, manejo de excepciones y precisión. Por otro lado la integridad es la característica relacionada con el soporte transaccional. Por otra parte, las características relacionadas al manejo de la configuración y costos relacionados son: las características regulatorias, el soporte a estándares, la estabilidad, el costo y la completitud y. Finalmente, las características de calidad de servicio relacionadas a la seguridad que el autor propone son: autenticación, autorización, confidencialidad, trazabilidad y auditoría, responsabilidad, encriptación de datos y no repudio. Este modelo ha sido utilizado para desarrollar ocho modelos de calidad, seguido por el modelo de calidad de Maximilien *et al. (*2004) que ha influenciado seis modelos de calidad y tanto el propuesto por IBM (Mani *et al.*, 2002) y W3C (Lee *et al.*, 2003) que han influenciado cinco modelos de calidad cada uno. Por otro lado, consideran el máximo número de trabajos que un modelo de calidad ha tenido en cuenta, el mismo que ha sido desarrollado por Frutos *et al.* (2009) con seis propuestas y WSMO-QoS (Wang *et al.*, 2006), Comuzzi *et al. (*2013) y Yin *et al.* (2010) con cuatro propuestas cada uno. El enfoque que refleja la evolución más larga es el de Yin *et al.* (2010), cuya evolución la realiza a través de siete modelos de calidad desarrollados desde el año 2003.

Finalmente, los autores han encontrado como dato de interés que los primeros modelos de calidad fueron desarrollados por organizaciones tales como IBM (Mani *et al.*, 2002) y W3C (Lee *et al.*, 2003). Modelos de calidad más

recientes tales como WSMO son desarrollados por el *WSMO Working Group* y soportados por W3C (Bruijn *et al.*, 2007) (Bruijn *et al.*, 2005), el grupo OASIS ha desarrollado además un modelo de calidad llamado WSQM (Kim *et al.*, 2005) (Kim *et al.*, 2012). Por otro lado, algunos modelos han sido desarrollados por redes europeas tales como S-Cube o proyectos europeos como BREIN (Frutos *et al.*, 2009).

Por otro lado, las características propuestas en los modelos de calidad fueron clasificadas en el mapeo sistemático de Oriol et al. (2014) cuyos resultados se muestran en la Figura 3-1, de acuerdo a su tamaño (cantidad) y cobertura en la definición (calidad). Con respecto al tamaño, los autores tuvieron en cuenta la cantidad de nodos presentados en los modelos de calidad y su nivel de profundidad. A través de este criterio, el modelo de calidad más extenso es el presentado en el modelo de referencia de calidad S-Cube (Gehlert et al., 2008) con 66 nodos seguido por GESSI (Ameller et al., 2008; Cabrera et al., 2012; Oriol et al., 2008) con 57 y (Truong et al., 2006) con 45 nodos. Por otro lado, y de acuerdo a su cobertura en la definición o completitud una amplia mayoría de las propuestas estudiadas, tienen una única y consistente definición de todos los factores de calidad que son presentados en el modelo de calidad, ya sea explícitamente definiendo la calidad del factor en el artículo o referenciando el artículo al que contenga una definición apropiada.

Como conclusiones de la revisión sistemática, se concluyó que los modelos de calidad para servicios web han constituido un importante campo de investigación desde 2001 hasta la fecha de publicación de la revisión sistemática, para lo cual los autores han distribuido cronológicamente las 47 propuestas, han mostrado las relaciones entre ellas e identificado cuáles se han consolidado, cuáles son las más influyentes y cuáles han sido más influenciadas. Además, han concluido que el tamaño de los modelos de calidad varía de 6 a 66 nodos (con un promedio de 24,38) y el número de niveles entre 1 y 3 (con un promedio de 2,23). Luego han concluido que una amplia mayoría de propuestas (51%) tienen una definición única y consistente para todos los factores de calidad.

Por otro lado, han concluido que en primer lugar la "confiabilidad" y luego la "seguridad", el "rendimiento" y la "eficiencia" son las características explícitamente definidas en al menos la mitad de las propuestas estudiadas. Con respecto a las sub-características las definidas en al menos la mitad de los estudios son: la "adecuación funcional" y la "disponibilidad". Además, el "comportamiento en el tiempo", la "capacidad" y la "confidencialidad" son consideradas también.

Por último los autores obtuvieron que hasta 19 atributos de calidad aparecen en al menos el 30% de los modelos estudiados. Entre ellos la "disponibilidad" que es la más usada, seguida del "tiempo de respuesta", el "rendimiento", el "costo" y la "precisión". De todo esto llegan a la conclusión de que la vista panorámica realizada sobre la anatomía de los modelos de calidad para servicios web puede ser una buena referencia para investigadores del área, especialmente con miras de evitar nuevas clasificaciones o definiciones que puedan ser contrarias a las prácticas establecidas.

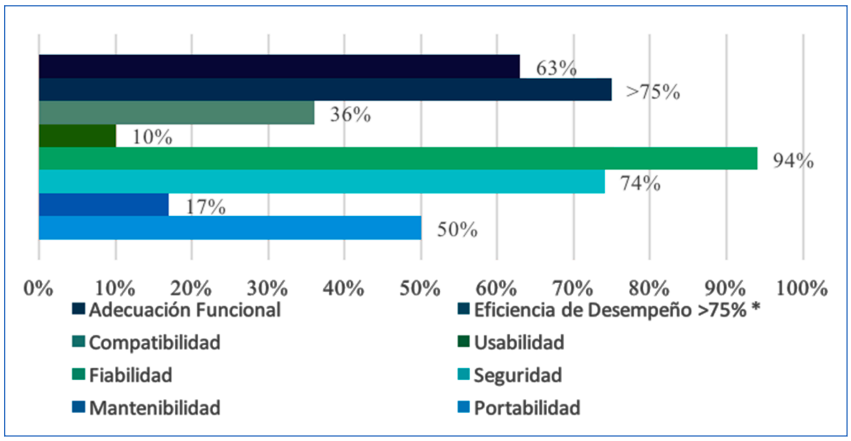


Figura 3-1. Características de Calidad Incluidos en Oriol *et al.* (2014)

Como se puede ver de todo lo anteriormente citado, dentro de los factores mayormente abordados en de las definiciones de los modelos de calidad los autores han utilizado mayoritariamente el estándar ISO/IEC 25010 (2011) como un marco de referencia para la comparación.

3.1.2. Calidad en arquitecturas de computación distribuida

La computación distribuida trata de una colección de computadoras ordenadas físicamente y conectadas entre sí por una red de comunicaciones; cada computadora posee sus componentes de hardware y software que el programador percibe como un solo sistema. Dentro de la computación distribuida están los sistemas de *Grid Computing*, en la cual todos los recursos de un número indeterminado de computadoras son englobados para ser tratados como un único superordenador de forma transparente.

Este tipo de computación comprende varias clases de recursos dinámicos y heterogéneos y provee a los usuarios servicios transparentes. Los cuales deben al igual que en el resto de paradigmas de computación ofrecer altos niveles de calidad de servicio. Varios estudios al respecto se están llevando a cabo, tal es el

caso de Zeng *et al.* (2004). En cuyo trabajo los autores ofrecen un modelo de calidad de servicio para *Grid Computing* basado en el protocolo DiffServ de servicios diferenciados. Por otro lado (Menascé *et al.,* 2004) proveen un estudio de la calidad de servicio para Grid Computing, en el cual ellos proveen un ejemplo que motiva el uso de Grid Computing en una organización y luego discuten la manera en la que la ubicación de los recursos puede afectar el SLA.

En el trabajo de Moraga *et al.* (2002), los autores han desarrollado un modelo de calidad para un portal Grid desde un modelo de calidad existente, llamado *Product Quality Management (PQM)* (Quality, Challenge, 2009) y lo han aplicado a dos portales Grid. Ellos recalcan la dificultad envuelta en usar recursos Grid debido a su arquitectura compleja. Los portales Grid aparecen debido a la necesidad de acceder de manera fácil a recursos Grid. En su trabajo han adaptado ciertas dimensiones de calidad a los portales Grid tales como la "adaptabilidad", que la definen como la habilidad de un portal Grid de ser adaptado a varios recursos (p. ej., PDAs, PCs, teléfonos móviles, etc.).

Otra característica de calidad identificada es el "acceso transparente a los recursos", que definen como la habilidad de un portal Grid para proveer acceso a recursos Grid mientras aíslan al usuario de la complejidad. Confiabilidad, definida como la habilidad del portal para ejecutar los servicios especificados. Así mismo consideran dentro de esta característica de calidad a la tolerancia a fallos y a la disponibilidad. La facilidad de respuesta también es considerada y dentro de esta característica la escalabilidad y el acceso eficiente. Otras características también son definidas y tomadas en cuenta dentro de este modelo de calidad, tal es el caso de la empatía, la calidad de los datos y de la información y la seguridad. Teniendo en cuenta esas características, el modelo de calidad resultante es mostrado en la Figura 3-2.

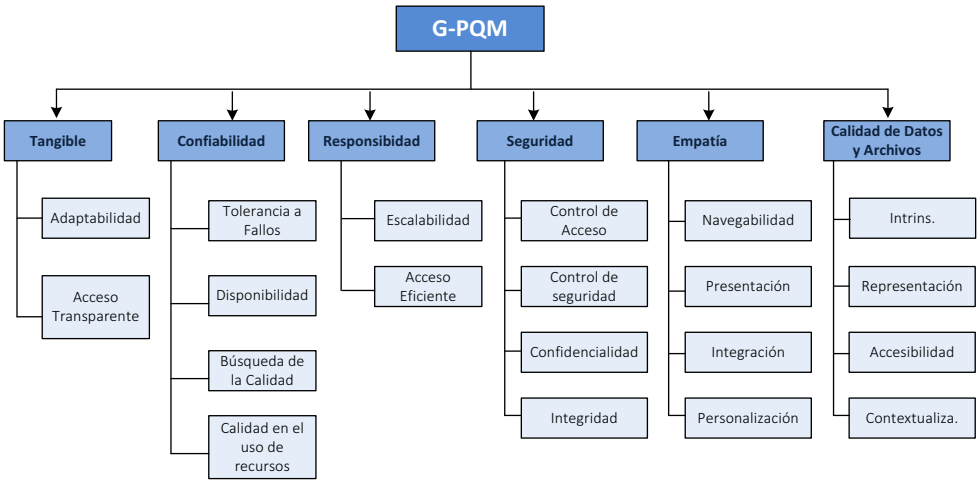


Figura 3-2. Características de Calidad. Moraga *et al.* (2002)

3.1.3. Calidad en la computación en la nube

La evolución de las tecnologías de computación en la nube, están promoviendo el desarrollo de nuevas técnicas para proveer servicios de alta calidad. Las infraestructuras de computación en la nube, proveen capacidad a clientes para usar software y servicios alojados en la plataforma de la nube.

Debido a la naturaleza del cloud, se puede observar un cambio fundamental en la forma en la que los servicios son construidos, desplegados, monitorizados, pagados y mantenidos. Como resultado, las técnicas de aseguramiento de calidad en ambientes cloud tienen que considerar ciertas características de calidad específicas en estos entornos como, por ejemplo, la "escalabilidad" y la "elasticidad".

En la computación en la nube, la cantidad de recursos es virtualmente ilimitada, por lo cual existen características de calidad propias del cloud, que deben ser consideradas. Por ejemplo, la provisión de recursos que traten con cargas de trabajo variantes y permitan de una manera elástica la entrega de servicios bajo demanda. Otros ejemplos particulares del cloud son aquellos en los que los proveedores realizan la facturación de sus servicios de acuerdo al uso de los recursos que el cliente consuma, la alta disponibilidad de los servicios, la posibilidad de tener múltiples instancias del servicio sirviendo a diferentes clientes, etc.

Como consecuencia de estas características, los arquitectos esperan que sus aplicaciones sean eficientes en el uso de recursos, para así ahorrar dinero mientras mantienen los requisitos de calidad del sistema. Esas características de

calidad necesitan ser evaluadas a través de modelos de calidad y métricas apropiadas. Por ejemplo, los clientes cloud y los proveedores necesitan negociar sus condiciones a través de los SLA.

Con respecto a los servicios en la nube, podemos abordar la calidad desde diferentes perspectivas (calidad de producto, calidad en uso y calidad de datos); de ahí que, en este trabajo nos enfocaremos en la calidad de producto, dado que el propósito de esta tesis es la monitorización de la calidad de los servicios. Sin embargo y dada la importancia de la calidad en este tipo de software, las otras perspectivas serán consideradas como prioridad dentro de nuestra investigación futura.

A continuación, se discute el estado actual de la calidad de los servicios cloud, haciendo hincapié en la calidad de servicios (QoS).

3.1.3.1 Mapeo Sistemático sobre Calidad de Servicios Cloud

Existen algunos estudios que permiten determinar el estado actual de la investigación de los enfoques de calidad de servicios cloud en donde se identifican el énfasis que los diferentes enfoques han puesto en esta área y las direcciones de investigación futura. Éste es el caso del mapeo sistemático presentado por Abdelmaboud *et al.* (2015), en el cual los autores han seleccionado 67 estudios primarios que fueron clasificados de acuerdo al enfoque, tipo de investigación y tipo de contribución.

Como resultado, estos autores han observado que la mayoría de artículos son dirigidos a la validación de la investigación (64%) mientras que el área de mayor investigación fue la de la calidad en las soluciones de Infraestructura como Servicio (IaaS) con un 48%, seguido por Software como Servicio (SaaS) con un 36%. Sus resultados confirman que los enfoques de QoS en computación en la nube se han convertido en un tópico importante en el área y existen varios retos que requieren de investigación futura.

En particular, herramientas, métricas y la evaluación de las propuestas existentes son necesarias para proveer servicios útiles y confiables a través del cloud que ofrezcan un adecuado nivel de calidad de servicios.

En el trabajo de Abdelmaboud *et al.* (2015), se habla del creciente interés tanto de la industria como de la comunidad académica con respecto a la calidad de servicio. De ahí la habilidad para especificar la calidad de servicio es un tema importante para clientes y proveedores de servicio. Sin embargo existe una falta de confianza entre clientes y proveedores además de dudas de cómo los clientes

pueden cerciorarse de que la calidad de servicio ofrecida por el proveedor es cumplida (Nallur *et al.*, 2013).

Los principales hallazgos del mapeo sistemático han sido presentados en dos facetas: la primera, presenta el área de investigación con el tipo de contribución y la segunda se centra en el área de investigación con el tipo de investigación. Además, los autores detallan hallazgos como el problema abordado, el enfoque básico, las limitaciones del enfoque, la validación y los resultados de la validación de cada estudio propuesto. Finalmente, han detallado los problemas y retos que aún están abiertos y requieren mayor atención. Los resultados obtenidos proveen una visión del estado actual de la investigación y muestran los posibles campos de acción para investigación futura.

Algunos de los estudios primarios de calidad de servicios cloud se centraron en el tópico de SaaS o aplicaciones cloud. Las aplicaciones cloud se ejecutan en infraestructuras cloud o centros de datos y proveen una respuesta rápida sin necesidad de control en la instalación, actualización o carga de nuevas versiones en un computador o servidor. Con las aplicaciones cloud, como ya se ha discutido en secciones anteriores, los datos pueden ser accedidos desde cualquier lugar a través de un navegador web una vez que se tenga conexión a Internet. En el modelo de negocio SaaS, la aplicación y los datos pueden ser ubicados con garantías de calidad de servicio. El proveedor cloud maneja, desarrolla y despliega las aplicaciones (Ardagna, *et al.,* 2013).

La mayoría de los estudios primarios que intervinieron en el estudio de Abdelmaboud *et al.* (2015) se centraron en los requisitos de calidad de servicio y en el manejo de la monitorización, y la escalabilidad de la aplicación, estas dos características son abordadas de la siguiente manera:

- En cuanto a los requisitos QoS, existen varios artículos que discuten estos temas, el problema básico está en que cada cliente de computación en la nube, tiene sus propios requisitos de rendimiento para completar las tareas para las cuales ellos requieren las ventajas de este tipo de computación. Ellos necesitan poder especificar las facilidades de computación requeridas para sus tareas específicas, permitiendo cualquier dependencia entre tareas. Los modelos de flujos de trabajo ayudan a los clientes a especificar sus tareas y los requisitos de recursos de computación para esas tareas (permitiendo las dependencias entre tareas) esos mecanismos permiten a los usuarios especificar sus requisitos en términos de calidad de servicio que quieren conseguir para sus necesidades específicas. Desde el punto de vista del proveedor, cada proveedor de servicio tiene muchos clientes y necesita asegurar que los

requisitos de calidad de sus servicios sean conseguidos. Esto tiene que ser realizado de manera efectiva en costos, entonces los proveedores de servicios necesitan poder identificar los períodos de demanda baja, en los cuales, por ejemplo los recursos de hardware pueden ser apagados para reducir costos eléctricos y períodos de alta demanda cuando recursos adicionales de hardware son demandados. Los clientes necesitan poder comparar diferentes proveedores de servicio con respecto a costos de sus requisitos de recursos esperados.

- Por otro lado, el rendimiento de la aplicación y el manejo de la monitorización también ha sido abordado dentro de los artículos estudiados en Abdelmaboud *et al.* (2015). La monitorización de los servicios en la nube ayudan a manejar y administrar los recursos cloud y las aplicaciones y esto ayuda a maximizar la eficiencia de los servicios. Sin embargo, la monitorización de las aplicaciones necesita mayor investigación, tales como herramientas y modelos para monitorizar apelaciones cloud, especialmente el rendimiento para asegurar la satisfacción de los clientes y las necesidades de las organizaciones.

Por otra parte, los autores investigan la forma de soportar el manejo de recursos y su QoS, el cual soporta el concepto de pago por consumo. La negociación automática con clientes individuales es necesaria para asegurar que un proveedor de servicios puede soportar los requisitos de muchos clientes distintos dados los recursos de los proveedores y los términos de uso. El resultado de tal negociación se ubica dentro de un Acuerdo de Nivel de Servicio (SLA). Los artículos que abordaron estos temas están acotados dentro del nivel de provisión de servicios (IaaS).

Entre los *enfoques de requisitos de calidad de servicio* abordados por Abdelmaboud *et al.* (2015), tenemos los siguientes trabajos descritos por su relación con el tema que se está abordando en este trabajo. Sin embargo, una lista completa puede ser encontrada en el mapeo sistemático en cuestión:

El trabajo de Liu *et al.* (2011) aborda el problema de cómo diseñar y desarrollar flujos de trabajo de sistemas cloud que soporten calidad de servicio cuyo enfoque básico es un proceso. El marco de trabajo para los sistemas de flujo de trabajos cloud tiene cuatro elementos: la especificación de requisitos, la selección de servicios, la consistencia de la monitorización y el manejo de violaciones. Este trabajo se ha validado a través de un caso de estudio. Y como resultado de la validación se tiene para el manejo de violaciones temporales una implementación concreta basada en el marco de trabajo propuesto.

El trabajo de Tolosana-Calasanz *et al.* (2012) aborda el problema de la definición de requisitos para las restricciones de calidad de servicio para aplicaciones científicas. Estos autores presentan un método y un proceso, ellos abordan la arquitectura de un flujo de trabajo para múltiples sistemas usando recursos físicos compartidos. Dentro de sus limitaciones está la necesidad de mejorar la exploración de la calidad de servicio. Estos autores han validado su propuesta a través de un caso de prueba con un montaje realista de un flujo de trabajo que muestra los beneficios del control de la calidad de servicio.

El trabajo de Wind *et al.* (2011) aborda cuestiones de la ingeniería de requisitos en aplicaciones basadas en componentes tales como modelos de facturación y problemas legales. Estos autores proponen un modelo, comparan el proceso de la ingeniería de requisitos a ser aplicadas y adaptan a los requisitos exactos de las soluciones basadas en cloud. El problema de su enfoque es la falta de validación para proveer resultados significativos de la Ingeniería de Requisitos.

El trabajo de Villegas y Sadjadi (2011) aborda la necesidad de los ingenieros de software de especificar requisitos de aplicación cuantitativamente para que así los proveedores de servicio puedan entender los requisitos y proveer la infraestructura necesaria. Ellos proveen un modelo. El manejador de IaaS permite a un clúster de máquinas virtuales ejecutar aplicaciones. El problema existente es que se deben mejorar los modelos de predicción para el rendimiento de las aplicaciones. Este enfoque está validado a través de un ejemplo, en el cual requisitos concretos pueden ser desarrollados y desplegados en un manejador cloud permitiendo requisitos no funcionales de alto nivel.

El trabajo de Schroeter *et al.,* (2012) identifica los requisitos para una arquitectura en tiempo de ejecución SaaS que permita maximizar los requisitos de calidad de servicios para multitenencia (quienes provean servicios a sus usuarios). Los autores proveen un modelo y engloban los requisitos para una arquitectura en tiempo de ejecución que direccione diferentes requisitos que pueden ser obtenidos extendiendo una arquitectura para aplicaciones dinámicas adaptativas. Su principal problema es que los requisitos no funcionales no son discutidos en detalle; necesitan servicios multitenencia específicos para la plataforma para autorizar a los usuarios cloud y asignar a cada huésped la configuración correcta. Otro problema de esta propuesta, es que no presenta ningún tipo de validación.

Liu *et al.* (2012) en su trabajo aborda los requisitos de QoS de los usuarios cloud durante el despliegue de los servicios de aplicación, ellos proveen un método para el manejo del despliegue de servicios para conseguir una reducción

de costos y mejorar la eficiencia. El problema de su propuesta es que no direcciona el despliegue de servicios de aplicación multi-objetivo basados en QoS. Ellos cuentan con la respectiva validación de su enfoque.

El trabajo de Niehörster *et al.* (2012) aborda la necesidad de los proveedores de encontrar configuraciones de datos virtuales eficientes y tratar con ambientes cloud dinámicos para cumplir con los objetivos de nivel de servicios; en este trabajo, los autores proponen un método, en la cual plantean una solución de software que reduce la necesidad de intervención humana para manejar centros de datos virtualizados. Como una limitación está que el enfoque propuesto no permite a los usuarios controlar y monitorizar los recursos, entonces es difícil para los usuarios asegurar que los requisitos de calidad de servicio sean cumplidos. Ellos presentan una validación cuyos hallazgos muestran que la aplicación puede adaptarse bien al cambio dinámico de ambientes cloud y maximizar la ganancia de los proveedores.

El trabajo de Ardagna *et al.* (2013), presenta el desarrollo de políticas efectivas de aprovisionamiento de servicios debido a la naturaleza dinámica de los servicios cloud. Los autores desarrollan un método de ubicación de recursos a través de la competición de proveedores SaaS. Como desventaja de este trabajo se tiene que el enfoque propuesto no direcciona la carga de trabajo compartida a través de varios ambientes cloud. Este enfoque ha sido validad, y demuestra su efectividad a través de simulaciones y un prototipo desplegad en Amazon EC2.

El trabajo de Alhamad *et al.* (2011) presentan una propuesta de gestión del SLA, confianza y medición del rendimiento para ambientes cloud. El trabajo presenta un modelo y métricas para medir rendimiento (uso de recursos). El problema de esta propuesta es que la mayoría de modelos discutidos se centran en evaluar el rendimiento de los servicios en plataformas locales en lugar de ambientes cloud. Además, la propuesta no ha sido validada empíricamente.

La propuesta de Vasar *et al.* (2012) aborda el problema de la monitorización y prueba de aplicaciones web en la nube. Los autores proveen un marco de trabajo para probar y monitorizar la escalabilidad de aplicaciones web en ambientes cloud. Sin embargo, el enfoque propuesto solo prueba instancias en Amazon. Los autores realizan una validación en la cual sus hallazgos muestran que su política basada en auto escalar es mejor que el mecanismo de Amazon Auto Scale con respecto al manejo exitoso de peticiones.

El trabajo de Katsaros *et al.* (2012) aborda la monitorización de aplicaciones desplegadas y ejecutadas en infraestructuras cloud dinámicas y de gran tamaño. Ellos plantean un método de monitorización y medición de calidad de servicio

a nivel de aplicación y de infraestructura. Sin embargo su trabajo muestra una falta de enfoques de balanceo de carga para comunicarse entre aplicaciones y mejorar la escalabilidad de la aplicación y el rendimiento. Los autores han validado su propuesta y los experimentos muestran un mejor rendimiento por habilitar la adaptación y cualquier aplicación con garantías de calidad de servicio.

Finalmente, en Freitas *et al.*, (2013) los autores realizaron un resumen de trabajos relacionados a la calidad en arquitecturas orientadas a servicios y en particular ciertas soluciones vinculadas a sistemas SaaS, teniendo en cuenta autores que proponen métodos de evaluación de la calidad, modelos de calidad, procesos de evaluación, análisis de requisitos y revisiones sistemáticas. Éstos estudios han sido resumidos en la Tabla 3-1, en donde el trabajo de Dubey *et al.* (2007) muestra los principales problemas y preocupaciones de las organizaciones encontrados en sistemas SaaS. Los autores mencionan que muchos de esos problemas pueden directamente afectar la calidad del producto e impactar sobre otros factores, tales como aspectos funcionales y comerciales, en el estudio de Dubey no se presentan métricas y criterios que permitan realizar evaluaciones de la calidad de dichos problemas, sin embargo su trabajo contribuye en esta tesis para mostrar ciertos aspectos de calidad que deben ser monitorizados, ya que impactan en la calidad de los servicios en la nube. Por otra parte, en el trabajo de O'Brien *et al.* (2007), los autores realizan un análisis entre una arquitectura orientada a servicios y atributos de calidad tales como disponibilidad, seguridad, confiabilidad, portabilidad e interfaces, en donde analizan preguntas y problemas relacionados con estas temáticas, éste estudio nos es de gran utilidad para definir y analizar cómo las características de calidad deben ser abordadas al momento de la monitorización. En el trabajo de Havelka *et al.* (1998) se propone un estudio empírico que obtiene un instrumento de tipo cuestionario para evaluar aspectos de calidad que impactan tanto positivamente como negativamente la adopción de servicios. Este estudio, es importante ya que nos permite analizar qué aspectos deben tenerse en consideración al momento de evaluar software orientado a servicios. Por otra parte el trabajo de Jureta *et al.* (2009) aborda aspectos vinculados a un método que permite la cuantificación de la calidad en sistemas SaaS. En este trabajo, los autores realizan un análisis sobre los estándares ISO 9126 e ISO 20000, proponiendo un método de calidad denominado QVDP (Quality-Value-Dependency-Priority). Su investigación toma en consideración diferentes perspectivas de los productos de software a ser evaluados. Por ejemplo características de portabilidad, soporte, personalización, etc.

Por otro lado, el trabajo de Cancian (2009) aborda un proceso para la evaluación de la calidad de SaaS. Además los autores realizan una síntesis y analizan los requisitos en relación a estándares y modelos de referencia existentes en la

literatura. Luego Cancian *et al.* (2009) propone una estructura genérica de para un SLA y define ciertos requisitos que deben ser observados en un modelo SaaS de tal manera que se pueda garantizar aspectos de calidad en los servicios. Si bien su solución nos ayuda a encontrar pautas para la búsqueda de un método de monitorización de la calidad, ésta presenta ciertos aspectos que hemos tratado de solventar a lo largo de este trabajo, como son temas de flexibilidad y facilidad en la definición de los requisitos no funcionales a ser evaluados. En el trabajo de Rosenberg *et al.* (2009) se abordan problemas en la composición de servicios y justifica temas de calidad en dicho proceso, los cuales representan una guía importante para nuestro trabajo dado que nos ayudan a entender ciertos aspectos de calidad que deben ser vigilados dado que son importantes a la hora de integrar servicios SaaS. Por último, el trabajo de Miguel *et al.* (2004) concluye que no existe un consenso en la literatura sobre qué modelo de calidad es el más apropiado para medir la calidad de un servicio IT. Dándonos una pista importante de que se necesita un método de monitorización que nos permita evaluar la calidad de una manera dinámica y flexible y que permita incluir métricas y características de calidad según se considere necesario.

Tabla 3-1. Resumen de soluciones que abordan temas de calidad de SaaS (Freitas *et al.*, 2013)

Trabajo Relacionado	Métodos de Evaluación	Modelos de Calidad	Procesos de Evaluación	Análisis de Requisitos	Revisiones Sistemáticas
Dubey *et al.* (2007)					x
O'Brien *et al.* (2007)		x		x	x
Havelka *et al.* (1998)				x	x
Jureta *et al.* (2009)	x	x			x
Cancian (2009)		x		x	
Cancian *et al.* (2009)				x	x
Rosenberg *et al.*(2009)	x		x		x
Miguel *et al.* (2004)				x	x

3.1.3.2 Cloud Service Measure Index (SMI)

Existe una gran variedad de servicios cloud de los cuales los clientes pueden hacer uso, siendo muy importante elegir el servicio "adecuado" (Garg *et al.* 2012). Los proveedores pueden satisfacer esos requerimientos. A menudo se deberán poner en una balanza los requisitos funcionales y no funcionales para evaluar a los distintos servicios como proveedores. De ahí no solamente es importante descubrir los servicios que existen sino además evaluarlos.

En este contexto, el *Cloud Service Measure Index Consortium* (2015), ha identificado métricas que son combinadas en forma de un Índice de Medición de Servicios (*Service Measure Index*, SMI), ofreciendo evaluación comparativas de

servicios cloud. Esos índices de medición pueden ser usados por clientes para comparar los servicios ofrecidos. Por tanto en su trabajo estos autores proponen un marco de trabajo (SMICloud) que puede comparar diferentes proveedores cloud basándose en sus requisitos. Este marco de trabajo permite comparar los servicios desde sus prioridades y algunas dimensiones.

Los atributos SMI son diseñados en base los estándares ISO 9126/25000 e ISO/IEC 20000 y teniendo en consideración conceptos introducidos por ISACA (Csmic, 2014), por el consorcio CSMIC Consortium (2015). Estos consisten en un conjunto de indicadores claves de desempeño (KPIs) que proveen un método estandarizado de medición y comparación de servicios de negocios. Provee además una vista de calidad de servicio necesaria para que los clientes puedan seleccionar un servicio cloud basándose en su manera de facturación, agilidad, aseguramiento del servicio, costo, rendimiento, seguridad, privacidad y usabilidad. Además proveen un modelo de calidad y un conjunto de métricas para servicios cloud a nivel de IaaS. Sin embargo, las métricas propuestas son únicamente para medir la usabilidad y se necesita definir nuevas métricas para medir los demás indicadores del SMI.

3.1.3.3 Modelos de Calidad para SaaS

Recientemente, varios modelos de calidad han sido propuestos para servicios cloud (Lee *et al.*, 2009) (Freitas *et al.*, 2013) (Wen *et al.*, 2013) (Zheng *et al.*, 2014) (Zhou *et al.*, 2015), pero ninguno cuenta con una recopilación completa de características y atributos de calidad, con sus respectivas métricas, y que a su vez esté alineado con estándares de calidad como el ISO/IEC 25010. A continuación, se discuten estos modelos por orden cronológica de aparición.

Lee *et al.* (2009) propone un modelo de calidad para evaluar SaaS. Los autores definen las características claves de SaaS, y luego, derivan los atributos de calidad desde esas características claves en base al estándar ISO/IEC 9126 para finalmente definir métricas para medir los atributos de calidad.

El modelo consta de solo cinco métricas, que se enfocan en el punto de vista del proveedor del servicio. En entornos cloud, existen otros stakeholders cuyo punto de vista son igualmente relevantes para la calidad del servicio: consumidor, facilitador (*broker*), cliente final y desarrollador.

Según esta propuesta, un SaaS puede ser evaluado por los proveedores de servicios y además los resultados pueden ser utilizados como un indicador para la mejora de la calidad del servicio. Las características claves identificadas por los autores se muestran en la Figura 3.3, mientras que la Figura 3-4 muestra el mapeo desde las características clave de SaaS a atributos de calidad.

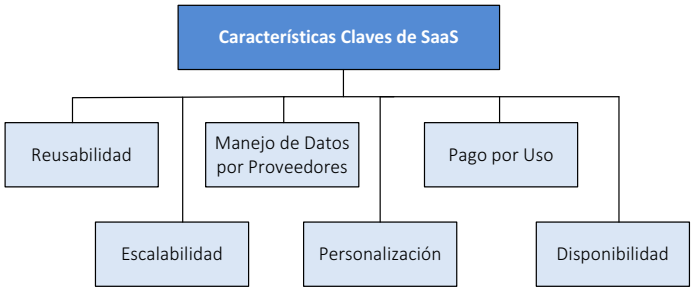


Figura 3-3. Características Claves de Calidad SaaS (Lee *et al.*, 2009)

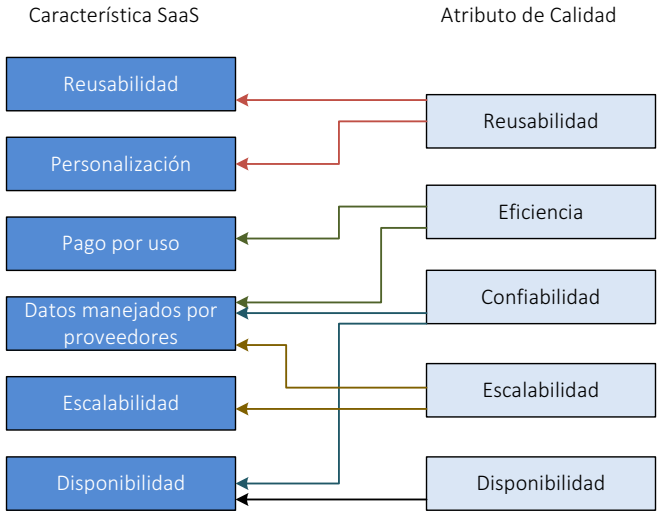


Figura 3-4. Mapeo desde las Características Clave de SaaS a Atributos de Calidad

Freitas *et al.* (2013), presentan un método para evaluar la calidad de un producto de software ofrecido como un servicio, llamado SaaSQuality. Este método contiene un modelo de calidad para el contexto SaaS, basado en el estándar ISO 9126 (ISO/IEC, 2001) y modelos para el manejo de IT (ITIL y COBIT). Los resultados experimentales que ellos obtuvieron a través de un caso de estudio muestran la evaluación de este tipo de software. Ellos presentan como los requisitos más relevantes en sistemas SaaS a la funcionalidad, usabilidad, seguridad, rendimiento, soporte, nivel de servicio y portabilidad. Los pasos de su método hacia evaluar la calidad de SaaS son: 1) la definición de objetivos y contextualización de la evaluación, 2) la definición y configuración del modelo de calidad que será usado para las evaluaciones, 3) la identificación de las métricas, 4) La evaluación de puntajes y criterio de juicio, 5) la evaluación del proyecto, 6) la ejecución de la evaluación.

Wen *et al.* (2013) presentan un modelo de calidad para evaluar los servicios SaaS desde las perspectivas de la plataforma, el proveedor y el cliente. Para construir su modelo los autores identifican características claves de los servicios SaaS pero utilizan las métricas de calidad de propuestas en el estándar ISO/IEC 25010 para medir estas características.

El modelo se compone de varios niveles: SaaS básico, SaaS estándar, SaaS optimizado y SaaS integrado, y para cada nivel el modelo cuenta con un conjunto de métricas para los puntos de vista del proveedor y del consumidor del servicio. Este modelo no presenta métricas nuevas, que son específicas de los servicios cloud, sino que solo hace una recopilación de las métricas generales propuestas en el estándar ISO/IEC 25010 y las clasifica en los niveles mencionados. Otra debilidad, es que solo evalúan servicios SaaS. Hay ciertas características de calidad (p. ej., elasticidad) que afectan también a PaaS e IaaS.

Zheng *et al.* (2014) presentan un modelo de calidad llamado CLOUDQUAL, que especifica seis dimensiones de calidad: usabilidad, disponibilidad, fiabilidad, responsividad, y elasticidad. Este modelo cuenta con solo cinco métricas, y está pensado para servicios cloud en general. Los autores también presentan una prueba de concepto para evaluar la eficacia de su modelo. Se han inspirado en los modelos de calidad SERVQUAL y el E-service para realizar su modelo de calidad. Una limitación que posee este modelo, es que es muy limitado, cuenta con muy pocos atributos y métricas, lo cual no garantiza una amplia evaluación de los servicios en la nube.

Zhou *et al.* (2015) presentan un modelo de calidad para servicios cloud. Este modelo se construye en base a seis características: usabilidad, seguridad, fiabilidad, tangibilidad, responsividad y empatía. Cada característica la han dividido en algunas sub-características y los autores también proponen un conjunto de métricas para evaluar esas sub-características, las cuales son agrupadas en dos categorías: objetivas y subjetivas. El modelo se basa en el estándar ISO/IEC 25010 para categorizar calidad del servicio en características, pero no utilizan todas las características que el estándar propone, si no que usan las características que consideran esenciales. Sin embargo, los autores no indican en base a qué criterio han propuesto dichas características. Otra limitación es que en este modelo se proponen solo cinco métricas, y que sirven para evaluar las sub-características cortesía (una métrica), disponibilidad (una métrica) y continuidad (tres métricas). Por lo que el trabajo está incompleto, ya que no se pueden evaluar todas las sub-características que proponen.

3.1.3.4 Métricas para Servicios Cloud

Li *et al.* (2012) presentan una revisión sistemática de la literatura que identifica estudios existentes sobre evaluación de servicios cloud y recoge métricas con el fin de construir un catálogo de métricas. Durante el estudio los autores encontraron que las evaluaciones de calidad existente se centran mayoritariamente en la característica de "rendimiento" de los servicios cloud. Los autores se centran en la evaluación de servicios cloud comerciales. Este estudio abarca a la Infraestructura como Servicio (IaaS) y Plataforma como servicio (PaaS) sin tener en cuenta el Software como Servicio (SaaS).

Un ejemplo de este tipo de estudios es el presentado por (Bolch *et al.,* 2006) donde se proponen varias métricas orientadas al rendimiento tales como tiempo de respuesta. Sin embargo, estas no son insuficientes para situaciones relevantes en entornos cloud; por ejemplo, este tipo de métricas no consideran cargas de trabajo cambiantes (Becker *et al.*, 2015).

En otro trabajo, Becker *et al.* (2015) propone métricas para medir la escalabilidad, elasticidad y eficiencia de sistemas cloud usando el método *Goal Question Metric (GQM)*. Los autores utilizan un ejemplo que permite utilizar un conjunto inicial de seis métricas. Sin embargo, no abordan el desarrollo de métodos de análisis y herramientas que permitan a los arquitectos de software verificar el cumplimiento de la escalabilidad, elasticidad y eficiencia de sus aplicaciones cloud en tiempo de diseño, por lo que dejan este aspecto como trabajo futuro.

Luego, en un trabajo posterior, Lehrig *et al.* (2015) del mismo grupo de investigación, examinan estas métricas desde el punto de vista de los clientes, proveedores cloud y arquitectos de software con respecto a los conceptos comúnmente usados. Estos autores también llevaron a cabo una revisión sistemática con el objetivo de obtener conceptos, definiciones, así como también más métricas de escalabilidad, elasticidad y eficiencia, y posteriormente evaluaron la manera en la que la literatura diferencia estas tres propiedades, ejemplifica las métricas y considera características y roles típicos del cloud. En este trabajo, los autores obtuvieron 418 estudios primarios de los que 20 fueron analizarlos en profundidad y recomendaron conceptos definiciones y métricas para cada propiedad. De este trabajo, los arquitectos de software pueden usar sus recomendaciones para evaluar la calidad de las aplicaciones cloud y luego los proveedores y clientes pueden especificar objetivos de nivel de servicio en base a estas métricas.

3.1.3.4 Otros enfoques orientados a Multi-Cloud y Multitenencia

He *et al.* (2012) introducen un nuevo enfoque dirigido por calidad para ayudar a los desarrolladores SaaS a seleccionar los servicios para componer aplicaciones SaaS con multitenencia, el cual consigue objetivos de optimización mientras cumple con diferentes restricciones de calidad de servicio. Sus resultados experimentales muestran que su enfoque permite mejorar la efectividad y rendimiento.

Por otra parte, Muntés-Mulero *et al.* (2013) se centran en proponer un enfoque para asegurar la calidad de servicios en entornos multi-cloud. Los autores definen una aplicación multi-cloud como una pieza de software que usa algunos servicios cloud alojados por dos o más proveedores diferentes. Usualmente, dos escenarios diferentes son considerados al referirnos a los ambientes multi-cloud. El uso de múltiples servicios cloud desde múltiples proveedores, agrega una nueva dimensión de complejidad a un escenario ya complejo como es el de computación en la nube.

Ser capaz de evaluar los riesgos y los atributos de calidad que están específicamente relacionados con entornos multi-cloud es esencial para el diseño de aplicaciones fiables basados en el uso de servicios cloud (Muntés-Mulero et al., 2013). Existe la necesidad no sólo para identificar los indicadores o métricas relevantes, sino también comprender cómo se relacionan las métricas con los riesgos potenciales. La heterogeneidad causada por proveedores independientes que han creado sus propios modelos de negocio, protocolos, procesos y formatos, genera un incremento en el número de riesgos que hay que tomar en consideración cuando se crea una nueva aplicación usando la estrategia multi-cloud. Muntés-Mulero *et al.* (2013) enfatizan tres aspectos esenciales que hay que tomar en consideración en los ambientes multi-cloud, y que consideramos que son desafíos a la hora de realizar la evaluación de la calidad del servicio:

- *La heterogeneidad de los servicios ofrecidos por diferentes proveedores y problemas de interoperabilidad*: la falta de interfaces estándar para servicios de las diferentes nubes y la creación de sistemas propietarios independientes por cada proveedor, hace a los ambientes multi-cloud muy heterogéneos. Los problemas de interoperabilidad pueden ir desde cuestiones técnicas, tales como interfaces de mensajería o la calidad del servicio, a cuestiones semánticas, organizativas o jurídicas. Esta heterogeneidad es un riesgo importante a tener en cuenta a la hora del diseño, ya que influirá en la decisión de un arquitecto en seleccionar entre un servicio y otro. En términos de calidad, un servicio será altamente interoperable con otros sistemas si se

puede combinar en colaboración con muchos otros servicios, de la misma o de otros proveedores de servicios en la nube.

- *La migración entre los servicios ofrecidos por diferentes proveedores cloud es una operación esencial para asegurar el cumplimiento de los requisitos de la aplicación*: una de las razones más comunes para desplegar una aplicación en un entorno multi-cloud, es el incremento del catálogo de servicios cloud y el aumento de la capacidad de los usuarios de migrar de un servicio a otro en caso de que no se cumplan los requisitos de una aplicación. A esto le denominamos la "intercambiabilidad de capacidad", y que representa la facilidad para migrar de un servicio a otro para reemplazar al primero. Será esencial para descomponer los procesos de migración de un servicio cloud a otro en varios pasos más detallados, y analizar los aspectos de calidad que deben considerarse en el proceso.
- *Las amenazas de seguridad se incrementan en entornos de computación multi-cloud*: aumentar el número de servicios y proveedores, aumentará la complejidad del sistema global y el número de ataques potenciales. El control sobre los datos de los clientes disminuye, sobre todo debido a la potencial migración entre servicios de distintos proveedores. La comunicación continua de datos entre servicios en diferentes nubes también puede traer como resultado que el almacenamiento de datos en los sistemas externos intermediarios sea menos seguros, el aumento general de la vulnerabilidad y potencialmente comprometer la información confidencial. En cuanto a la privacidad de los datos, el multi-alquiler hace que sea más difícil garantizar la confidencialidad de la información sensible.

3.1.4. Discusión

A partir del análisis realizado podemos concluir que la computación en la nube tiene características específicas (elasticidad, escalabilidad, multitenencia, etc.) que plantean nuevos retos que tienen que ser abordados para asegurar la calidad de los servicios en un entorno distribuido, heterogéneo y dinámico.

Tanto en la computación distribuida como en la computación en la nube y en general en las arquitecturas orientadas a servicios, se hace necesario contar con mecanismos que permitan definir y medir la calidad en estos entornos. Los modelos de calidad juegan un rol central en la definición y evaluación de la calidad de los servicios cloud ya que sirven de base para establecer los requisitos de calidad y proporcionar mecanismos para medidos.

Aunque en los últimos años varios modelos de calidad específicos para el cloud han sido propuestos, ninguno de ellos cuenta una definición holística de los servicios cloud, que integran todas las características, atributos y métricas

relevantes para este dominio y permita evaluar distintas categorías de servicios (SaaS, Paas e IaaS) y artefactos cloud (especificaciones cloud, arquitecturas cloud, servicio cloud en uso) en distintas fases del ciclo de vida del servicio, y que además esté alineado con estándares de calidad de producto como la ISO/IEC 25010.

Por otra parte, se han propuesto numerosas métricas para evaluar la calidad de los servicios cloud pero no existe un estudio que recoja esta información y la clasifique con respecto a las características internas y externas del servicio (QoS) y características en uso de los servicios cloud (*Quality of Experience*, QoE).

Finalmente, los trabajos existentes no proporcionan evidencias sobre el uso de los modelos de calidad así como de otras técnicas de aseguramiento de calidad antes mencionadas en entornos cloud. En estos entornos, no podemos detener el servicio para evaluar su calidad, lo que implica que los modelos de calidad se tienen que utilizar en tiempo de ejecución, junto con otros mecanismos que permitan monitorizar la calidad del servicio en tiempo real, extraer información de los servicios para calcular las métricas y utilizar la información para detectar incumplimientos del SLA, mejorar la calidad del servicio o para facturación.

3.2. Métodos de apoyo al cumplimiento de acuerdos de nivel de servicio

Como se ha mencionado antes, un acuerdo de nivel de servicio (SLA) es un documento formal, negociado que define en términos cuantitativos (e incluso cualitativos) los requisitos del servicio que será aprovisionado a un cliente. Cualquiera de las métricas incluidas en un SLA debería ser capaz de ser medida sobre una base regular. Este es un contrato legalmente acordado entre dos o más partes. Estos acuerdos juegan un rol central en el ciclo de vida de los servicios, ya que capturan las expectativas de los servicios y responsabilidades de las entidades dirigidas por las decisiones de la ingeniería al nivel de concepción (durante, por ejemplo, el diseño del servicio) y decisiones operacionales (durante, por ejemplo, la entrega y el uso del servicio). Los SLA habilitan a las entidades participantes a acordar que servicios serán ofrecidos, como los servicios serán entregados y quien será el responsable de la ejecución, ajustes, fallos potenciales y aspectos de privacidad.

Sin embargo, los SLA son acuerdos limitados a la descripción de expectativas y responsabilidades. Como se describe en Kyriazis (2013): *"Un SLA no puede garantizar que obtendrá el servicio que describe, no más que una garantía puede garantizar que el coche nunca se romperá"*. En particular, un SLA no puede hacer un buen

servicio de un mal servicio. Al mismo tiempo, un SLA puede mitigar el riesgo de la elección de un mal servicio".

Estudios en este campo (Alhamad *et al.*, 2011; Comuzzi *et al.*, 2013; Correia *et al.,* 2011; McCarthy, 2011) muestran la necesidad de herramientas de soporte y mecanismos que puedan ser usados durante las distintas fases del ciclo de vida del SLA, tales como la monitorización de la ejecución de los servicios, la adhesión a los términos del acuerdo y el cumplimiento obligatorio a través del disparo de acciones para ayudar a cumplir los requisitos deseados. El principal objetivo de tales marcos de trabajo es asegurar que el servicio es entregado de acuerdo a niveles de calidad esperados.

3.2.1. Métodos de apoyo al cumplimiento de acuerdos de nivel de servicio para arquitecturas orientadas a servicios

Muchas compañías e instituciones académicas se han centrado en desarrollar plataformas de servicios web, herramientas y aplicaciones. IBM por ejemplo presentó su arquitectura *Web Service Conceptual Architecture* (WSCA) (Kreger, 2001). Microsoft anunció su marco de trabajo .NET en el 2000, pudiendo a través de ésta construir desplegar y ejecutar servicios web XML y aplicaciones (Microsoft).

Hewlett-Packard fue pionera en el desarrollo de la plataforma de servicios web con su solución E-speak en 1999 ("E-Speak Web"). Por otro lado, existen también Sun Microsystems con Sun Open Net Environment (Sun One), Oracle con Oracle9i/Web Service Framework y BEA Systems con J2EE Web Service Platform.

Existen retos dentro de los servicios web tales como la interoperabilidad. Los servicios Web son desarrollados y desplegados por varias compañías. Entonces de ahí vienen las preguntas de cómo son ellos descubiertos y luego cómo se comunican unos con otros. Tres estándares han sido propuestos para mitigar este problema:

1. El lenguaje de definición de servicios web (WSDL) para definir las funciones expuestas por un servicio web;
2. La descripción universal, descubrimiento e integración (UDDI) para publicar y descubrir servicios, y finalmente,
3. El protocolo de acceso simple a los objetos (SOAP) para la comunicación entre servicios web en XML.

El siguiente conjunto de preguntas relevantes a ser abordadas son: después de descubrir múltiples servicios web similares, ¿cuál es el correcto en términos

de disponibilidad, costo, tiempo de respuesta, duración total, etc., y ¿cómo podemos asegurar que el "servicio correcto" es siempre "correcto"? Esas preguntas tienen que ser respondidas bajo las necesidades de las organizaciones y sus procesos de negocio. Los Acuerdos de Nivel de Servicio proveen una respuesta a esta pregunta (Jin *et al.*, 2002).

En muchos casos, los servicios web son puntos de acceso a procesos de negocio que son llevados por proveedores de servicios. Mientras, un proceso de negocio puede ser compuesto por uno o más servicios web que son provistos por otras unidades de negocio u organizaciones.

Correia *et al. (*2010) propusieron un enfoque basado en modelos para SLA de servicios que permitía su especificación y verificación, su especificación proponía un lenguaje de SLA para definir atributos de calidad como RNF en el contexto del manejo de tecnologías. Este lenguaje se denominó SLALOM (Correia *et al.*, 2011), para la definición de este lenguaje ellos propusieron un meta-modelo que permitiera una fácil especificación de SLA así como también su monitorización. Sin embargo ellos no se basan en estándares de especificación de SLA, lo que hace que se aparten de las soluciones basadas en estándares e impide una estandarización en cuanto a sus meta-clases y a su solución en general.

Por otra parte, Bianco *et al.* (2008) en su trabajo hablan sobre la importancia que los atributos de calidad juegan en la selección de los servicios en arquitecturas orientadas a servicio SOA. Además mencionan las generaciones por las cuales las arquitecturas orientadas a servicio han pasado tales como la primera generación basada en componentes monolíticos y luego la segunda generación que está basada en componentes integrados verticalmente que pueden ser adaptados y reconfigurados en tiempo de instalación.

En el futuro, además ellos vislumbran una tercera generación de servicios que serán integrados de una manera sensible al contexto y reconfigurable de una manera bajo demanda. Para ello se tendrán en cuenta aspectos de calidad los cuales son descritos en los SLA y su automatización. De ahí ellos proponen una nota técnica que provee una visión del estado de la práctica en el desarrollo y manejo de los SLA en el contexto de SOA.

En su reporte estos autores consideran preguntas importantes tales como los mecanismos que pueden ser usados para asegurar la calidad del servicio por medio de un contrato; los atributos de calidad que pueden ser descritos en un SLA, los mecanismos que son usados por los proveedores de servicios para conseguir y monitorizar esos atributos de calidad, la tecnología que soporta estos aspectos, tales como herramientas, especificaciones y estándares, los cuales están

disponibles para expresar los requisitos de calidad y así ver que se considerará en el futuro.

Este trabajo constituye un muy buen punto de partida ya que detalla ciertos lenguajes de descripción y los contrasta, siendo de gran utilidad en la adaptación de los SLA y la manera de expresar ciertas características de calidad en los mismos. Los lenguaje de descripción abordados y analizados por estos autores son claves en la manera en la que los SLA son descritos y han sido un punto de partida para la especificación de los SLA en tecnologías tales como la computación en la nube, lo que hace muy interesante como parte de este trabajo el análisis de dicho reporte.

3.2.2. Métodos de apoyo al cumplimiento de acuerdos de nivel de servicios para servicios en la nube

En los últimos años se han presentado muchos trabajos que abordan métodos que ayudan al cumplimiento de los SLA comenzando desde su especificación hasta la monitorización de los servicios desplegados y la forma en que se verifica el cumplimiento de estos contratos.

A nivel Europeo se han realizado varios esfuerzos para conseguir métodos que apoyen el cumplimiento los SLA, haciendo de éste un tema de gran relevancia dentro del ámbito de los servicios cloud.

A continuación, se presenta un breve resumen de algunos de los proyectos más relevantes en este tema y que entre otros también han sido abordados en el reporte de resultados de investigación *"Cloud Computing Service Level Agreements Exploitation of Research Results"* (2013)

Morfeo 4CaaSt (2007-2013): El proyecto 4CaaSt crea una plataforma avanzada PaaS, la cual soporta un hosting optimizado y elástico para aplicaciones de Internet multi-nivel. 4CaaSt incluye todas las características necesarias tales como fácil programación de aplicaciones y habilita la creación de un ecosistema de negocios donde las aplicaciones vienen de diferentes proveedores que pueden ser mezcladas y tratadas por diferentes usuarios. Este proyecto desarrolla un lenguaje propio de descripción llamado "blueprint" que facilita la definición de aprovisionamiento y reglas de negocio con respecto a la elasticidad y multitenencia, como también la inclusión de marcas para el desarrollador de la aplicación para mapear términos de aplicaciones de alto nivel en parámetros de bajo nivel. El manejo de los SLA se centra en la gestión del mismo, trasladando los requisitos del usuario hacia una arquitectura de despliegue y dimensionamiento de la arquitectura y reglas de elasticidad.

Cloud4SOA (2010-2013): Provee un marco de trabajo interoperable para desarrolladores PaaS. El sistema soporta desarrolladores de aplicación con multi-platafoma. Habilita la negociación dinámica del SLA y herramientas que permiten analizar los ofrecimientos y rendimiento y adaptarse a los SLA convenientemente. El framework permite a los proveedores y clientes negociar flexiblemente entre el estándar y un SLA personalizado mientras soporta dinámicas de negocios a través de métricas relacionadas al SLA que serán monitorizadas y analizadas. Este proyecto utiliza como lenguaje de descripción de SLA el WS-Agreement.

CloudScale (2012-2015): Este proyecto trata de analizar, predecir y resolver problemas de escalabilidad en ambientes cloud. Identifica las causas de violaciones del SLA, cuando los servicios no escalan de la forma prevista.

Contrail (2010-2014): Este proyecto ofrece servicios elásticos PaaS sobre una federación de nubes IaaS, mientras se encarga de problemas relacionados a la calidad de servicio, manejo SLA, seguridad, interoperabilidad y escalabilidad. Además, el proyecto ha desarrollado un modelo de calidad para capturar diferentes parámetros de interés para clientes y proveedores

EGI Federated Cloud (2011-2014): Este proyecto es visto como un conjunto de redes académicas privadas y virtualizadas, construidas alrededor de estándares abiertos y centradas en los requisitos de la comunidad científica. Dentro de este proyecto, se desarrollan plantillas SLA para una instanciación fácil y automatizada para pequeñas comunidades de investigación que no puedan permitirse negociación de SLA personalizados. Cuando se usa los recursos provistos por este proyecto, los investigadores y comunidades de investigación pueden tener un control total sobre las aplicaciones desplegadas, recursos elásticos basados en necesidades reales, cargas de trabajo procesadas inmediatamente, una infraestructura extendida a través de proveedores de recurso en Europa y un rendimiento escalable con consumo de recursos elástico.

ETICS: Automatiza y mejora la ejecución de construcciones y pruebas. El sistema ETICS está diseñado para simplificar el proceso de desarrollo y para soportar la ejecución de escenarios de prueba complejos mientras mejora la calidad del software producido. Este proyecto tiene como objetivo entregar un nuevo control de red, manejo y planificación de tecnologías para mejorar la QoS a través de los proveedores de red, mejorando la cadena de valor. Contiene plantillas de SLA propias mejoradas para dominios de negocios.

GEYSER: Green nEtworked data centers as energY proSumers in smaRt city environments. Este Proyecto El proyecto provee mecanismos para coordinar y aprovisionar recursos de red y tecnológicos, para evitar las limitaciones de la

segmentación de red, análisis de negocio, servicios y dependencias a través de un marco de trabajo de negocios y composición de infraestructuras lógicas siguiendo los recursos de infraestructura. Ellos han desarrollado un marco de trabajo para generar SLA propios. Están orientados a IaaS.

IRMOS: Desarrolla soluciones Cloud que permiten la adopción de aplicaciones interactivas y de tiempo real, habilitando un conjunto de atributos y su integración eficiente en infraestructuras cloud. Traduce las especificaciones de calidad de alto nivel a especificaciones de bajo nivel. En este proyecto se han propuesto dos tipos de SLA (SLA al cliente y SLA técnico). Además, dentro de los entregables de este proyecto se encuentra un reporte técnico (Menychtas *et al.*, 2009) en donde se presenta detalladamente la manera en la que abordan el SLA, partes, aspectos de innovación, evaluación de la calidad a partir de los SLA, etc.

MODAClouds: Provee métodos, un sistema de soporte de decisiones, un IDE de código abierto y un ambiente en tiempo de ejecución para el diseño de alto nivel, prototipado, generación de código semi-automático de aplicaciones con garantía de calidad de servicio. Para ello realizan una monitorización de los SLA a nivel de PaaS y de IaaS con el fin de evaluar el cumplimiento de los mismos. Luego utilizando modelos en tiempo de ejecución realizan un ajuste de los servicios con la finalidad de proporcionar un cumplimiento de los mismos.

La herramienta del SLA, creada como un prototipo dentro de sus entregables, es la responsable de generar un documento que describa el SLA entre las partes envueltas en el MODACloud: clientes, proveedores de aplicaciones y proveedores cloud. Esta es una salida que juega un rol de interface entre el análisis del SLA y el tiempo de ejecución. La Figura 3.5 presenta la arquitectura del proyecto MODACloud.

Optimis (2010-2013) - Optimized Infrastructure Services: Habilita a las organizaciones la posibilidad de externalizar sus servicios y aplicaciones de una manera confiable y auditable por los proveedores cloud, mientras se optimiza el ciclo de vida completo de ingeniería de servicios, provisión, operación, entrega y uso. Su manifiesto de servicio contiene un núcleo común, las extensiones del proveedor del servicio, las extensiones del proveedor de la infraestructura. Para describir estos aspectos extienden el WS-Agreement. Se centran principalmente en características tales como la confiabilidad y la auditabilidad.

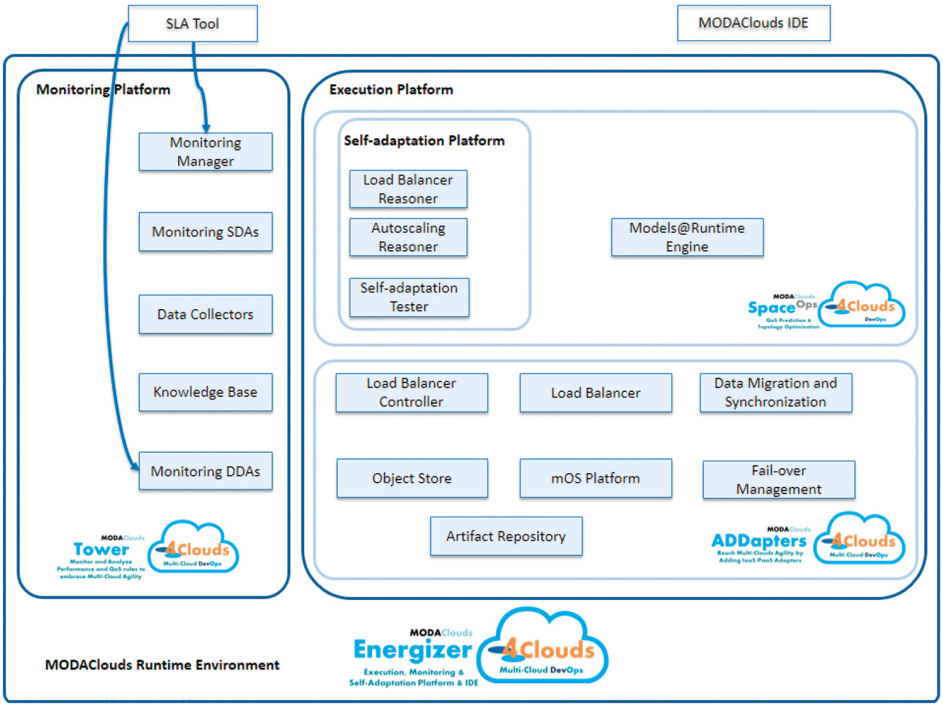


Figura 3-5. Dependencias de la Herramienta que Gestiona el SLA en MODAClouds

Q-ImPrESS (2008-2010): El proyecto ha desarrollado un método para software dirigido por la calidad, y evolución, en donde las consecuencias de las decisiones de diseño y los recursos del sistema cambian en rendimiento, confiabilidad y mantenibilidad a través de un análisis de impacto y simulación. Desarrollan un marco de trabajo propio para el manejo de los SLA que permite determinar el impacto y *tradeoff* de cada uno de los parámetros o atributos de calidad mientras considera diferentes estados. Entre las publicaciones realizadas a partir de este proyecto, Ardagna, *et al.* (2008) discuten el rol de los modelos en el desarrollo de software, centrándose en modelos que soportan razonamiento sobre las propiedades no funcionales (rendimiento y confiabilidad).

Adicionalmente, se centran en sistemas evolutivos adaptativos, los cuales pueden reconfigurar su estructura y comportamiento para responder los cambios continuos en los requisitos y en el ambiente. Estos autores ponen énfasis en que los modelos no solamente pueden ser usados en tiempo de diseño para guiar las estrategias del desarrollo de software sino adicionalmente en tiempo de ejecución. En tiempo de diseño el objetivo de los modelos de rendimiento y confiabilidad es proveer predicciones cuantitativas en tiempo de ejecución. Adicionalmente abordan la monitorización de las posibles violaciones de estas

características y la manera en la cual se pueden disparar mecanismos de reconfiguración. Tópicos que actualmente siguen investigándolos, sin embargo ellos se centran en aspectos muy específicos de calidad.

SERSCIS (2008-2016) -Semantically Enhanced Resilient and Secure Critical Infras-tructure Services: Desarrollan una infraestructura orientada a servicios auto-adaptativa, para crear, monitorizar y manejar seguridad y sistemas de información altamente disponibles. Están orientados a nivel de servicios IaaS y SaaS. Con respeto al SLA, desarrollan una arquitectura en capas para el manejo de los SLA, esta arquitectura, no está basada en arquitecturas anteriores.

SLA@SOI (2008-2011): El proyecto SLA@SOI direcciona el ciclo de vida del SLA a través de una negociación autónoma, aprovisionamiento, monitorización y adaptación de los SLA, mientras se encarga de toda la pila de servicios desde los aspectos de negocio hasta la infraestructura física. Con respecto al SLA, desarrolla el lenguaje de especificación SLA(T), para la descripción de las características funcionales y no funcionales, usando una sintaxis abstracta. Este proyecto de investigación tiene como objetivo proveer un hito mayor para la evolución hacia una economía orientada a los servicios, en donde los servicios basados en IT pueden estar flexiblemente tratados como mercancías, por ejemplo bajo condiciones muy bien definidas con costos asociados. Eventualmente, esto permitirá redes de valor dinámico que pueden ser instanciadas de manera flexible, proporcionando innovación y competitividad.

SLA@SOI propone la provisión de tres beneficios mayores al aprovisionamiento de los servicios.

- *Predictibilidad y Dependencia*: Las características de calidad de los servicios pueden ser predichas y forzadas a ser cumplidas en tiempo de ejecución.
- *Manejo transparente del SLA*: Los acuerdos de nivel de servicios (SLA) definen las condiciones exactas bajo las cuales los servicios son provistos/consumidos que pueden ser manejadas de manera transparente a través de toda la pila de negocios e IT.
- *Automatización*: Todo el proceso de negociación del SLA y aprovisionamiento, entrega y monitorización de servicios puede ser automatizado permitiendo el consumo altamente dinámico y escalable de servicios.

Además, otros trabajos de investigación abordan el cumplimiento de características de calidad establecidas en los SLA, entre ellos está el trabajo de Emeakaroha *et al. (*2010) en el cual ellos abordan temas como el alto costo de las violaciones del SLA y la reacción ante fallos; manifestando la necesidad de

estrategias que incluyan conceptos de monitorización avanzadas. Por otro lado, estos autores hablan sobre la existencia de herramientas poco escalables y que no soportan un mapeo entre métricas de bajo nivel y parámetros de alto nivel para parámetros específicos del SLA. Por tanto, ellos presentan un marco de trabajo que permita manejar estos mapeos y lo han llamado LoM2HiS, embebido en la infraestructura FoSII, la cual facilita un manejo autonómico del SLA. De ahí, el LoM2HiS detecta violaciones del SLA y notifica los problemas existentes.

En un trabajo posterior Emeakaroha *et al. (*2012) propone una arquitectura de monitorización llamada CASViD, la cual permite la detección de las violaciones de los SLA en la capa de aplicación e incluye herramientas para la asignación de recursos, programación de tiempos y despliegue. CASViD se centra en una monitorización a nivel de aplicación, la cual es relevante cuando múltiples usuarios comparten los mismos recursos en un ambiente Cloud.

Gonzalez *et al. (*2012) en su trabajo estudian el nivel de disponibilidad ofrecido a una máquina virtual durante un período de SLA en nubes con diferentes tamaños y técnicas de redundancia y tolerancia a fallo. Finalmente, su trabajo propone el uso de un "presupuesto" SLA para la implementación de políticas en: 1) la asignación de servidores cuando las máquinas virtuales son ejecutadas y 2) el uso dinámico de diferentes licencias de tolerancia a fallos. Usando políticas que resultan en una considerable reducción de la probabilidad de violar las garantías del SLA, haciendo un uso eficiente de los recursos disponibles de la nube.

Por otro lado, Myerson (2013) ha identificado las mejores prácticas para desarrollar SLA para computación cloud, entre las que se destacan las siguientes recomendaciones:

1) Identificar los stakeholders cloud con sus responsabilidades bien definidas.
2) Evaluar las políticas del nivel de negocio.
3) Entender claramente SaaS, PaaS e IaaS y entender sobre qué tipo de nube se están ejecutando los servicios (privada, pública o híbrida).
4) Establecer claramente las métricas que serán usadas para alcanzar los objetivos de rendimiento.
5) Considerar los requisitos claves de seguridad de la nube. Por tanto, evaluar la sensibilidad, requisitos regulatorios y capacidades de seguridad de los proveedores cloud. Tener en cuenta que cada país tiene un conjunto distinto de leyes regulatorias de seguridad, razón por la cual los consumidores deberán conocer claramente en qué país los datos serán almacenados.

6) Identificar los requisitos de manejo de servicios. Esto incluye qué debe ser monitorizado, reportado y medido.
7) Preparar el manejo de los fallos de los servicios. Esto permite establecer que soluciones deben ser provistas (por ejemplo créditos por servicios mal provistos, conocidos también como penalizaciones). Establecer planes de recuperación de fallos, etc.

Comuzzi *et. al* (2013) discuten los problemas que pueden presentar las tecnologías cloud, tales como seguridad, interoperabilidad, entre otros, lo que actúa como una barrera para la adopción de éstas tecnologías en organizaciones. Estos problemas hacen necesario poseer mecanismos contractuales para articular las salidas deseadas de provisión de servicio y su comportamiento aceptable. Los autores presentan un trabajo que se enfoca en los mecanismos contractuales para este propósito y más específicamente en los SLA. Estos autores han conducido un estudio cualitativo entrevistando a los expertos de la industria para entender la manera en la cual obtuvieron una especificación detallada del SLA para diferentes modelos de despliegue de cloud, tanto para redes cloud públicas como privadas y en un marco de trabajo investigar el rol de los elementos del SLA en un proceso de toma de decisiones.

Finalmente Muller *et al. (*2014) discute ciertos problemas que presenta la gestión del SLA, tales como la falta de expresividad de los mismos para modelar escenarios realistas, la problemática que puede presentar la dificultad en la configuración de una especificación de un determinado SLA. La poca facilidad de entender y hasta de ser precisos al proponer un SLA o la baja cohesión entre sus elementos. Por tanto, ellos proponen una solución desde un modelo de referencia conceptual a los problemas antes mencionados, creando una plataforma que reciba un SLA convenido entre las partes como entrada y reporte las violaciones del SLA como salida, ejecutando una configuración de monitorización que cumpla ciertas restricciones. La plataforma propuesta la han denominado SALMonADA, la misma que ha sido evaluada demostrando un bajo nivel de consumo de recursos durante la provisión de los servicios.

A continuación, se presenta una tabla con un resumen de varios de los proyectos que se han llevado a cabo a nivel europeo, los mismos que han sido anteriormente citados y dejan patente la necesidad de soluciones que contemplen el cumplimiento del SLA y de los requisitos no funcionales involucrados con proyectos para *Cloud Computing*.

Tabla 3-2. Resumen de soluciones existentes que abordan el cumplimiento de los SLA

Acrónimo	Estándar Utilizado para Especificación	Modelo de Servicio	Objetivo
4CaaSt	Desarrolla un lenguaje propio de descripción llamado "blueprint" que facilita la definición de aprovisionamiento y reglas de negocio con respecto a la elasticidad y multitenencia, como también la inclusión de marcas para el desarrollador de la aplicación para mapear términos de aplicaciones de alto nivel en parámetros de bajo nivel.	PaaS	Soporta la optimización y alojamiento elástico de aplicaciones multi-capa. Integra características que facilitan programar aplicaciones enriquecidas para habilitar la creación de un sistema de negocios donde las aplicaciones de diferentes proveedores pueden ser ubicados a diferentes usuarios, mezcladas y puestas juntas.
Cloud 4 SOA	WSAgreement	PaaS	Provee un marco de trabajo interoperable para desarrolladores PaaS. El sistema soporta desarrolladores de aplicación con multi-plataforma. Habilita la negociación dinámica del SLA y herramientas que permiten analizar los ofrecimientos y rendimiento y adaptarse a los SLAs convenientemente. El framework permite a los proveedores y clientes negociar flexiblemente entre el estándar y un SLA personalizado mientras soporta dinámicas de negocios a través de métricas relacionadas al SLA que serán monitorizadas y analizadas
Cloud Scale	Scale DL (Scalability Description Language) el cual caracteriza los requisitos de escalabilidad de un servicio.	PaaS	Permite analizar, predecir y resolver problemas de escalabilidad en ambientes cloud. Identifica las causas de violaciones del SLA, cuando los servicios no escalan de la forma prevista
Cloud-TM	No especifica	IaaS	Permite reducir el desarrollo y costos operacionales de las aplicaciones basadas en la nube. El enfoque es realizado a través de un marco de trabajo que habilita la auto-optimización y auto-ubicación de los recursos de la infraestructura basándose en diferentes cargas de servicio.
Contrail	SLA(T) - SLA@SOI y extendió al uso de un estándar OVF para recursos virtuales	Paas	Ofrece servicios elásticos PaaS sobre una federación de nubes IaaS, mientras se encarga de problemas relacionados a la calidad de servicio, manejo SLA, seguridad, interoperabilidad y

			escalabilidad. Además, el proyecto ha desarrollado un modelo de calidad para capturar diferentes parámetros de interés para clientes y proveedores
Cumulo Nimbo	No especifica	PaaS	Este proyecto fue desarrollado para proveer alta escalabilidad sin sacrificar la consistencia de los datos y la fácil programación. (01-10-10/30-12-2013) 4.9 M Euros / 3.0 M Euros
EGI Federated Cloud Infrastructure	Plantillas SLA propias	IaaS	Desarrolla plantillas SLA para una instanciación fácil y automatizada para pequeñas comunidades de investigación que no puedan permitirse negociación de SLAs personalizados
ETICS	Plantillas de SLA propias (Business Enhanced SLA Templates)	IaaS	Entrega un nuevo control de red, manejo y planificación de tecnologías para mejorar la QoS a través de los proveedores de red, mejorando la cadena de valor
GEYSERS	SLA Management Framework propio	IaaS	El proyecto ha entregado mecanismos para coordinar y aprovisionar recursos de red y tecnológicos, para evitar las limitaciones de la segmentación de red, análisis de negocio, servicios y dependencias a través de un marco de trabajo de negocios y composición de infraestructuras lógicas siguiendo los recursos de infraestructura.
Helix Nebula	No especifica	IaaS	Establece una infraestructura cloud con multitenencia, multi-proveedor mientras identifica y adopta políticas de seguridad, confiabilidad y confianza. El proyecto establece la necesidad de un catálogo común con información básica para todos los proveedores.
IRMOS	Ha propuesto dos tipos de SLAs (SLA al cliente y SLA técnico)	SaaS, IaaS	Desarrolla soluciones Cloud que permiten la adopción de aplicaciones interactivas y de tiempo real, habilitando un conjunto de atributos y su integración eficiente en infraestructuras cloud. Traduce las especificaciones de calidad de alto nivel a especificaciones de bajo nivel.

MCN (Mobile Cloud Networking)	No especifica	SaaS	Desarrolla una plataforma de comunicación y aplicaciones, entregando a un sistema de redes móviles computación descentralizada y almacenamiento inteligente. Su principal objetivo es soportar SLAs de servicios compuestos y plataforma de aplicaciones, entregando un sistema para redes móviles.
MODA Clouds	No especifica	IaaS, PaaS	Provee métodos, un sistema de soporte de decisiones, un IDE de código abierto y un ambiente en tiempo de ejecución para el diseño de alto nivel, prototipado, generación de código semi-automático de aplicaciones con garantía de calidad de servicio
mPlane	SLA Measurement Definition para SLAs	IaaS	El proyecto clama en desarrollar un soporte inteligente de medición para Internet para recolectar y analizar mediciones en redes de larga escala.
OPTIMIS	Manifesto de Servicio que contiene un núcleo común, las extensiones del proveedor del servicio, las extensiones del proveedor de la infraestructura. Para ello extienden el WS-Agreement	No especifica	Habilita a las organizaciones para externalizar sus servicios y aplicaciones de una manera confiable y auditable por los proveedores cloud, mientras se optimiza el ciclo de vida completo de ingeniería de servicios, provisión, operación, entrega y uso.
Presto PRIME	Desarrollaron su propia especificación de SLAs con 21 capacidades, 12 características de interés, 15 métricas, 12 términos de calidad, 4 restricciones, 6 términos de facturación, 7 términos de penalización	SaaS	Desarrolla una infraestructura de manejo de servicios para la preservación a largo plazo de objetos digitales de audio y video, programas y colecciones. Los contratos entre clientes y proveedores son manejados a través de SLAs para la preservación de los servicios entre productores y clientes.
Q-ImPress	Desarrollan un marco de trabajo propio para el manejo de los SLAs que permite determinar el impacto y tradeoff de cada uno de los parámetros o atributos de calidad mientras considera diferentes estados	SaaS	El proyecto ha desarrollado un método para software dirigido por la calidad, y evolución, en donde las consecuencias de las decisiones de diseño y los recursos del sistema cambian en rendimiento, confiabilidad y mantenibilidad a través de un análisis de impacto y simulación
SERSCIS	Desarrollan una arquitectura en capas para el manejo de los SLA, es propia no expresan el lenguaje utilizado.	SaaS, IaaS	Desarrollado como una infraestructura orientada a servicios auto-adaptativa, para crear, monitorizar y manejar seguridad y sistemas de información altamente disponibles.

SLA@SOI	Desarrolla el lenguaje de especificación SLA(T), para la descripción de las características funcionales y no funcionales, usando una sintaxis abstracta	IaaS, PaaS, SaaS	SLA@SOI direcciona el ciclo de vida del SLA a través de una negociación autónoma, aprovisionamiento, monitorización y adaptación de SLAs, mientras se encarga de toda la pila de servicios desde los aspectos de negocio hasta la infraestructura física.
STREAM	Paraleliza consultas, no especifica el lenguaje de descripción de SLAs.	SaaS	Habilita el procesamiento de datos de una manera distribuida, habilitando paralelización y escalabilidad para operadores de consultas
VISION CLOUD	Proponen un lenguaje basado en lenguajes tradicionales de especificación, pero enriquecido	PaaS	Introduce una infraestructura poderosa para la entrega de servicios de almacenamiento confiables facilitando la convergencia de medios y telecomunicaciones

3.2.3. Discusión

A lo largo de esta sección se ha mostrado la importancia y el auge de las soluciones que apoyan a la detección de incumplimientos de los SLA y se ha visto que este tema ha sido ampliamente investigado en varios proyectos europeos. Estas investigaciones han dado lugar a la creación de marcos de trabajo, metodologías, lenguajes de especificación, etc., los mismos que buscan evaluar el cumplimiento de estos acuerdos así como también buscan una manera de automatizar los acuerdos de nivel de servicio que muchas veces están escritos inclusive en lenguaje natural. Para solucionar este problema, se han desarrollado algunos lenguajes de dominio específico que ayudan en la especificación de los SLA. Además, se han desarrollado arquitecturas que apoyan al cumplimiento de los SLA, teniendo como paso central la monitorización para poder medir el nivel del cumplimiento de los mismos.

En las siguientes secciones se discute con más en detalle las herramientas de monitorización existentes, y cómo muchas de estas soluciones están encaminadas a monitorizar la calidad del servicio como mecanismo para la detección de incumplimientos en las cláusulas de los SLA.

3.3. Métodos de monitorización de calidad

La monitorización de la calidad de los servicios en la nube es de gran importancia debido a los beneficios que presenta el contar con la información oportuna sobre la manera en la que se están aprovisionando los mismos. Estos beneficios se ven reflejados en las acciones preventivas y correctivas que pueden

tenerse cuando la información de los niveles de calidad del servicio es provista. Existen varias aproximaciones tanto de propósito general, como aquellas que permiten la monitorización de servicios en la nube y cada una de ellas presenta beneficios e limitaciones. Sin embargo, como se muestra en las secciones a continuación, las herramientas de monitorización de propósito general no cuentan con ciertas consideraciones necesarias para la monitorización de servicios pero han constituido una base sólida sobre la cual se han podido proponer herramientas específicas para monitorizar distintos tipos de artefactos de software.

3.3.1. Métodos y soluciones de monitorización de propósito general y computación distribuida

Antes de la aparición de la computación en la nube, un número considerable de herramientas han estado disponibles para monitorizar diversas infraestructuras de IT tales como redes y nodos de cómputo. Algunas se especializan en dominios particulares tales como HPC clústeres y Grids.

Según Fatema *et al. (*2014), las infraestructuras de monitorización de propósito general típicamente utilizan un modelo cliente-servidor, instalando un agente en cada sistema a ser monitorizado. Los agentes de monitorización miden los valores de las métricas de los componentes monitorizados y los envían al servidor de monitorización. El servidor almacena las métricas recolectadas en una base de datos, las analizan y envían las alertas. Esto puede generar grafos, reportes de tendencias y reportes de SLA. Cuando un agente de monitorización es iniciado para recolectar métricas, este mide los valores relevantes de las métricas desde los componentes monitorizados y las pasan sobre el servidor de monitorización. Dependiendo del sistema de configuración, el servidor puede enviar alertas a las partes interesadas en las ocurrencias de un evento.

Teniendo en cuenta los trabajos relacionados analizados por Povedano-Molina, *et al. (*2013), existen soluciones diseñadas para presentar infraestructuras grandes de IT como son los ambientes grid. Un ejemplo de ellas es "Nagios - The Industry Standard In IT Infrastructure Monitoring," (2012). Nagios es un servicio de monitorización de servidores muy popular utilizado en centros de datos para monitorizar el estado de servicios y recursos. Adopta una arquitectura cliente servidor en la cual el servidor central es el responsable de capturar la información de monitorización desde nodos remotos. Éste soporta dos principales alternativas para monitorizar recursos locales. Estos son el *Nagios Remote Plugin Executor* (Nagios NRPE)- instalado con cada recurso monitorizado- que habilita las actualizaciones y las interacciones disparadas por un servidor central; y el Nagios Service Check Adaptor (Nagios NSCA) el cual soporta

mediciones asíncronas y eventos desde nodos monitorizado el servidor central. Nagios adicionalmente es más orientado a la visualización del estado de servicio en lugar de a los cambios de monitorización continuos de los recursos del sistema, tales como memoria y CPU en escenarios altamente dinámicos. En adición, este no adopta ningún formato estandarizado para la representación de los datos de monitorización.

Ganglia Monitoring System, (2012) es otro sistema relevante ampliamente empleado para monitorizar grids y clústeres de computadoras. Para conseguir clústeres de monitorización escalable, Ganglia adopta una arquitectura distribuida: cada nodo ejecuta un agente llamado gmond, que envía información sobre el uso de los recursos a otro agente llamado gmetad, actuando como un agregador y ejecutando a un alto nivel de la jerarquía. Además, Ganglia permite la interacción pull y push entre agentes, y soporta los formatos XDR y XML.

Las infraestructuras para monitorizar grids distribuidas han comenzado a reconocer la importancia del descubrimiento dinámico. MonALISA (Legrand *et al.*, 2009) enriquece la arquitectura tradicional cliente-servidor vía la introducción de un registro centralizado usado para descubrir dinámicamente las entidades a ser monitorizadas; adicionalmente, este soporta la interacción pull y push, filtrado de datos y el formato de intercambio XDR. Hyperic HQ es una solución de monitorización desarrollada por Vmware Inc, específicamente diseñado para soportar el descubrimiento de recursos locales y servicios así como para monitorizar hosts que tienen que ser explícitamente diseñados y registrados.

Otro ejemplo de una arquitectura de monitorización es la presentada por Chung *et al. (*2009), un sistema de monitorización para grids con una alta precisión y con un bajo consumo de ancho de banda. Esta característica es alcanzada por ajustar el período entre mediciones de acuerdo al promedio de ejecución de intervalos de tiempo entre cambios de información. Este sistema además filtra las actualizaciones entregadas a consumidores de acuerdo a su valor: si estos cambios son significativos ellos no serán entregados, entonces se ahorran recursos.

Basados en una infraestructura súper-escalar (GRIDSs), Reyes, *et al. (*2010) presentan un sistema de monitorización para aplicaciones grid. Esta propuesta presenta algunas limitaciones en cuanto a la escalabilidad y se basa en un único servidor para procesar y almacenar la información de monitorización recogida por agentes.

Otra posible métrica a ser considerada en computación Grid es el cumplimiento de QoS en el proceso de ubicar recursos para la ejecución de

trabajos en el grid (Torres, *et al.*, 2011). En este trabajo, se menciona que muchas herramientas han sido desarrolladas con la finalidad de asignar los trabajos a los recursos más apropiados para las necesidades específicas de la aplicación mientras se balancea la carga de trabajo entre los recursos.

Sin embargo, pocos de ellos se centran en evaluar el nivel de servicio provisto por los servicios grid. Al contrario, grid generalmente ofrece el servicio del mejor esfuerzo a todas las aplicaciones. Sin calidad de servicio (QoS), los trabajos son ejecutados sin ninguna garantía sobre el tiempo de ejecución o rendimiento. Además, el soporte de atributos cualitativos, tales como la seguridad, está limitada a la información provista por los recursos. La falta de certeza sobre el rendimiento final es un problema que afecta a la satisfacción del usuario, reduciendo el interés hacia las infraestructuras Grid. En su trabajo estos autores presentan una arquitectura de software que cubre todos los pasos necesarios para integrar la QoS en el proceso de asignar recursos para la ejecución de trabajos en el grid.

Ben Hamida *et al. (*2012) proponen un marco de trabajo de monitorización multi-origen, el cual relaciona los mensajes pasados al nivel de negocio con observaciones relativas a los recursos de infraestructura. En su trabajo presentan la arquitectura de monitorización y un caso de estudio de una parte del Proyecto ChOReOS.

Los elementos de monitorización están enfocados en el conocimiento del estado del ambiente donde los servicios se ejecutan. En este sentido, estas fuentes proveen soporte para la monitorización de recursos, ambos en términos de estado de salud y utilización de los mismos. Además, utilizan una fuente responsable de monitorizar los mensajes intercambiados entre servicios cooperando ya sea con un flujo de trabajo o una coreografía, de tal manera que se analice la secuencia temporal de mensajes pasados a través del bus y tratar de ver las violaciones de la especificación de la coreografía con respecto a las violaciones de las funciones, QoS, o SLA. Además, como otra fuente de datos monitorizan los eventos, los mismos que pertenecen a una infraestructura genérica basada en eventos que permite dirigir las notificaciones que vienen de otras dos fuentes.

Por otra parte, monitorizan la infraestructura en términos de utilización y estado de salud. Esto permite al sistema ejecutar acciones correctivas para así mantener el uso óptimo de recursos, evitando tanto sobrecarga y gasto de recursos). El subsistema de monitorización de recursos que proponen estos autores tiene dos componentes principales:

1) Un conjunto de *recolectores de datos* que recolectan información local tal como el promedio de carga, tasas de entrada y salida y utilización de la red; y
2) Un *mecanismo de notificación* que detecta eventos potencialmente relevantes, tales como promedios excepcionales de carga o espacio de disco insuficiente, generando una notificación que es enviada al subsistema de monitorización.

Finalmente, estos autores hacen uso de un servicio de monitorización de negocios, el cual es responsable de proveer la funcionalidad de monitorización que relaciona los servicios del negocio y la coreografía. Este asegura un servicio de supervisión multi-nivel, desde la QoS a un control del flujo de los negocios, realizando una supervisión incremental desde un nivel de granularidad fino a uno grueso.

Como se puede observar en la literatura antes mencionada, existe un gran interés por la monitorización de servicios en ambientes distribuidos, lo que hace que se hayan creado una gran variedad de soluciones de monitorización. Estas soluciones hacen uso de agentes, paso de mensajes, entre otros. Se ha considerado importante dar un breve repaso a estas soluciones ya que posteriormente se dirigirá nuestra atención al cloud, siendo éstas un excelente punto de partida para ahondar en soluciones específicas que permitan monitorizar servicios en la nube.

En la Tabla 3-3 se muestra soluciones de monitorización de propósito general, las mismas que han sido estudiadas más a detalle en el trabajo de Fatema *et al. (*2014), se ha tomado de la tabla original las columnas más representativas a efecto de este trabajo.

Tabla 3-3. Herramientas de monitorización de propósito general (Fatema *et al.*, 2014)

Herram.	Recursos Monitorizados	Tecn/leng empleado	Agent OS	Limitaciones	Experiencia de Uso
Nagios	Recursos del sistema, red, sensores, aplicaciones y servicios	C	Linux / Unix (Windows vía agente proxy)	Dificultad de configuración, no se puede trabajar con una frecuencia alta de muestreo.	Monitorización de infraestructuras Grid, Monitorización de máquinas virtuales

Collectd	Recursos del sistema, sensores, bases de datos, aplicaciones y servicios	C	Linux/Unix y Mac OS	No incluye una plataforma de visualización	Recoge métricas desde la nube para propósitos forenses
Opsview Core	Recursos del sistema red, bases de datos, aplicaciones y servicios	Perl, C	Linux/Unix y Windows	N/A	N/A
Cacti	Principalmente redes	PHP, C	Linux y Windows	N/A	Desarrollo integrado de varias herramientas provistas por terceros
Zabbix	Recursos del Sistema, red, sensores, bases de datos aplicaciones y servicios	C, PHP, Java	Linux/Unix, Mac y Windows	Ineficiente en el auto-descubrimiento de servicios	Se ha probado en una máquina de un broker cloud al momento de escalar recursos Cloud dinámicamente
Open NMS	Principalmente redes	Java	Linux/Unix, Mac y Windows	Limitada capacidad de auto-descubrimiento de servicios, no generado para todos los servicios de red	N/A
Ganglia	Recursos del sistema	C, Perl, PHP, Python	Linux/Unix, Solaris, Windows, Mac	Dificultad en la personalización, sobrecarga el sistema y la red por las actualizaciones multi-cast que realiza y codificación de los eventos en XML	Datos del lado del servidor para monitorización en la nube y aprovisionamiento de recursos en Clústeres

Hyperic HQ	Recursos del sistema, redes, bases de datos, aplicaciones y servicios	Java	Linux, Unix, Windows y Mac	Requisitos muy altos de memoria y dificultad de personalización de interface gráfica	Para recolectar un inventario de software en CloudAlloc, un sistema de monitorización para clústeres
IBM Tivoli	Recursos de red, sistema, bases de datos, aplicaciones y servicios	Java	Linux/Unix, Windows, Mac	N/A	Monitorización de la nube de IBM en manejo de servicios de empresas
Kiwi Application Monitor	Procesos de aplicación y actividad de usuario	N/A	Windows	N/A	En medir picos de memoria en aplicaciones y tiempo de CPU en la simulación de una computadora cuántica eficiente
R-OSGi	Aplicaciones distribuidas	Java	N/A	Problemas al registrar el servicio y configuración estática	Recolecta información de dependencia de una aplicación distribuida
DAMS	Aplicaciones distribuidas	Java	N/A	Afecta el rendimiento del sistema	Manejo de un sistema de manejo de educación a distancia de una universidad

3.3.2. Métodos y soluciones de monitorización específicas para la nube

Como lo mencionan Aceto *et al. (*2013) en su estudio sobre las diferentes herramientas de monitorización para la nube, el número de servicios basados en la nube, se ha visto incrementado rápidamente en los últimos años. Ante ello, estos autores han visto la necesidad de realizar un detallado análisis de los trabajos que abarcan la monitorización de estos servicios.

Estos autores analizan y discuten las propiedades de un sistema de monitorización, los problemas que conllevan tales propiedades y como esos problemas han sido abordados en la literatura. Además describen plataformas actuales, tanto comerciales como de código abierto y servicios para monitorización de servicios en la nube, poniendo énfasis en cómo ellos relacionan las propiedades y problemas identificados.

Finalmente, estos autores discuten los retos y direcciones futuras en este campo. Uno de los retos mencionados es la necesidad de monitorización de servicios en la nube para medir continuamente y evaluar la infraestructura o el comportamiento de las aplicaciones en términos de rendimiento, confiabilidad, uso de energía, habilidad para cumplir con los SLA, seguridad, etc., según lo han manifestado Kutare *et al. (*2010), o para ejecutar análisis de negocios (Kumar *et al.*, 2006), capacidad y planificación de recursos (Aljohani *et al.,* 2011; Almeida *et al.,* 2002), capacidad y manejo de recursos (Katsaros *et al.*, 2011; Lakshmanan *et al.*, 2010; Shao *et al.*, 2010), manejo de los centros de datos (C. Wang *et al.*, 2011), gestión del SLA (Massonet *et al.*, 2011), facturación (Li *et al.*, 2010; Ngan *et al.*, 2012; Samimi *et al.*, 2011), manejo de rendimiento (Schad *et al.*, 2010), manejo de seguridad (Chen *et al.,* 2010; Rochwerger *et al.*, 2009; Spring, 2011), entre otros.

Katsaros *et al.* (2012b) presentan un sistema de monitorización que facilita la configuración en términos de los intervalos de monitorización y los parámetros de monitorización. El enfoque propuesto presenta un marco de trabajo de monitorización multi-capa para medir la calidad de servicio a nivel de aplicación e infraestructura. En este trabajo los autores se centran en ofrecer un mecanismo de monitorización escalable y adaptable requerido para adquirir, agregar y evaluar los valores de parámetros de monitorización críticos en un intento de proveer garantías de calidad de servicio en ambientes cloud. Además, estos autores comentan que las garantías de nivel de servicio dependen fuertemente de marcos de trabajo de monitorización y requiere una atención específica a nivel de aplicación e infraestructura.

Povedano-Molina *et al. (*2013) hablan en su trabajo sobre la importancia de conocer el estado y disponibilidad de los recursos físicos y servicios presentes en la infraestructura. Por tanto señalan que un amplio conocimiento y control de los estados actuales de esos recursos permiten a los administradores Cloud diseñar mejor las estrategias de aprovisionamiento y evitar las violaciones del SLA. Además señalan la dificultad de manejar tal información de una manera confiable y escalable, especialmente cuando se consideran ambientes cloud compartidos por algunos huéspedes y cuando se necesitan armonizar las diferentes necesidades de monitorización a diferentes capas de la nube.

Teniendo en cuenta ello, estos autores proponen una arquitectura distribuida para manejo de recursos y monitorización en la nube llamada DARGOS, y la describen como una arquitectura distribuida y altamente eficiente, que asegura la medición de recursos físicos y virtuales manteniendo un bajo nivel de sobrecarga.

Ellos además califican a su solución como flexible y adaptable a diferentes métricas; situándola a nivel físico y virtual, mientras proponen un control de la sobrecarga que se puede generar. Ellos se basan en un paradigma publicar/suscribir basado en los datos y especialmente diseñado para la monitorización de sistemas con multitenencia, permite personalizar la granularidad de los datos y la frecuencia de la monitorización de acuerdo a los requisitos de cada huésped. Además se basa en tecnologías de comunicación estandarizadas, y más precisamente en el servicio de distribución de datos (DDS). Sin embargo, estos autores abordan la confiabilidad, eficiencia, soporte multitenencia y alta escalabilidad mientras introducen una baja sobrecarga. Sin embargo, no permiten la especificación clara de métricas y no abordan la monitorización a nivel SaaS en detalle, situándose más a nivel físico.

Por otro lado, Alhamazani *et al. (*2014) presenta una visión de las herramientas comerciales de monitorización de servicios en la nube, dimensiones de investigación, problemas de diseño y estado del arte. Estos autores investigan los conocimientos fundamentales en la provisión de aplicaciones y conceptos de monitorización, identifican las principales dimensiones de investigación y problemas de ingeniería de software basados en los recursos cloud y tipos de aplicación y finalmente presentan las direcciones de investigación futura. Hablan sobre la importancia de la monitorización para mantener una alta disponibilidad y rendimiento del sistema y su importancia tanto para clientes como para proveedores, justificando esta aseveración con los trabajos de Chaves *et al. (*2011), Grobauer *et al.*, Walloschek *et al. (*2011) y Moses *et al. (*2011).

En la Tabla 3-4 se muestra una lista de herramientas para la monitorización en la nube, las mismas que han sido analizadas por Fatema *et al. (*2014).

Tabla 3-4. Herramientas de monitorización específicas para la nube (Fatema et al., 2014)

Herram.	Recursos Monit.	Port.	Escal.	Sistema Operativo	Lenguaje de Implementación	Limitaciones Reportadas
Amazon Cloud Watch	Recursos AWS (Amazon Web Services) y aplicaciones y servicios ejecutándose en AWS	No	Si	Linux, Windows	Scripts en Windows, Powershell para Windows y Perl para Linux	Trabaja con recursos para Amazon solamente, trabaja en modelos centralizados, no asegura disponibilidad, problemas potenciales de seguridad

Azure Watch	Recursos basados en Azure	No	Si	Windows	.Net	Trabaja para Azure basado únicamente en recursos
Nimsoft	Recursos varios en la nube, S.O. (Sistema Operativo), red y aplicaciones	Si	Si	Linux, Netware, Unix, Windows, Mac	C/C++, Java, Perl, VB y .Net	No soporta chequeos del SLA para durabilidad de datos o localización
Monitis	Red, servidor, recursos cloud y aplicaciones	Si	Si	Linux/Unix, Windows, Mac	C++	N/A
Cloud Kick	Recursos cloud, aplicaciones y protocolos	Si	No	Linux, Windows	C#, .NET, Python, PHP, Java y Ruby	Trabaja en modelos centralizados y no asegura disponibilidad de servicios, puede monitorizar solamente vía el agente Cloud Kick, solamente habilitado para monitorización de infraestructura
Boundary Application Monitor	Aplicaciones cloud	Si	Si	Linux, Windows, Mac	N/A	N/A
mOSAIC	Aplicaciones cloud	Si	No	Sistemas basados en Linux	Java	Monitoriza las aplicaciones desarrolladas con el API Mosaic únicamente
CASViD	Aplicaciones cloud	Si		Unix, Linux	Python, Java	No soporta monitorización multi-capa
PCMONS	Recursos cloud	No	No	Linux	Python, Perl, Bash script	Trabaja solamente para redes cloud privadas

Herramientas de Monitorización Comerciales: En el trabajo de Alhamazani *et al. (*2014) se muestra un estudio sobre varias herramientas de monitorización comerciales entre las que observa las siguientes:

- **Network & IT Systems Monitoring – Monitis, (2005):** Tiene un cuadro de mando unificado en el que el consumidor puede abrir múltiples widgets para monitorización. Un consumidor Monitis necesita ingresar sus credenciales para acceder a su cuenta en la nube. Además, un consumidor Monitis puede remotamente monitorizar cualquier sitio web, en servidores propios para determinar la carga del CPU, memoria o disco I/O, instalando agentes Monitis para recuperar datos sobre los dispositivos. Un agente Monitis puede además ser usado para recoger datos de dispositivos de red (detrás de un firewall). Esta técncia es usada en lugar de instalar a un agente Monitis en cada dispositivo. Los widgets pueden ser enviados además como de solo lectura para compartir la información de monitorización. Monitis provee características enriquecidas para reportar el estado de las instancias en donde los consumidores pueden especificar el modo de que un reporte debe ser visualizado como por ejemplo a través de un cuadro o de un gráfico. Esto además habilita a los consumidores a compartir el reporte con otros consumidores.

- **LogicMonitor, (2008):** es una solución que funciona con NetApp, VMWare, Dell, HP. Facilita a sus consumidores el monitorizar a través de las capas de la nube (IaaS, PaaS, SaaS). Éste además permite las operaciones de aplicación de la monitorización sobre recursos multi-cloud. Utiliza el protocolo SSL. Además, LogicMonitor usa SNMP como un método de recuperar datos sobre recursos virtuales o físicos distribuidos.

- **Nimsoft, (2011):** soporta monitorización en multi-caas para recursos físicos y virtuales. Además, Nimsoft habilita a sus consumidores vistas y moniotriza sus recursos en caso de que ellos estén alojados en diferentes infraestructuras cloud. Por ejemplo, Nimsoft habilita a sus clientes la vista y monitoriza sus recursos en caso de que estén ubicados sobre diferentes infraestructuras cloud.

- **Nagios, (2007):** soporta una monitorización multi-capa. Habilita a sus consimidores a moniotrizar sus recursos sobre diferentes infraestructuras cloud como también infraestructuras internas. Ngios utiliza SNMP para monitorizar recursos de red. Además, Nagios ha sido

extendido con funcionalidades de monitorización para instancias tanto virtuales como servicios de almacenamiento usando una arquitectura basada en plugins. Típicamente un servidor Nagios recolecta información de monitorización, el cual podría ubicarse como una solución centralizada. Además, es posible usar configuraciones Nagios para crear múltiples servidores para reducir las desventajas de un servidor centralizado.

- **SPAE, (2002):** SPAE es una herramienta desarrollada por SHALB provee una solución de monitorización llamada *Security Performance Availability Engine (SPAE)*. Es una herramienta típica de monitorización de red que soporta una variedad de protocolos tales como HTTP, HTTPS, FTP, SSH, etc. Esta usa SNMP para ejecutar todos sus procesos de monitorización y enfatiza en la monitorización de la seguridad y vulnerabilidaes. Sin embargo, SPAE no soporta monitorización a diferentes capas (IaaS, PaaS y SaaS). Esto habilita a sus consumidores a monitorizar recursos de red incluyendo infraestructura cloud.

- **Amazon Cloud Watch, (2011):** es una de las herramientas comerciales más populares para monitorizar la nube. Esta es provista por Amazon para permitir la monitorización de los recursos residentes en EC2 de sus clientes. No soporta la monitorización de infraestructuras multi-cloud. Los enfoque técnicos utilizados en CloudWatch para recolectar datos están implicitos y no expuestos a los usuarios. Esta herramienta es limitada a monitorizar recursos a través de las capas de red. Soin embargo una API es provista a los usuarios para recolectar métricas a cualquier capa de red, sin embargo requiere que los usuarios escriban código adicional.

- **OpenNebula, (2010):** Es un sistema de monitorización de código abierto que provee manejo para centros de datos. Este usa SSH como el protocolo para permitir a los consumidores ganar acceso y recoger información sobre sus recursos. Principalmente, OpenNebula está relacionado a infraestructuras de monitorización físcias envueltas en centros de datos, tales como nubes privadas.

- **CloudHarmony, (2010):** Comenzo a monitorizar recursos a inicio de 2010. Esta herramienta provee un conjunto de puntos de referencia de rendimiento de redes públicas. Este está principalmente preocupado en moniotrizar las étrcias comunes del sistema operativo que están relacionadas a (CPU, disco y meoria). Además, el rendimiento de red de nube a nube es evaluado en térmnos de RTT y rendimiento.

- **Windows Azure FC:** Azure Fabric Controller adopta una arquitectura de red centralizada. Es un sistema multicapa pero no soporta monitorización a través de diferentes infraestructuras Cloud. Además, Azure FC utiliza SNMP para ejecutar la monitorización.

Por otro lado, en la revisión sistemática de Alhamazani *et al. (*2014), se presentan los componentes básicos que pueden ser considerados como dimensiones de evaluación para evaluar una herramienta de monitorización para computación en la nube, las dimensiones son las siguientes:

- **Arquitecturas de Monitorización:** en la monitorización en la nube, la red y la información relacionada al sistema es recolectada por los sistemas de monitorización. Por ejemplo, la utilización del CPU, retardo en la red y pérdida de paquetes. Esta información es luego usada por las aplicaciones para determinar acciones tales como migración de datos al servidor más cercano al usuario para asegurar que los requisitos del SLA son reunidos. Típicamente, la monitorización en la red pueden ser ejecutados sobre arquitecturas de redes centralizadas y de-centralizadas. En las arquitecturas centralizadas los recursos PaaS e IaaS envían actualizaciones de la calidad de servicio a un servidor monitorizado de monitorización. En el trabajo de Anand, (2012), los autores muestran que una arquitectura de monitorización centralizada permite un mejor manejo para aplicaciones en la nube. Sin embargo, los enfoques centralizados presentan los siguientes problemas:

  - Propensión a un simple punto de fallo.
  - Falta de escalabilidad
  - Costos altos de comunicación de red (cuellos de botella, congestión, etc.) y
  - Posible falta de poder computacional requerido para servir un gran número de requisitos de monitorización.

  Por otra parte, las herramientas de descentralización de las herramientas de monitorización en la nube han ganado momento. La descentralización de herramientas de monitorización pueden resolver los problemas relacionados a los actuales sistemas centralizados. Una herramienta de configuración de monitorización es considerada como descentralizada si ninguno de los componentes en el sistema es más importante que otro. En caso de que uno de los componentes falla, este no influencia la operación de los otros componentes en el sistema.

  - Punto a punto estructurado: Busca tener una capa de red donde una autoridad central es desactivada y ha conllevado a una red

estructurada punto a punto. En tal red, un punto central de falla es eliminada. Naptser (Davis *et al.*, 2008) es un sistema punto a punto estructurado.

- Punto a punto no estructurado: Las redes punto a punto no estructuradas cuya diferencia está que el directorio de búsqueda, el cual no está centralizado a diferencia de las redes punto a punto estructuradas Gnutella es una bien conocida solución de sistemas punto a punto no estructuradas (Davis *et al.*, 2008).
- Redes híbridas punto a punto: Es una combinación de sistemas de red estructuradas y no estructuradas. Los súper pares pueden actuar como hubs de búsqueda central en pequeñas porciones de la red, en donde el ámbito general de la red pertenece a un sistema no estructurado punto a punto. Kazaa es un híbrido entre un sistema de red Napster centralizado y un Gnutella descentralizado.

- **Interoperabilidad:** La perspectiva de interoperabilidad en la tecnología se centra en las capacidades técnicas del sistema de interactuar entre organizaciones y sistemas. Esta además se centra en la misión resultante de compatibilidad o incompatibilidad entre los sistemas y sus pares. Las aplicaciones de negocios modernas desarrolladas sobre la nube son a menudo complicadas y requieren interoperabilidad. Por ejemplo, un propietario de la aplicación puede desplegar un servidor web sobre la nube de Amazon mientras el servidor de la base de datos puede estar alojado en Microsoft Azure. A menos que los datos y las aplicaciones no estén integradas a través de las nubes de forma correcta, los beneficios del alojamiento en la nube no podrán ser conseguidos. La interoperabilidad además es necesaria para evitar bloqueos en el sistema.

  En cuanto a la interoperabilidad, esta revisión sistemática presenta el siguiente cuadro de herramientas comerciales:

Tabla 3-5. Herramientas de Monitorización y su Interoperabilidad (Alhamazani et al., 2014)

Plataforma	Interoperabilidad, Cloud-Agnostic (Multi-Clouds)
Monitis	Si
RevealCloud - IDERA	Si
LogicMonitor	Si
Nimsoft	Si

Nagios	Si
SPAE	Si
CloudWatch	No
OpenNebula	No
CloudHarmony	Si
Windows Azure FC	No

Esta dimensión se refiere a la habilidad de un marco de trabajo de monitorización para monitorizar aplicaciones y sus componentes que pueden estar desplegados sobre múltiples proveedores de cloud.

- **Calidad de Servicio (QoS):** no es trivial para desarrolladores de aplicacones entender qué parámetros de calidad de servicio y objetivos se necesitan para especificar y monitorizar a través de cada capa de una pila cloud incluyendo PaaS (p. ej. servidores web, servidor de streaming, servidor de indexación, etc.) e IaaS (p. ej. servicios de cómputo, almacenamiento y red)

- **Monitorización a través de varias capas:** Los componentes de la aplicación (servidor de streaming, servidor web, servidor de indexación, servicio de cómputo, servidor de almacenamiento y red) relacionados a múltiples aplicaciones de streaming son distribuidas a través de capas en la nube, incluyendo PaaS e IaaS. Para garantizar la consecución de los objetivos de calidad de servicio para las aplicaciones multimedia, es crítico monitorizar los parámetros de QoS a través de múltiples capas Nathuji *et al. (*2010). De ahí, el reto es desarrollar herramientas de monitorización que puedan capturar y razonar acerca de los parámetros de QoS de componentes de aplicación a través de las cpas IaaS y PaaS.

- **Interfaces de Programación:** Las interfaces de programación permiten el desarrollo de sistemas de software para habilitar la monitorización a través de diferentes capas de la pila cloud. Esto envuelve algunos componentes tales como PAIs, widgets y la línea de comando para habilitar al consumidor a moniotirzar algunos componentes de los sistemas complejos cloud de una manera unificada.

Finalmente, el trabajo de Grati, *et al. (*2012) propone un marco de trabajo para monitorización de servicios Web compuestos usando el proceso BPEL y desplegándolo en un ambiente cloud. El marco de trabajo de estos autores está compuesto de tres módulos básicos para recolectar información de alto y bajo nivel, analizar la información recolectada y tomar acciones correctivas si existe una violación al SLA.

Este marco de trabajo además no modifica ni la implementación del cliente ni la del servidor. Además los autores presentan una prueba de concepto en la medición del tiempo de respuesta. Esta solución se llama QMoDeSV en donde la información es recolectada mirando localmente cada servicio compuesto. Luego, basado en los patrones de composición del servicio compuesto, la información de QoS es computada. Esta información es usada por nuestro marco de trabajo para detectar violaciones del SLA. Además esta solución clama por ser útil a los proveedores de servicios quienes luego pueden tomar acciones correctivas para mejorar sus servicios y evitar penalizaciones.

3.3.3. Discusión

La gran variedad de tecnologías y plataformas proveedoras de servicios cloud ha tenido como consecuencia que las herramientas se especialicen en una plataforma en concreto y es difícil encontrar una solución multiplataforma. Por una parte, la necesidad de los proveedores de ofrecer un servicio de confianza y calidad a los consumidores ha producido que por su parte los proveedores de servicios en la nube ofrezcan herramientas de monitorización propias especializadas en las características de sus plataformas, lo que muchas veces plantea problemas de interoperabilidad y dificulta la monitorización de servicios en diferentes plataformas.

Las características de la computación en la nube también dificulta la monitorización. La elasticidad es una de las propiedades más interesantes de la computación en la nube y a su vez es la que más difícil hace el proceso de monitorización, sobre todo en los casos que la monitorización es llevada a cabo por agentes. Dado que un agente es asociado a un recurso virtual como una máquina virtual que provee un servicio, cuando estas instancias se duplican no solo es un problema la identificación de cada una de las máquinas, sino que la infraestructura de monitorización debe adaptarse (lanzando más agentes o aumentando la carga de agentes libres con las tareas asociadas a esas nuevas máquinas virtuales). A su vez es complicado establecer que recursos físicos está empleando cada máquina virtual y es necesario establecer correlaciones entre los datos aportados por distintas máquinas virtuales. El problema de la elasticidad y la monitorización es tratado por Moldovan *et al.* (2013) ofreciendo distintas soluciones a nivel de arquitectura y de funciones de cálculo sobre estos datos.

Adicionalmente, poco se habla del grado de flexibilidad que una herramienta de monitorización debe poseer cuando la misma está orientada a dominios cambiantes de requisitos.

Las herramientas analizadas al respecto de la monitorización de servicios en la nube destacan el esfuerzo realizado por encontrar métodos de extracción de

la información de los servicios eficientes y precisos, tratando de salvar los obstáculos presentados por la tecnología en la nube. En su mayoría optan por las técnicas basadas en agentes por ser las que más datos pueden extraer del servicio.

En su mayoría centran la atención en la extracción de datos de los niveles de infraestructura y plataforma de bajo nivel y se deja en manos de otras aplicaciones o del usuario el análisis de estos datos con el fin de establecer correspondencias entre atributos de calidad (elasticidad, seguridad, fiabilidad) y datos crudos procedentes del servicio. Sin embargo, muchas herramientas ya disponen de la posibilidad de incorporar métricas personalizadas al proceso de extracción de datos, pero sin embargo, estas deben ser en muchas ocasiones implementadas por el usuario en el propio servicio, incapaces de aprovechar la infraestructura de monitorización más que para, la nada trivial, tarea de comunicar estos datos con un servidor de monitorización.

Todas estas limitaciones ponen de manifiesto la necesidad de una solución que permita monitorizar servicios en la nube que sea flexible, interoperable y que soporte cambios en los requisitos de monitorización o en el cálculo de las métricas utilizadas para medir la calidad de los servicios.

3.4. Soluciones que utilizan modelos en tiempo de ejecución para la monitorización

Los modelos en tiempo de ejecución pueden ser usados para monitorizar un sistema, simulando futuras evoluciones a través del análisis del impacto a nivel del modelo y predecir las propiedades del sistema tales como rendimiento y confiabilidad en caso de reconfiguración del mismo. Los modelos habilitan la monitorización, simulación y predicción a un nivel alto de abstracción (cercano al espacio del problema), direccionando el problema de baja abstracción resultante de los artefactos escritos a mano (Szvetits *et al.*, 2013).

Los autores Szvetits *et al. (*2013) han realizado una revisión sistemática en la cual han analizado el uso de los modelos en tiempo de ejecución. En esta revisión ellos han estudiado los diferentes enfoques existentes en relación con los objetivos, técnicas, arquitecturas y tipos de modelos usados actualmente en el campo de los modelos en tiempo de ejecución. De este estudio, en esta sección, se enfoca el uso de los modelos en tiempo de ejecución para la monitorización de diferentes tipos de sistemas.

El primer artículo incluido en su revisión sistemática con respecto al uso de los modelos en tiempo de ejecución es el trabajo de Le Duc *et al. (*2010) en el

cual los autores proponen un marco de trabajo para monitorización auto-adaptativa llamado ADAMO que tiene en cuenta diferente información de calidad sobre cadenas de datos dinámicas, además engloba el tiempo de vida, precisión, granularidad y tipos de datos de monitorización. El marco de trabajo permite acceder a cadenas de datos dinámicos para múltiples clientes con diferentes necesidades de calidad como también la generación y configuración de elementos apropiados de acuerdo a restricciones de calidad. ADAMO se basa en un modelo de calidad el cual formaliza las fuentes de datos, consultas de monitorización y recursos de sistemas y usa la resolución de restricciones para configurar las fuentes de datos de acuerdo a las necesidades del cliente y restricciones de recursos.

Por otro lado existe la herramienta SM@RT (Supporting Models at RunTime) utilizada por varios enfoques (Song *et al.,* 2010; Song *et al.*, 2009, 2011) para mantener la conexión causal entre el sistema y el modelo de la arquitectura. En esta herramienta, los cambios del sistema dirigen los cambios de la arquitectura en un modo bidireccional. Esto es obtenido por medio de la creación de una infraestructura de una arquitectura en tiempo de ejecución, sin modificar el sistema objetivo a través de transformaciones de modelos. El modelo de arquitectura resultante habilita la monitorización y control del sistema a un alto nivel de abstracción. La transformación bidireccional es además usada en los enfoques de (Giese *et al.*, 2006, 2009; Giese *et al.*, 2012; Hebig *et al.*, 2012) y (Vogel *et al.*, 2010; Vogel *et al.*, 2011; Vogel *et al.*, 2009) donde Gramáticas de Triple Grafo (TGG) son usadas para soportar la adaptación y la monitorización arquitectural.

En el enfoque TGG, un modelo fuente (bajo nivel, para monitorizar o adaptar el sistema) está causalmente conectado a uno o más modelos objetivos (alto nivel, especificando vistas en el sistema) mientras la sincronización es especificada declarativamente por las reglas TGG a nivel de meta-modelo para los modelos fuente y objetivo.

Lehmann *et al. (*2010) describen el meta-modelado de modelos en tiempo de ejecución y presenta un proceso de meta-modelado. Un enfoque similar de tres capas es propuesto por Cheng *et al. (*2002) y Garlan *et al. (*2001):

- Una capa en tiempo de ejecución observa las propiedades del sistema y ejecuta operaciones de adaptación a bajo nivel
- Una capa de modelo interpreta los datos observados con la ayuda de modelos de arquitectura analizable y chequea violaciones de restricciones, y
- Una capa de tarea determina los requisitos de QoS como una representación de las necesidades computacionales.

En esta propuesta, los modelos computacionales son representados como grafos de componentes que interactúan, incrementando la monitorización a un alto nivel de abstracción.

Por otra parte, Bertolino *et al. (*2011) proponen un enfoque dirigido por propiedades para monitorización en tiempo de ejecución que está basado en el Meta-modelo de Propiedades (PMM) y en una infraestructura genérica configurable de monitorización.

Esta solución soporta la definición de propiedades cuantitativas y cualitativas de una manera procesable por el ordenador haciendo posible configurar la monitorización dinámicamente, su solución sin embargo, representa una herramienta de monitorización de propósito general, la misma que si bien utiliza un meta-modelo, este no está basado en estándares existentes, además su propuesta consta de cinco pasos hacia la monitorización que son i) la recolección de la información, ii) la interpretación local de la información, iii) la transmisión de datos, iv) las funciones de agregación y v) los reportes.

Shao *et al. (*2010) abordan el tema de la importancia de mejorar la calidad del servicio en ambientes cloud. Ya que esto ayuda a escalar la utilización de los recursos de manera adaptativa, para identificar los defectos en los servicios y para identificar patrones de uso de numerosos usuarios finales.

Sin embargo, debido a la heterogeneidad de los componentes en la nube y a la complejidad de la riqueza de información en tiempo de ejecución, los autores proponen un modelo en tiempo de ejecución para monitorización en la nube denominado RMCM. Además, han utilizado su modelo en un marco de trabajo flexible de monitorización, el cual puede conseguir un balance entre la sobrecarga en tiempo de ejecución y las capacidades de monitorización vía el manejo adaptable de la misma. Su contribución se basa en los siguientes objetivos:

- Proponen un modelo en tiempo de ejecución para monitorización cloud (RMCM), la cual provee una representación intuitiva de una nube ejecutándose centrada en los principales objetivos de la monitorización.
- Basados en su modelo (RMCM), implementan un marco de trabajo de monitorización, el cual clama por conseguir un balance entre la sobrecarga en tiempo de ejecución y las capacidades de monitorización vía manejo adaptativo de las facilidades de la misma.
- Finalmente, los autores aplicaron enfoque de monitorización basado en el modelo en tiempo de ejecución en una nube real.

Los autores en su solución identifican que un cambio en el sistema puede ser reflejado en el modelo sin necesidad de manipular los datos crudos, dándose así la conexión causal propia de los modelos en tiempo de ejecución. Además los autores miran la calidad desde diferentes roles: i) operadores cloud, ii) proveedores de servicios cloud y iii) usuarios finales.

La diferencia con la solución presentada a lo largo de este trabajo, es que ellos no se centran en la calidad como tal y por tanto no definen métricas, ni aspectos de calidad específicos, sino más bien la actividad del usuario y su comportamiento; los autores hablan sobre la monitorización de requisitos no funcionales considerándolos como restricciones pero no definen la manera en la cual se deben expresar dichas restricciones, por lo que no queda del todo claro cuáles son las limitaciones en cuanto a la monitorización de la calidad, ni hablan de estándares ni lenguajes en los cuáles basarse para monitorizarla.

Bai *et al.*. (2009), hablan sobre la necesidad de la monitorización para la evaluación de la calidad de servicios web. Además, recalcan que un sistema de monitorización que posea sensores con políticas es ampliamente usado para recolectar datos en tiempo de ejecución, detectar anomalías en el comportamiento y generar alertas.

Los sensores codificados son caros de desarrollar y mantener y ellos no son flexibles ante cambios. Su propuesta trata de un enfoque dirigido por modelos para facilitar la generación automática de sensores basados en dependencias y sus estrategias. Su solución está enfocada a los servicios web y habla de la utilización de los modelos para generar sensores de interface y sensores de integración.

La propuesta de Bai *et al.* (2009), sin embargo, no trata de una monitorización de la calidad de servicios que utiliza métricas para medirla y presentar su estado, sino de que éstos no muestren un comportamiento anormal durante su ejecución. Las métricas aquí no pueden ser del todo expresadas y no permite expresar los requisitos no funcionales haciendo uso de un estándar o lenguaje de representación utilizado dentro del ámbito de la calidad del software.

3.4.1 Discusión

Como se puede concluir de los trabajos relacionados, los modelos en tiempo de ejecución están tomando fuerza y representan una solución óptima ante sistemas en continuo cambio, sin embargo, entre las propuestas presentadas por los diferentes autores, no existe una solución de monitorización que utilice modelos en tiempo de ejecución y que hagan uso de estándares que permitan

modelar de una manera sistemática y adecuada las características de calidad de los servicios cloud.

Por otro lado, dadas las características propias de la nube (elasticidad, modo de facturación, etc.), hace falta un sistema que permita la monitorización de sus servicios y de los SLA a ellos asociados.

Así también cabe destacar que los SLA, pueden verse modificados frecuentemente debido a renegociaciones entre los diferentes actores (clientes, proveedores) y que por tanto, el sistema de monitorización debería ser capaz de acoplarse a dichos cambios de una manera fácil y sin requerir mayores modificaciones.

De todo ello, se desprende que los modelos en tiempo de ejecución pueden proporcionar una solución precisa para la monitorización de servicios cloud, dado el alto nivel de flexibilidad que éstos presentan.

3.5 Conclusiones

Con relación a los requisitos de calidad que deben satisfacer los servicios desplegados en la nube, éstos se han abordado mediante una revisión de los modelos de calidad para SaaS y de las características específicas de la computación en la nube. Luego, se ha realizado un análisis de las aproximaciones existentes para soportar el modelado y verificación de los Acuerdos de Nivel de Servicio (SLA), haciendo un hincapié en los lenguajes de dominio específico.

Las principales conclusiones del análisis con respecto a los modelos de calidad existentes son que si bien existen varios modelos de calidad propuestos para los servicios cloud, ninguno contaba con una recopilación completa de los atributos de calidad con sus respectivas métricas y que a su vez esté alineado con la ISO/IEC 25010, problema que ha sido abordado por Navas (2016), quien ha propuesto un modelo de calidad que solventa el problema antes mencionado. Además, con la solución propuesta por Navas (2016), se evalúa la calidad interna/externa y en uso de los servicios cloud, lo que supera otro de los problemas existentes en trabajos previos.

En cuanto a los lenguajes de dominio específico, las principales conclusiones de nuestro análisis son: que si bien existen lenguajes de dominio específico como el WS-Agreement para el manejo de los SLA, éste se basa en un conjunto de restricciones que permiten calcular las métricas, siendo difícil integrar estas restricciones en el modelo en tiempo de ejecución. De ahí que se ha utilizado en este trabajo el WSLA, dado que éste permite mediante XML especificar los

operandos y definir la función de medición de las métricas de una manera más comprensible para el modelo en tiempo de ejecución.

Por otro lado, es también importante resaltar la atención que actualmente tiene la comprobación del cumplimiento del SLA, dado que las aproximaciones de desarrollo de servicios y dentro de éstas la computación en la nube está atrayendo cada vez a más organizaciones quienes buscan una manera de obtener servicios de software con alta disponibilidad, inversión baja, mínima necesidad de mantenimiento y elasticidad, características que permiten que el software se adapte fácilmente al crecimiento y demanda de la organización.

Se han estudiado también las características de calidad deseables en los sistemas de monitorización, sus niveles de abstracción, las diferencias entre la monitorización en grid computing y cloud computing, y por último, se han analizado las aproximaciones y herramientas existentes para la monitorización de servicios en la nube. Estos estudios nos han permitido concluir que la computación en la nube tiene sus propias características y necesidades de monitorización y que si bien existen varias herramientas que permiten evaluar la calidad de los servicios, la mayoría de soluciones están enfocadas en características y atributos cuyas métricas son estáticas y pre-establecidas por quienes han desarrollado la solución de monitorización, demostrando poca versatilidad ante cambios en la manera de realizar los cálculos y evaluar la calidad de los servicios.

En su mayoría centran la atención en la extracción de datos de los niveles de infraestructura y plataforma de bajo nivel y se deja en manos de otras aplicaciones o del usuario el análisis de estos datos con el fin de establecer correspondencias entre atributos de calidad (elasticidad, seguridad, fiabilidad) y datos crudos procedentes del servicio. Sin embargo muchas herramientas ya disponen de la posibilidad de incorporar métricas personalizadas al proceso de extracción de datos, sin embargo estas deben ser en muchas ocasiones implementadas por el usuario en el propio servicio, incapaces de aprovechar la infraestructura de monitorización más que para, la nada trivial, tarea de comunicar estos datos con un servidor de monitorización.

Finalmente, se han analizado los modelos en tiempo de ejecución y sus aplicaciones, entre las cuales están temas relacionados a predicciones, auto-adaptabilidad, aplicaciones en ambientes altamente variables y heterogéneos, localización de errores y monitorización. Si bien en el trabajo de Bertolino et al. (2011) ya se propone el uso de los modelos en tiempo de ejecución para tareas de monitorización, ésta solución es de carácter más general, sin efocarse a las particularidades de la computación en la nube. De ahí que se han utilizado los

modelos en tiempo de ejecución como una solución adecuada para la verificación del cumplimiento de los SLA en ambientes cloud y la evaluación de la calidad a través de un método de monitorización. Así, la utilización de los modelos en tiempo de ejecución permite solucionar las limitaciones de flexibilidad y baja interoperabilidad de las soluciones de monitorización actuales. Además, con su utilización no hace falta realizar implementaciones adicionales o modificaciones significativas en la infraestructura de monitorización. En este trabajo, el modelo en tiempo de ejecución ha sido utilizado para establecer las correspondencias entre los requisitos de monitorización (cláusulas del SLA), con las características, atributos de calidad y métricas independientes de la plataforma, con los contadores específicos de la plataforma cloud que permite extraer los datos del servicio en tiempo de ejecución y operar con ellos. Por lo tanto, el modelo en tiempo de ejecución contiene las configuraciones de monitorización que son consumidas por el motor de monitorización y análisis.

Por otro lado el trabajo de Cianciaruso et al. (2015) presenta una aproximación de monitorización que hace uso de modelos en tiempo de ejecución, los autores analizan ciertos atributos de calidad con la finalidad de autoadaptar los servicios cloud, sin embargo ellos no utilizan modelos que permitan dirigir el proceso de configuración de los requisitos no funcionales para su posterior monitorización, ni tampoco permiten capturar dichos requisitos de los SLA con la finalidad de verificar su cumplimiento. Todo esto dentro de su propuesta de un marco de trabajo CloudMF propuesto por Ferry et al. (2014), en el cual los autores presentan su porpuesta en dos fases principales: 1) la construcción de un ambiente de modelado soportado por un lenguaje para modelar el aprovisionamiento y despliegue de servicios en sistemas multi-cloud y 2) un ambitente de modelos en tiempo de ejecución que permita el aprovisionamiento, despliegue y adaptación de los servicios en dichas plataformas. Además, en el trabajo de Almeida *et al.* (2015), se muestra la necesidad de considerar un tratamiento dinámico de la información relevante durante la especificación de requerimientos. Aquí, los datos asociados pueden ser recolectados de la ejecución de una aplicación y mostrar cómo se cumplen los requisitos no funcionales establecidos. En su trabajo, ellos presentan una infraestructura dinámica, para soportar tanto la representación de los requisitos no funcionales, como la monitorización y posterior satisfacción de requisitos en tiempo de ejecución. Estos autores hacen uso de lenguajes de dominio específico para representar los requisitos a ser monitorizaods y una infraestructura de monitorización que evalúa dichos requisitos en tiempo de ejecución, finalmente seleccionan la configuración más adecuada, basándose en el grado de satisfacción de los RNF. En su propuesta, además enfatizan que su solución

puede ser utilizada en cualquier dominio, aunque en su evaluación abordan aplicaciones desplegadas en la nube. Su trabajo sin embargo no utiliza como base un lenguaje de dominio específico encaminado al cumplimiento del SLA y no utiliza un modelo de calidad específico y basado en un estándar, ya que se orienta a todo tipo de aplicaciones, buscando a través de un proceso de toma de decisiones la mejor configuración disponible para evaluar el grado de satisfacción del usuario de acuerdo a la configuración seleccionada.

Consecuentemente, la utilización de los modelos en tiempo ejecución abre la puerta a la auto-adaptación de los parámetros de monitorización según el estado de los servicios o aplicaciones, por ejemplo, en lo referente a la frecuencia de captura de datos, posible sobrecarga innecesaria generada por la herramienta de monitorización, entre otros. Siendo en este caso el modelo en tiempo de ejecución una solución precisa para establecer la causalidad desde el servicio hacia el modelo para así poder afinar los parámetros de monitorización.

Capítulo 4

Cloud MoS@RT: un Método para la Monitorización de Servicios Cloud

En este capítulo se presenta Cloud MoS@RT (*Monitoring Services @Run.Time*), un método de monitorización de servicios cloud que utiliza modelos en tiempo de ejecución. Cloud MoS@RT permite monitorizar servicios en la nube y proporciona una serie de ventajas con respecto a otras soluciones, tales como flexibilidad, modificabilidad al momento de realizar la monitorización de nuevos requisitos no funcionales e interoperabilidad con otras herramientas a través de distintos escenarios de recolección de datos.

La sección 4.1 describe, en forma general, el proceso de monitorización de servicios en la nube, sus actividades principales y artefactos de entrada y salida.

La sección 4.2 describe las actividades que se contemplan dentro del proceso de monitorización, especificándolas en detalle en las sub-secciones correspondientes.

Finalmente la sección 4.3 presenta las conclusiones del capítulo.

4.1. Definición del método de monitorización

El método de monitorización propuesto consiste en tres actividades principales, cada una de las cuales está dividida en tareas específicas. Este método de monitorización ha sido presentado en Cedillo *et al. (*2014) y refinado a lo largo del presente trabajo de investigación. El método está basado en la técnica del bucle autonómico de control, cuyo principio es medir los parámetros del sistema, analizarlos, planear acciones correctivas si es necesario y ejecutar esas acciones para mejorar el sistema.

Otro beneficio de esta técnica es la necesidad reducida de intervención humana, que a menudo conlleva a problemas de mantenimiento, reusabilidad y baja abstracción (Szvetits *et al.*, 2013). Aquí, se explicará el método hasta la tarea de análisis de resultados, el cual provee un reporte de incumplimientos del SLA, y en investigación futura, se conectará el middleware de monitorización, con un middleware de reconfiguración para cumplir el bucle autonómico de control. Las actividades que comprenden nuestro enfoque están presentadas en la Figura 4-1, para la presentación de las actividades involucradas, se ha hecho uso de SPEM 2.0 (*Software & Systems Process Engineering Meta-Model Specification*) (OMG, 2008),

que permite definir los procesos de desarrollo de hardware y software (Ver Definición del proceso con SPEM 2); el proceso de monitorización comienza con la tarea de *Configuración de la Monitorización*, que recibe una serie de artefactos, que representarán los insumos para la obtención de *Modelo en Tiempo de Ejecución*. Este modelo será usado por el middleware de monitorización en la actividad de *Monitorización*.

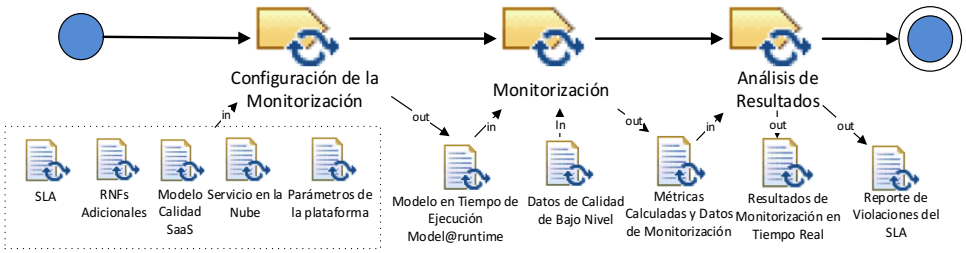


Figura 4-1 Método de Monitorización de Calidad de Cloud MoS@RT (Cedillo *et al.*, 2014)

La actividad de *Monitorización* captura datos de bajo nivel desde los servicios en ejecución utilizando diversos escenarios de captura, especificados en el modelo en tiempo de ejecución, y alimenta a la infraestructura de monitorización con información útil y filtrada, la cual es usada por la actividad de *Análisis de Resultados* para poder determinar las violaciones del SLA y la visualización del estado de los servicios.

La actividad de *Análisis de Resultados* usa los datos generados por la actividad de *Monitorización*, los compara con los valores esperados. Las siguientes sub-secciones, describen sistemáticamente cada tarea y las actividades del método de monitorización.

A nivel general los artefactos de entrada y salida del proceso serán en primer lugar la lista de requisitos no funcionales (RNF) a ser monitorizados, estos requisitos están conformados por aquellos especificados en el *Acuerdo de Nivel de Servicio (SLA)* y en los *RNF adicionales*, los mismos que son de interés para cualquiera de las partes, y que pueden ser definidos con la finalidad de tomar acciones correctivas, preventivas o a manera de generación de reportes sobre estado de calidad de los servicios.

Todos los RNF medibles de interés sirven de artefacto de entrada, así como el artefacto a ser monitorizado, en este caso el *Servicio Cloud* desplegado que se requiere evaluar.

Por otro lado, está el *Modelo de Calidad SaaS*, el mismo que será descrito en las próximas secciones y que a manera de introducción se puede observar como una herramienta guía que permitirá la categorización organizada de los RNF a ser

monitorizados. Este modelo de calidad se ha especificado en base al estándar ISO 25010 (2011). Y contiene la descomposición de las características, sub-características, atributos, métricas y operacionalizaciones de calidad que pueden ser utilizadas durante la monitorización, este modelo es extensible y constituye una base de conocimiento de cuestiones de calidad, que pueden ser empleadas en cualquier método de monitorización de calidad de SaaS.

Como salida de la configuración de la monitorización se obtiene el *Modelo en Tiempo de Ejecución*, este modelo tendrá todas las configuraciones necesarias para realizar la monitorización del servicio en cuestión, los RNF a ser monitorizados, las opciones de acceso al servicio, los atributos de calidad y las métricas a ser empleadas para las mediciones, el método de recolección de datos y los umbrales a ser alcanzados y que están contenidos en los SLA y RNF adicionales.

Una vez recibido el modelo en tiempo de ejecución, la actividad de monitorización se encargará de realizar la captura de la información, realizar las mediciones haciendo uso de las métricas especificadas en el modelo en tiempo de ejecución, obtener los valores desde el servicio cloud y realizar los cálculos necesarios para obtener los valores de las métricas, que serán los datos de salida de la actividad de monitorización y de entrada a la siguiente actividad del método.

Finalmente, se realiza el análisis de la información de monitorización, la misma que está dirigida a la verificación del SLA o a la visualización de los valores obtenidos tras la monitorización, versus los valores umbrales esperados y definidos en los requisitos de monitorización que fueron pasados al modelo en tiempo de ejecución.

Las actividades relacionadas al proceso completo de monitorización, a su vez pueden estar divididas en tareas que permitan cumplir con estas actividades. De esta manera, el proceso se ve dividido en actividades y éstas a su vez en tareas.

4.2. Descripción de las actividades de monitorización

En las siguientes sub-secciones se presenta una descripción detallada de cada una de las actividades que conforman el método de monitorización, así como también las tareas involucradas.

4.2.1 Configuración de la monitorización

La configuración de la monitorización es la actividad responsable de la preparación del modelo en tiempo de ejecución, que es el que contiene las directivas de monitorización, utilizadas durante el método. A breves rasgos, ésta es una actividad en la que intervienen los usuarios directamente y en la que se

pueden identificar dos actores principales, estos son: (1) el *Planificador de la Monitorización*, el mismo que se encargará de establecer los requisitos de monitorización que se tendrán en cuenta a lo largo del método y de definir a través de los SLA y RNF adicionales, el Modelo de Requisitos de Monitorización. Luego, se distingue el actor (2) *Configurador de la Monitorización*, quien se encargará de realizar la configuración de la monitorización, hasta la obtención del Modelo de Calidad en Tiempo de Ejecución.

Las tareas de esta actividad son descritas en detalle a continuación, y están ilustradas en la Figura 4-2.

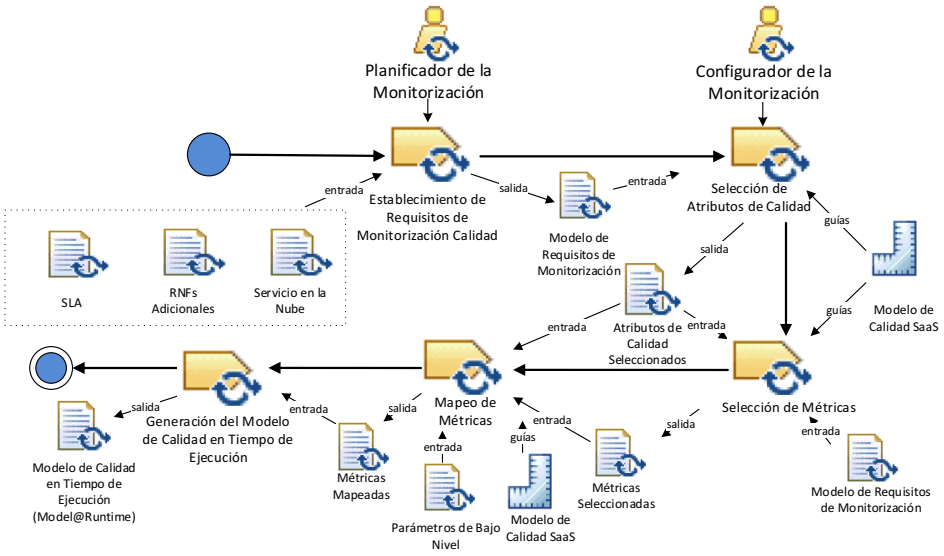


Figura 4-2. Diagrama SPEM de la Actividad de Configuración de la Monitorización (Cedillo *et al.*, 2014)

4.2.1.1 Establecimiento de los requisitos de monitorización

Esta tarea recibe tres artefactos como entrada: (1) el SLA, (2) los RNF adicionales que se necesiten monitorizar y (3) los artefactos los cuales serán analizados por el proceso de monitorización (p. ej. servicios, aplicaciones). La salida de esta tarea es la especificación de los requisitos a través del *Modelo de Requisitos de Monitorización*. Este contiene los RNF a ser monitorizados, además este modelo especifica las métricas que serán usadas durante el método y que pueden ser el resultado de la negociación del cliente con el proveedor de servicios a través del SLA o bien la métrica que la persona interesada en la monitorización quiere utilizar para usar durante el proceso. Así también el modelo de requisitos de monitorización contiene todos los umbrales que tienen que cumplirse durante la provisión de los servicios y que han sido especificados

como parte del SLA o en los valores esperados a ser cumplidos por los requisitos adicionales de monitorización.

Como ya se ha comentado, el planificador de la monitorización, es el encargado de generar el modelo de requisitos de monitorización, de acuerdo a las necesidades de monitorización especificadas en los SLA y en los RNF adicionales expresados por las partes involucradas en la provisión de los servicios (cliente o proveedor de servicios).

La ventaja de contar con un modelo de requisitos de monitorización, es que al llevar las necesidades a un formato verificable automáticamente, se elimina la ambigüedad presentada en un SLA, en donde los RNF suelen venir expresados en diferentes formatos e incluso en lenguaje natural. Para ello se han definido algunos lenguajes de especificación de SLA.

Los requisitos de monitorización se expresarán utilizando el estándar WSLA para la especificación de los RNF. Para ello el *Planificador de la Monitorización*, deberá traducir los RNF contenidos en el SLA y así como también RNF adicionales al lenguaje WSLA. Para ello se ha establecido un modelo de requisitos de calidad basado en este estándar, el mismo que será presentado en detalle en el próximo capítulo, ya que este modelo forma parte del conjunto de artefactos de la infraestructura de monitorización definida para llevar a cabo este método.

A continuación se presenta un ejemplo de los RNF especificados en un SLA y expresados en WSLA a fin de tener una idea global de lo que implica la preparación del modelo de requisitos de monitorización:

Requisito No-Funcional (Lenguaje Natural):

La eficiencia de la operación X del servicio Y será medida a través del tiempo de respuesta, se ha establecido que será de máximo 500 milisegundos y será calculada en base al tiempo de ejecución de la petición (ver Figura 4-3).

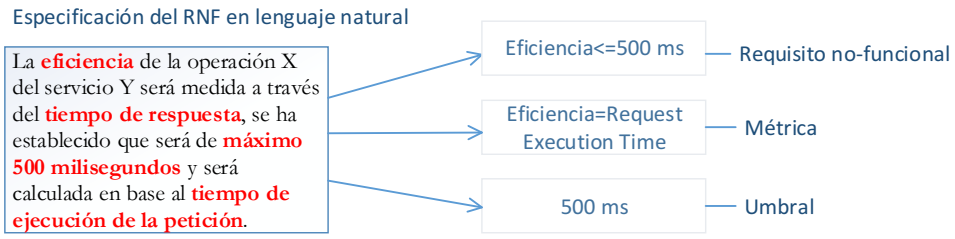


Figura 4-3 Descomposición de RNF en lenguaje natural

Requisito No-Funcional expresado usando WSLA

```
<Operation name="X">
        <SLAParameter name="Efficiency" type="float" unit="milliseconds">
                <Metric>EfficiencyMetric</Metric>
                <Communication>
                        <Source>CoCoMeORCProvider</Source>
                        <Pull>XYZAuditing</Pull>
                        <Push>XYZAuditing</Push>
                </Communication>
        </SLAParameter>
</Operation>
<Metric name="EfficiencyMetric" type="float" unit="milliseconds">
    <Source>AuditorandServiceMeasurements</Source>
    <Function type="RequestExecutionTime" resultType="float">
        <Schedule>hourlySchedule</Schedule>
        <Operand>
                <Metric>RequestExecutionTime</Metric>
        </Operand>
  </Function>
</Metric>
```

4.2.1.2 Selección de los atributos de calidad

Dentro de la configuración de la monitorización, se hace necesario enmarcar cada uno de los RNF contenidos en el modelo de requisitos de monitorización, dentro de un atributo de calidad correspondiente al modelo de calidad SaaS, recibido como entrada a la actividad de configuración de la monitorización. La finalidad de esta tarea, es la de mapear de una manera rápida, fácil, estandarizada y reproducible los RNF a los atributos planteados en el modelo de calidad; para que, en las siguientes tareas, sea posible convertir estos RNF y sus métricas de alto nivel, en atributos medibles que utilicen los datos de bajo nivel obtenidos de los servicios cloud, para poder realizar las mediciones.

La descomposición completa y explicación del modelo de calidad SaaS, serán estudiadas con detenimiento en el capítulo 5, dado que éste al igual que el modelo de requisitos de calidad forma parte de la infraestructura de monitorización. En la Figura 4-4 se muestra el mapeo de los RNF a atributos de calidad contenidos en el modelo de calidad SaaS. En la Tabla 4-1 se incluye un ejemplo de un modelo de calidad SaaS. En donde se puede apreciar que, de la eficiencia se deriva el comportamiento en el tiempo y de ahí el tiempo de respuesta, atributo que corresponde al RNF expresado en el SLA.

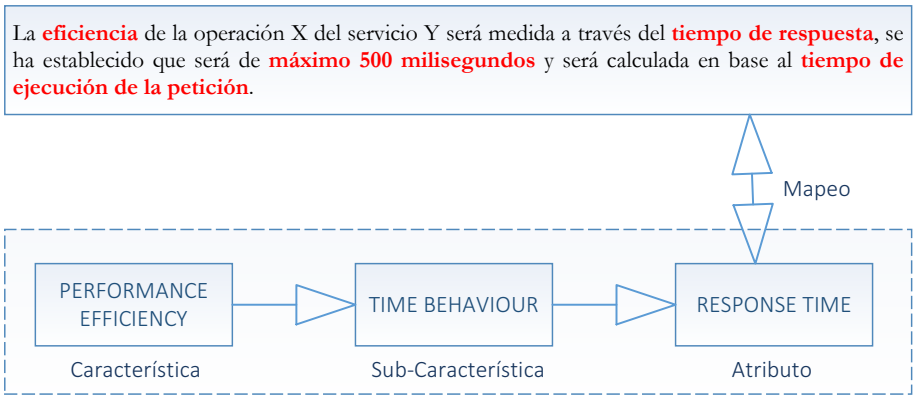


Figura 4-4 Mapeo de Requisito No-funcional a Atributo de Calidad

Tabla 4-1. Extracto de un modelo de calidad SaaS

Características	Sub-Características	Atributos	Métricas	Operacionalizaciones
Performance Efficiency	Time Behavior	Response Time	Latency	Request Execution Time + Request Response Time
			Request Execution Time	Execution Time
			Time Behavior (TB)	Execution Time / Total Service Invocation Time
		Data Exchange Workload		

La salida de esta tarea por tanto será la lista de atributos que han sido mapeados con el modelo de requisitos de monitorización.

4.2.1.3 Selección de métricas de calidad

Dentro de la configuración de la monitorización, se realiza la selección de métricas de calidad; aquí, se pueden escoger la métrica y de entre un conjunto de "operacionalizaciones" de esa métrica, contenidas en el modelo de calidad SaaS, la más adecuada para medir el atributo de calidad, la métrica debe ser equivalente a la pactada en el SLA y contenida en el modelo de requisitos de monitorización.

Si la operacionalización no está disponible en el modelo de calidad SaaS, el usuario podrá incluir una nueva operacionalización, alimentando de esta manera dicho modelo; las operacionalizaciones añadidas al modelo de calidad SaaS podrán ser utilizadas para sucesivas configuraciones de monitorización, conformando así una ***base de conocimiento*** que permita a los usuarios contar con más formas de calcular una métrica, complementando el modelo de calidad

SaaS y haciendo que éste permita una selección guiada y fácil de a la hora de configurar las métricas y operacionalizaciones que serán utilizadas durante la monitorización de los servicios.

En la Figura 4-5 se pueden apreciar tanto las métricas como las operacionalizaciones que pueden ser utilizadas para la configuración del ejemplo de la sección anterior, en este caso el atributo *Response Time* perteneciente al extracto del modelo de calidad SaaS presentado, contiene la métrica a ser utilizada *Request Execution Time* cuya operacionalización es el *Execution Time*, aquí podrían existir más operacionalizaciones o se puede incluir alguna otra si es el caso, sin embargo la métrica a ser usada según el ejemplo es el *Execution Time* la misma que será considerada para la configuración de este caso de monitorización.

4.2.1.4 Mapeo de las métricas de calidad

Una vez que se han escogido las operacionalizaciones que serán utilizadas para medir los atributos de calidad, es necesario mapear dichas operacionalizaciones con los datos de bajo nivel, que se obtienen desde los servicios monitorizados. Estos datos constituyen datos crudos, los mismos que representan valores de bajo nivel recogidos desde los servicios, o valores que mediante los envoltorios u otras formas de recolección de información son capturados.

Para ejemplificar esta tarea y continuando con el ejemplo, se verá una manera de recuperar el valor del *Request Execution Time* desde la plataforma en la que está desplegado el servicio a ser monitorizado. Para este ejemplo suponemos que la plataforma es Microsoft Azure. De ahí, tomamos la lista de contadores específicos de la plataforma (que representa una de las maneras de captura de información) como se muestra en la Figura 4-5, en este caso el servicio Diagnostics de Microsoft Azure provee un contador llamado "\ASP.NET\ RequestExecutionTime", el cual será mapeado a la métrica con la que se calculará el valor de monitorización.

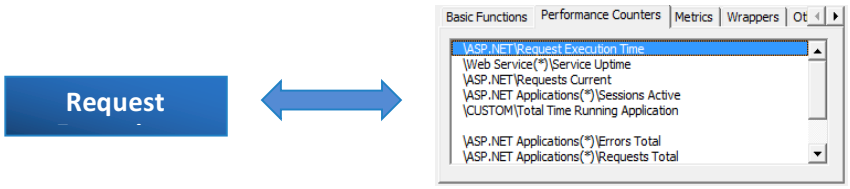


Figura 4-5 Selector de contador dependiente de la plataforma

4.2.1.5 Generación del Modelo de Calidad en Tiempo de Ejecución

Finalmente, se realiza en base a las directivas fijadas a lo largo de la actividad de configuración de la monitorización, la generación del modelo de calidad en tiempo de ejecución, dicho modelo será consumido a lo largo de las siguientes actividades del método de monitorización.

El modelo de calidad en tiempo de ejecución será presentado dentro del capítulo de la infraestructura de monitorización ya que constituye un artefacto de la misma.

4.2.2 Monitorización

La actividad de *Monitorización* recibe como un artefacto de entrada el *Modelo en Tiempo de Ejecución*, este modelo contendrá todos los parámetros necesarios para poder realizar esta tarea, los mismos que han sido incluidos durante la configuración de la monitorización. Estos parámetros obtendrán los valores de calidad actuales de los RNF que se esperan monitorizar. El modelo en tiempo de ejecución contiene una descripción de las características, sub-características, atributos y métricas a ser tenidas en cuenta durante las fases siguientes de la monitorización; de ahí, que este modelo está conformado por un subconjunto de las clases que componen el *Modelo de Calidad SaaS* y a su vez contiene los parámetros necesarios de acceso y configuración de los servicios a ser monitorizados.

Dentro de la monitorización, se tiene en cuenta la aplicación de las métricas para la medición de las características de calidad de los servicios monitorizados. Aquí se obtienen los datos de bajo nivel emanados por los servicios y recogidos a través de varios escenarios de recolección de datos que serán presentados en capítulos posteriores con mayor detenimiento. Estos escenarios de monitorización serán descritos en los capítulos siguientes ya que forman parte de la infraestructura de monitorización aquí planteada.

Por consiguiente, los datos crudos son recogidos y con ellos se realiza el cómputo de los valores actuales de calidad basados en las métricas especificadas tanto entre cliente y proveedor a través de los SLA, como también establecido por una de las partes dentro de los RNF adicionales especificados en el modelo de requisitos de la monitorización. Estas métricas están contenidas en el modelo de calidad en tiempo de ejecución, a modo de métricas dependientes de la plataforma, ya que en el modelo de calidad, luego de realizar el mapeo de las métricas, éstas ya están en un formato entendible para la plataforma en la que está desplegado el servicio a ser monitorizado.

De esta manera, los valores de bajo nivel obtenidos tras la monitorización de los servicios, pueden ser transformados en métricas de alto nivel, más significativas tanto para el cliente como para el proveedor del servicio.

4.2.3 Análisis de la monitorización

En esta actividad se compara los valores obtenidos por la monitorización con los umbrales establecidos en los RNF y contenidos en el modelo de calidad en tiempo de ejecución, analiza los resultados y reporta el análisis de los datos de monitorización. Estos resultados podrían utilizarse para planificar una estrategia de reconfiguración de la arquitectura de los servicios, de manera que el sistema se adapte para cumplir con los requisitos no funcionales cerrando de esta manera el bucle de control autónomo.

Cabe destacar que dentro del análisis de monitorización se presentarán los resultados en diferentes formatos, en primer lugar se podrá obtener una visión en tiempo real de los resultados, para ello se contará con gráficos estadísticos, de radar, etc., así también se podrá obtener un reporte de cumplimiento del SLA, el cual puede servir tanto para el cliente como para el proveedor de servicios como un referente claro de cómo los servicios están siendo provistos.

Las formas de visualización serán contempladas en trabajos futuros, sin embargo se ha realizado un prototipo que permite ver un reporte de cumplimiento del SLA como primera aproximación de esta actividad.

Por otro lado existen muchas librerías y herramientas que permiten presentar resultados de varias maneras que pueden ser adaptadas a esta solución, preocupándonos en este caso únicamente en la recolección exacta de datos y el tratamiento de los mismos para verificar el cumplimiento de los SLA y los RNF adicionales.

4.3. Conclusiones

En este capítulo se ha presentado el método de monitorización, para ello se ha propuesto el uso de modelos en tiempo de ejecución. Este método permite la especificación de los RNF descritos por un SLA, así como también RNF adicionales de interés para proveedores de servicio. Además, se han descrito cada una de las actividades principales del método de monitorización, como se puede ver cada una de ellas tiene sus entradas y sus salidas. Este método es útil ya que permite medir los atributos de alto nivel especificados en los SLA y provee flexibilidad cuando el evaluador necesita cambiar o añadir más RNF a la monitorización; esto debido a la contribución que prestan los modelos en tiempo de ejecución, ya que cualquier modificación se la realiza sobre el modelo sin

necesidad de realizar implementaciones adicionales sobre la infraestructura de monitorización. En primer lugar, se ha definido la actividad de configuración de la monitorización; en la cual, se han establecido los roles de usuario de la misma. Luego se tienen actividades automáticas que funcionan midiendo los atributos de calidad y finalmente mostrando los resultados.

Se contempla como trabajo futuro, el establecimiento automático de ciertos parámetros de configuración como es el caso del tiempo de recogida de datos crudos, con la finalidad de optimizar el rendimiento y fijando tiempos idóneos en los que haya un balance entre la precisión de los datos de monitorización y la sobrecarga que implica la captura de esa información.

Capítulo 5
Infraestructura de Monitorización

En este capítulo, se presenta la Infraestructura de Monitorización, que ha sido diseñada para soportar el método de monitorización definido en el capítulo anterior. La infraestructura aquí propuesta es independiente de la plataforma de monitorización, constituyendo un diseño que puede ser implementado para funcionar en cualquier plataforma.

La sección 5.1 se presenta la arquitectura general de la infraestructura y sus componentes principales. La sección 5.2 muestra el configurador de la monitorización y los metamodelos que serán necesarios para la infraestructura. La sección 5.3, muestra el middleware de monitorización y el funcionamiento del motor de medición y de análisis. A continuación, en la sección 5.4 se muestran los métodos de recolección de la información desde los servicios. La sección 5.5 presenta la manera en la que se realizará la posterior instanciación de la infraestructura en plataformas específicas y finalmente, las conclusiones del capítulo se presentan en la sección 5.6.

5.1. Arquitectura general de la infraestructura

La arquitectura general de la infraestructura se encuentra La Figura 5-1 muestra la infraestructura de monitorización con sus componentes principales la misma que ha sido presentada en (Cedillo et al., 2014).

La infraestructura de monitorización se divide en dos componentes principales, los cuales permiten:

1. La especificación y configuración de RNFs a ser monitorizados.
2. Una interacción con los servicios cloud para evaluar su calidad en tiempo de ejecución.
3. La generación de reportes con eventuales violaciones al SLA o informes presentados a través de gráficos estadísticos de diversa índole presentados en tiempo real.

Esta infraestructura permite lograr estos objetivos y proveer el grado de flexibilidad requerido cuando se definen o modifican los NFRs y sus métricas, ya sea porque se requiere la monitorización de una nuevo requisito de calidad o

porque se ha dado una renegociación del SLA que ha generado la necesidad de cambiar los parámetros de monitorización; así también soporta diferentes escenarios para recoger información de los servicios cloud. Para ello, se han definido un conjunto de componentes y artefactos que conforman la infraestructura de monitorización y utilizan modelos en tiempo de ejecución.

La Infraestructura de Monitorización tiene dos componentes principales: El *Configurador de la Monitorización* y el *Middleware de Monitorización y Análisis*. El configurador de la monitorización usa el *Modelo de Requisitos de Monitorización* y el *Modelo de Calidad SaaS* para configurar la monitorización de los servicios y obtener el *Modelo en Tiempo de Ejecución*.

El middleware de monitorización y análisis usa el modelo en tiempo de ejecución y contiene dos motores: el *Motor de Medición*, el cual permite la monitorización de servicios cloud a través del uso de datos de calidad tomados directamente desde los servicios y la respectiva realización de mediciones, y el *Motor de Análisis*, el cual compara los valores esperados con los valores monitorizados y pueden generar el reporte de violaciones del SLA. Los detalles de cada proceso y artefactos que intervienen serán detallados en las próximas secciones de este capítulo.

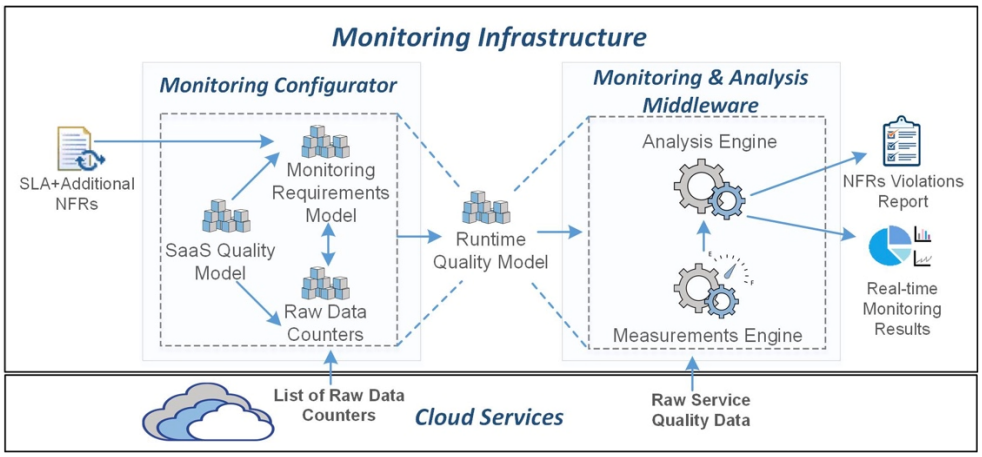


Figura 5-1 Infraestructura de Monitorización (Cedillo, *et al.*, 2015)

En la Figura 5-2, se detalla la comunicación entre de cada uno de los componentes de la infraestructura de monitorización, en donde como se puede observar en la Figura 5-2 (1) que el Modelo de Requisitos de Monitorización, ingresa como arte9facto de entrada y éste interactúa directamente con el Configurador de la Monitorización Figura 5-2 (4) en donde se realiza el mapeo entre lo que se desea medir y el cómo se realizará esta medición. Para realizar

este mapeo se utiliza el Modelo de Calidad SaaS, Figura 5-2 (2), el mismo que será el soporte que permita la clasificación y posterior selección de métricas adecuadas para la monitorización. El Configurador de la Monitorización, Figura 5-2 (4), permite generar el Modelo de Calidad en Tiempo de Ejecución Figura 5-2 (3). El mismo que servirá de insumo para el Middleware de Monitorización y Análisis, Figura 5-2 (5), el cual inte/ractúa con los servicios a ser monitorizados Figura 5-2 (6) para poder recuperar los datos de acuerdo a los parámetros establecidos en el Modelo de Calidad en Tiempo de Ejecución.

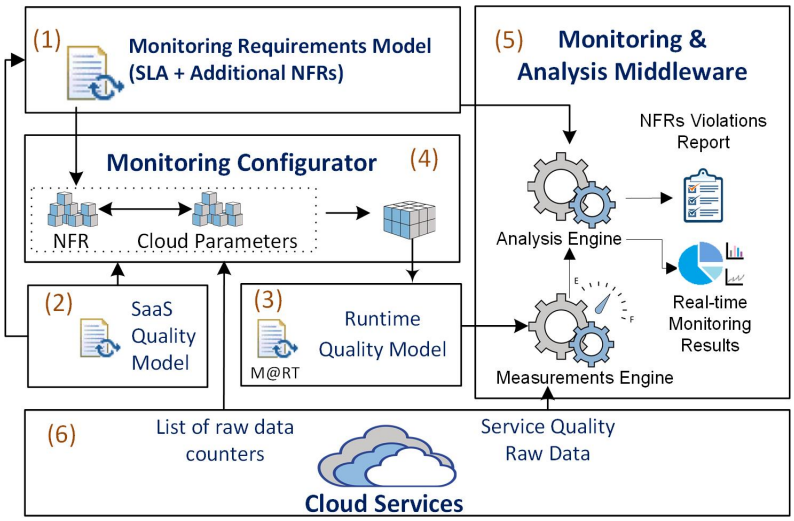


Figura 5-2 Arquitectura de los Componentes de la Infraestructura y su Comunicación

5.2. Configurador de la monitorización

El *Configurador de la Monitorización* es un componente de la Infraestructura de Monitorización, el mismo que permite mediante una interface de usuario configurar las directivas de monitorización. Este componente facilita la configuración tanto de los RNF de alto como de bajo nivel que son incluidos en el Modelo de Requisitos de Monitorización para que puedan ser mapeados con los datos crudos tomados de los servicios. El Modelo de Calidad SaaS juega un papel crucial al momento de realizar el mapeo, ya que actúa como una guía, que permite la clasificación de los NFR con el objetivo de realizar una selección de operaciones apropiadas para las métricas. Cuando el mapeo es realizado por los *stakeholders*, el Modelo de Calidad en Tiempo de Ejecución es generado y puede ser consumido por el Middleware de Monitorización y Análisis. Una descripción detallada de los artefactos envueltos en el Confi-gurador de la Monitorización y sus interacciones se muestra a continuación.

5.2.1. Meta-modelo de requisitos de monitorización

El Modelo de Requisitos de Monitorización especifica los RNF a ser monitorizados, de manera que puedan ser consumidos por el configurador de la monitorización. Este modelo se basa en el lenguaje de especificación WSLA propuesto por Ludwig *et al. (*2003) para representar NFRs de una manera estandarizada. El hecho de que el modelo se base en un estándar constituye una gran ventaja dado que los RNF que se especifican por ejemplo en los SLA, muchas veces están expresadas de manera informal y hasta en lenguaje natural, lo que hace de esta solución, una manera ordenada de especificación de requisitos que permite una vez que los RNF han sido correctamente configurados, que éstos puedan ser automatizados con diferentes fines. Además, en esta solución, el modelo es extendido para soportar los RNF adicionales que no son parte del SLA pero que pueden ser de interés para los *stakeholders*.

La Figura 5-3 muestra el meta-modelo de requisitos de monitorización, el cual incorpora todas las secciones que se incluyen de manera común en los SLA. El SLA especifica las partes que intervienen en el contrato, las cuales se dividen en partes signatarias y partes de soporte. Las partes signatarias están conformadas por una parte, por el proveedor de servicios y por otra, el cliente del servicio, quienes "firman" el SLA. Además, las partes de soporte son patrocinadas por las partes signatarias, para proveer mediciones de los servicios y auditorías. El meta-modelo incluye la meta-clase *SLAParameter*, la cual representa los NFR a ser monitorizados y la meta-clase *Metrics* es usada para ejecutar las mediciones. La meta-clase *Service Object* es la abstracción de un servicio, cuyas características de calidad y atributos son relevantes con respecto a los términos definidos en el SLA. Las características y atributos son especificadas como *SLAParameters* en donde cada *SLAParameter* puede ser medido utilizando métricas. La meta-clase *SLAParameter* tiene un atributo llamado *isSLATerm,* el cual diferencia un término SLA de un RNF no incluido en el SLA. Esta diferenciación es útil cuando lo que se requiere es obtener un informe que presente exclusivamente el cumplimiento del SLA. La meta-clase *Obligation* contiene dos tipos de obligaciones: i) un *Service Level Objective*, el cual garantiza el objetivo a perseguir con respecto al estado de un parámetro SLA en un período dado de tiempo (p. ej. el tiempo de respuesta promedio deberá ser 5ms) y ii) La meta-clase *Action Guarantee*, la cual especifica las acciones a tomarse por parte del proveedor ante una situación específica (p. ej. si existe una violación de una garantía especificada, una notificación es enviada indicando una penalización). Los valores usados como umbrales son obtenidos de esta meta-clase (p. ej. el tiempo de respuesta debe ser menor que 0.7 a menos que la tasa de la transacción sea mayor que 1000). En este meta-modelo, se debe utilizar

como métrica aquella que ha sido acordada entre las partes, o en su defecto una métrica claramente compatible o equivalente a la que se manifiesta en el SLA. Una especificación más detallada del WSLA junto con ejemplos ha sido presentada por Ludwig *et al. (*2003).

5.2.2. Meta-modelo de calidad SaaS

El modelo de Calidad SaaS está alineado con el estándar ISO/IEC SQuaRE (2011). La Figura 5-4 muestra el meta-modelo utilizado para definir el Modelo de Calidad SaaS. Este modelo permite la definición de todo el conjunto de características, sub-características, atributos, su impacto (relación entre atributos), las métricas y sus respectivas operacionalizaciones; las mismas que se pueden utilizar durante la monitorización de los RNF propuestos para evaluar la calidad de los servicios cloud. La operacionalización de una métrica puede ser considerada a diferentes niveles cloud (SaaS, PaaS, IaaS). Esto es útil debido a que existe un número de requisitos de calidad (p. ej. escalabilidad, elasticidad, seguridad) que necesita ser monitorizadas a diferentes niveles de provisión de servicios (Aceto *et al.*, 2013). Además, es importante especificar la perspectiva del *stakeholder* que utilizará la información de monitorización; por ejemplo, si el *stakeholder* es un proveedor de servicios, puede ser interesante para él, conocer el número promedio de usuarios que solicitan un servicio en un momento dado. De manera similar, si el *stackeholder* representa un usuario que contrata el servicio cloud, sin duda lo que le interesará es evaluar las características propias de la provisión de su servicio. El propósito de tener perspectivas asociadas con cada operacionalización es para expresar todas las posibles funciones de medición relacionadas a una métrica, dependiendo del punto de vista desde el que se quiera medir el servicio. Sin embargo se deberá tener presente, que estas perspectivas serán también enfocadas en la construcción misma del SLA y de los RNF a ser monitorizados, adjuntándose desde el mismo Modelo de Requisitos de Monitorización la métrica que será utilizada durante el proceso.

La información contenida en el Modelo de Calidad SaaS es de mucha utilidad durante los procesos de contrastar, mejorar las mediciones o escoger distintas funciones de medición con las cuales medir cada RNF. La meta-clase *DirectMetric-Operationalization* representa una medida de un atributo que no depende de ninguna otra medida, mientras que la meta-clase *IndirectMetricOperationalization* representa medidas que son derivadas de otras *DirectMetricOperationalizations* o *IndirectMetricOperationalizations*.

Las meta-clases *Platform* y *MeasurementMethod* han sido añadidas al Modelo de Calidad SaaS para mantener una lista de contadores e instrucciones propias de la plataforma, las cuales son dependientes de la misma ya que ellas permiten

mantener una lista de información de recuperación para una plataforma específica, esta solución tiene como ventaja que con el uso de esta solución, y a medida de que los usuarios agreguen funciones de medición dependientes de la plataforma, se consigue una base de conocimiento que para futuras configuraciones puede ser de mucha utilidad, como trabajo futuro, se podría crear un repositorio de instrucciones dependientes de la plataforma que faciliten la configuración de la monitorización de manera significativa a los diferentes usuarios de esta metodología e infraestructura. Finalmente, el meta-modelo incluye particularidades de cada operacionalización, tales como la meta-clase *Unit*, la cual expresa la magnitud relacionada a una cantidad particular. Finalmente, la meta-clase *Scale*, representa un conjunto de propiedades de valores continuos o discretos, que son usados para especificar la operacionalización que será utilizada y así poder realizar correctamente el mapeo de la misma, de acuerdo a la escala utilizada.

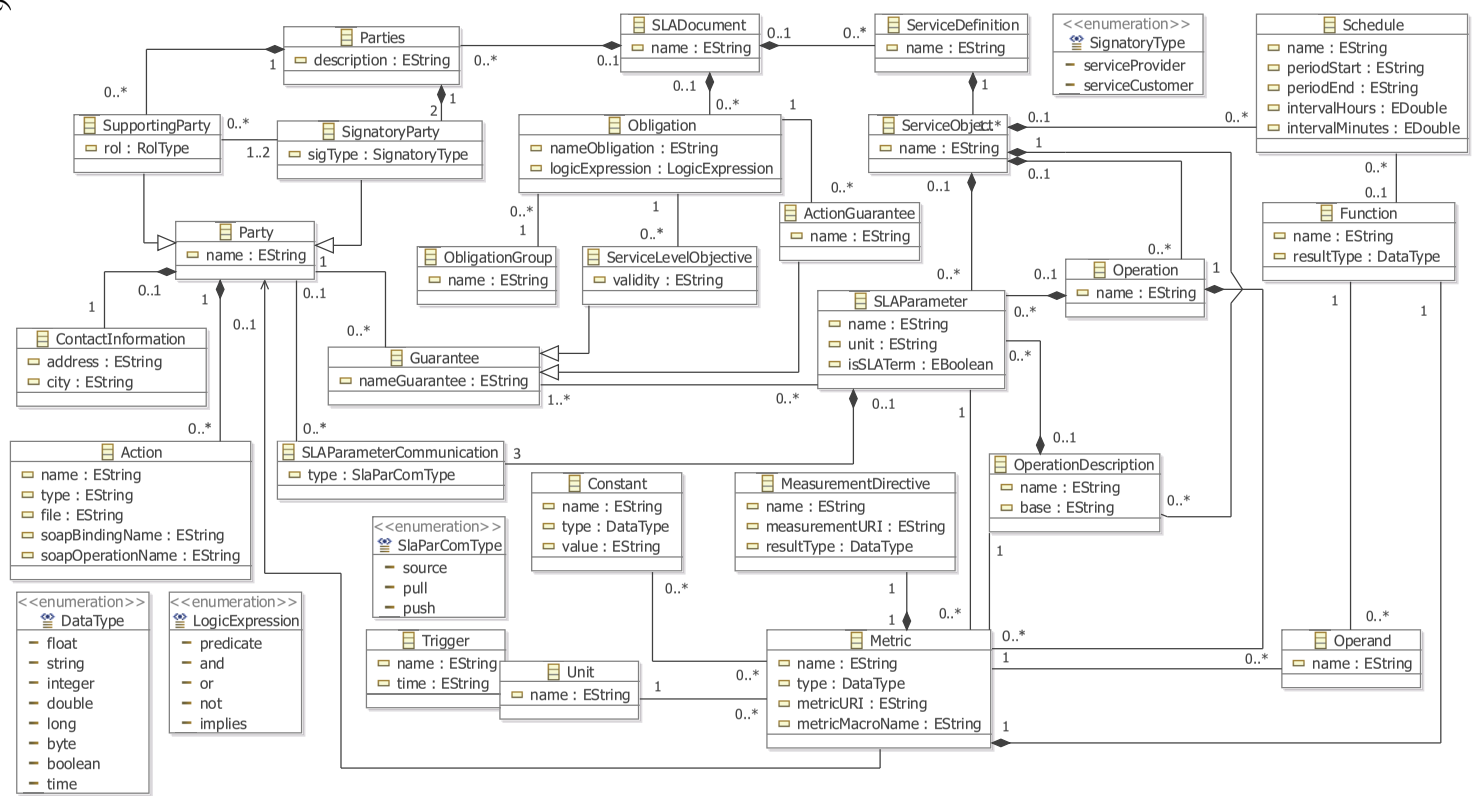


Figura 5-3 Modelo de Requisitos de Monitorización (Cedillo, Gonzalez-Huerta, *et al.*, 2015)

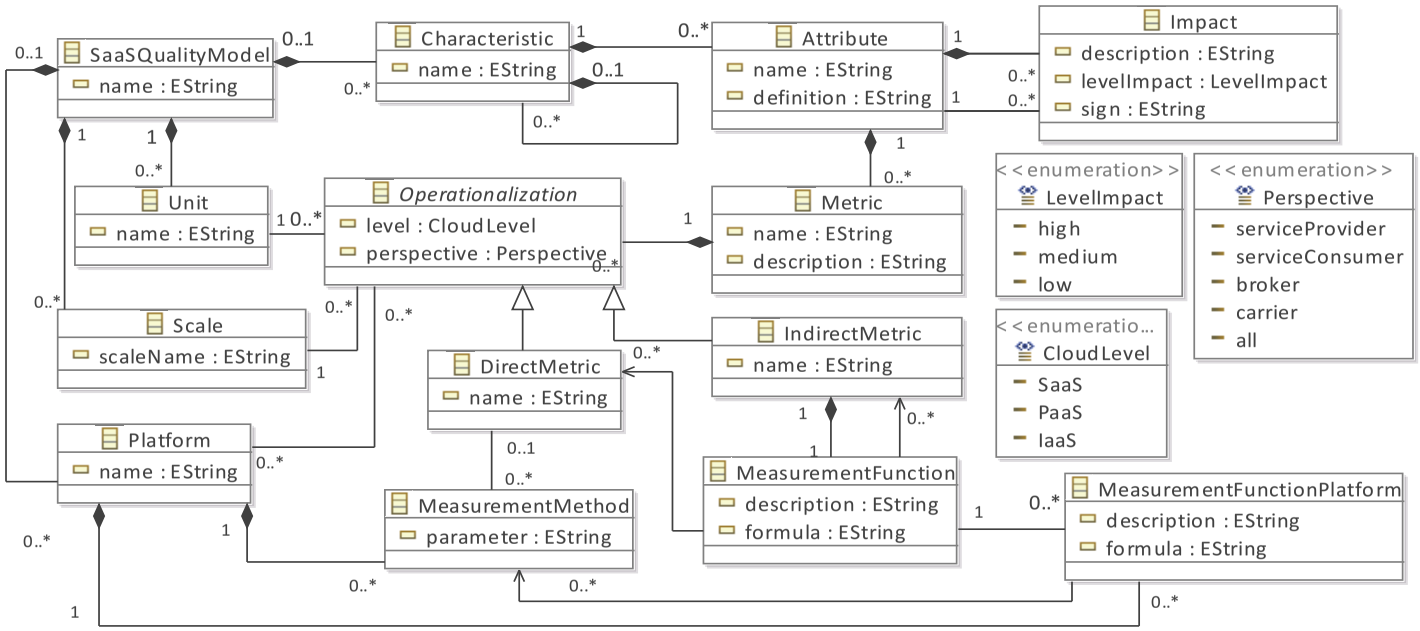


Figura 5-4 Modelo de Calidad SaaS (Cedillo, Gonzalez-Huerta, *et al.*, 2015)

5.2.3. Meta-modelo de calidad en tiempo de ejecución

El Modelo de Calidad en Tiempo de Ejecución especifica los requisitos de monitorización, métricas, operacionalizaciones y configuraciones que serán utilizadas durante la monitorización, este modelo es el resultado de la configuración de la monitorización y contiene las directivas a ser utilizadas por el motor de medición que forma parte del middleware de monitorización y análisis mostrado en la Figura 5-4.

Lehmann *et al. (*2010) argumenta que los modelos en tiempo de ejecución proveen constructores de modelado que habilitan la definición de:

a) Una parte prescriptiva del modelo, que especifica cómo el sistema **debería ser**.
b) Una parte descriptiva del modelo que especifica cómo el sistema **es**,
c) Las **modificaciones válidas** sobre las partes **descriptivas** del modelo, ejecutable en tiempo de ejecución,
d) Las **modificaciones válidas** sobre las partes **prescriptivas**, ejecutables en tiempo de ejecución.
e) La **conexión causal**, la cual está en la forma de un flujo de información entre el modelo y la entidad a ser monitorizada.

La Figura 5-5 muestra el Modelo de Calidad en Tiempo de Ejecución, el cual contiene muchas de las meta-clases contenidas en el Modelo de Calidad SaaS, por lo que puede ser considerado como una extensión del mismo, a más de meta-clases que representan la parte prescriptiva, descriptiva y las características de la plataforma cloud que permiten la conexión causal.

Para motivos de explicación, se ha realizado una agrupación de meta-clases, según su parte prescriptiva, descriptiva y conexión causal, a través de líneas entrecortadas de diferentes colores, en el primer caso, tenemos la parte prescriptiva, la cual contiene las meta-clases *Additional NFR, SLATerm* y *Threashold,* estas tres meta-clases muestran claramente lo que se quiere monitorizar conjuntamente con el umbral que se ha ofrecido alcanzar, esto representa una "prescripción" de la calidad mínima para cumplir con el SLA y las expectativas de provisión de calidad de servicio. Por otro lado, se tiene la parte descriptiva, la misma que está representada por las meta-clases *RawDataInstance* y *CalculatedMetric*, estas meta-clases contienen los datos que se obtienen de los servicios tras la monitorización, por tanto representan la parte "descriptiva" del estado de calidad de los servicios monitorizados en una instancia de tiempo dado. Finalmente, para la conexión causal se tiene la meta-clase *ConfigurationFile* esta meta clase permite establecer los parámetros de

conexión hacia los servicios, contribuyendo a la comunicación entre el middleware de monitorización que utiliza el modelo en tiempo de ejecución y el servicio monitorizado.

A continuación se describirá cada una de las clases que intervienen en el Modelo de Requisitos de Monitorización: La meta-clase *QualityRuntimeModel*, que contiene el nombre del modelo de calidad en tiempo de ejecución y una breve descripción del mismo. Esta meta-clase será la contenedora de todo el modelo. La meta-clase *CloudService* describe el servicio a ser monitorizado, ésta contiene el nombre del servicio y la referencia al SLA que debe cumplirse. La meta-clase *Threshold* representan el umbral que debe cumplir cualquier término del Modelo de Requisitos de Monitorización, este umbral puede ser un umbral de *SLATerm*, obtenido de la parte de las obligaciones del SLA, o un *AdditionalNFR* configurado por el *stakeholder*. La meta-clase *RawDataInstance* contiene los valores crudos capturados directamente desde el servicio *cloud* durante su provisión; por otro lado, la meta-clase *CalculatedMetric*, contiene los resultados de las mediciones de las métricas calculadas. La meta-clase *ConfigurationFile* contiene información específica para cada plataforma que permite una interacción entre la infraestructura de monitorización y el servicio cloud. Esta meta-clase contendrá la dirección del servicio, y los parámetros de conexión del mismo. Finalmente, la meta-clase *Indicator* representa una medida que es derivada de otras medidas usando un *Modelo de Análisis* como un enfoque de medición (García *et al.*, 2006). Las clases aquí descritas, son las clases que se han añadido al Modelo de Calidad SaaS para formar el Modelo de Calidad en Tiempo de Ejecución. En conclusión, el *Modelo de Calidad en Tiempo de Ejecución* permite a nuestra propuesta obtener las características deseables de flexibilidad y mantenibilidad, debido a que cambios en el Modelo de Calidad en Tiempo de Ejecución pueden ser fácilmente reflejados en el Middleware de Monitorización y Análisis que consume este modelo al momento de la monitorización de los servicios.

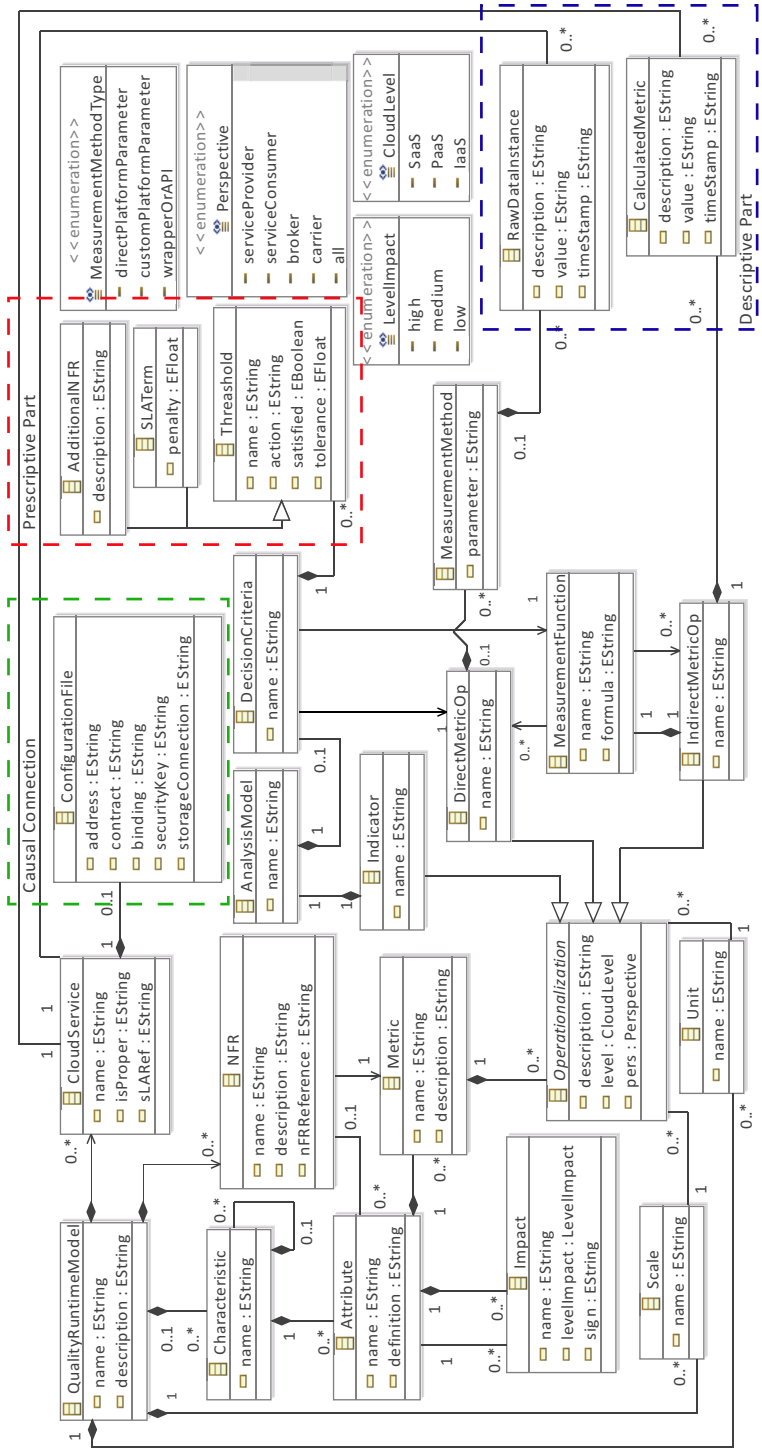


Figura 5-5 Modelo de Calidad en Tiempo de Ejecución (Cedillo, Gonzalez-Huerta, *et al.*, 2015)

5.3. Middleware de monitorización y análisis

El Middleware de Monitorización y Análisis contiene el *Motor de Medición*, el cual utiliza el modelo en tiempo de ejecución que fue obtenido utilizando el configurador de la monitorización, presentado en la subsección anterior. Este motor aplica las métricas contenidas en el modelo en tiempo de ejecución, con las cuales procede a medir la calidad de los servicios. Además se ha creado un Motor de Análisis el cual permite el análisis de la calidad y los reportes asociados. En la Figura 5-6, se presenta una visión general de Middleware de Monitorización y Análisis, en la que se ve la interacción de todos y cada uno de los componentes del mismo con los servicios y los métodos de recolección, de una manera general. Luego se presentará cada uno de los escenarios y sus interacciones con los servicios. En la Figura 5-6, por cuestiones de facilidad de ilustración y por considerarse métodos de interacción similares, los escenarios 3 y 4, se presentan como 3a y 3b; sin embargo, al momento de la representación en detalle, estos escenarios se desglosan y se explican por separado. En dicha figura se aprecia además la parte del almacenamiento de la información de monitorización, con el fin de almacenar históricos sobre la calidad de los servicios, con la finalidad generar informes bajo demanda, así como también para realizar funciones de agregación con los datos, las mismas que pueden ser necesarias cuando se toman períodos definidos de tiempo, sumatorias, etc.

Se ve adicionalmente un "Extractor" que lo que hará es interactuar con la base de datos de las métricas calculadas para realizar las mediciones y así también una primera aproximación que muestra las posibilidades de recolección de datos desde una vista de más alto nivel, para poder entender el funcionamiento general del middleware.

En las siguientes sub-secciones, se procede a mostrar el funcionamiento general de estos motores, sus objetivos específicos y la manera en la que se puede instanciar dichos componentes a una determinada plataforma.

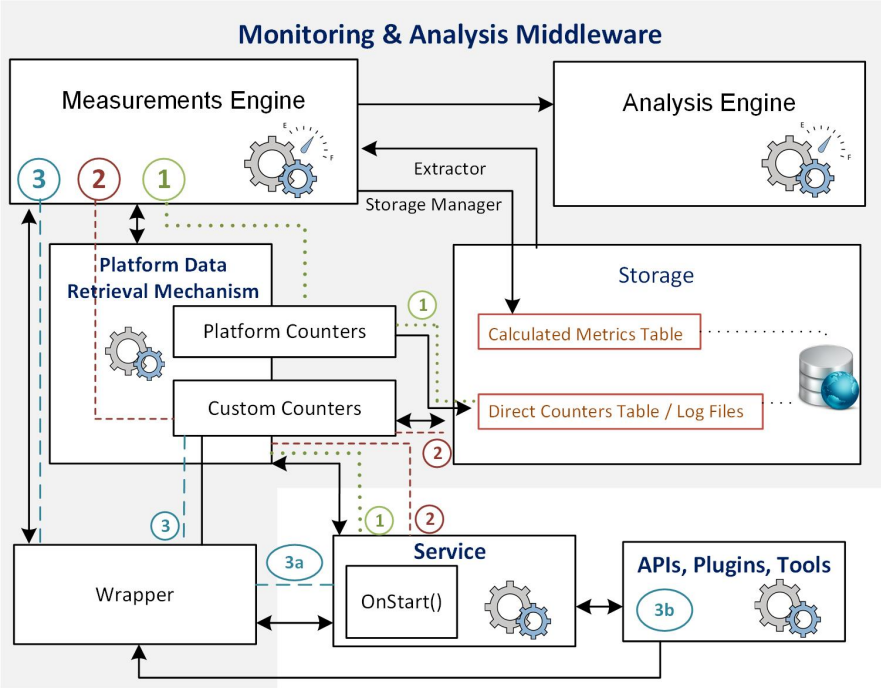


Figura 5-6 Vista detallada de la infraestructura de monitorización

5.3.1. Motor de medición

El motor de medición es una de las partes esenciales de la infraestructura de monitorización, dado que en éste se producen las actividades nucleares del proceso de extracción y cálculo de datos crudos a fin de que éstos se transformen en datos útiles para establecimiento del estado de calidad de los servicios.

El motor de medición es el encargado por tanto, de realizar los cálculos sobre los datos extraídos por los diferentes mecanismos de recolección de datos de la plataforma. Éste calcula las métricas basándose en las operacionalizaciones aportadas por el Runtime Quality Model y las almacena para su posterior análisis.

Puntualmente, las actividades que debe realizar el motor de medición, son:

1. Recibir el modelo en tiempo de ejecución y configurar el servicio para su monitorización.
2. Extraer los datos de monitorización utilizando los diferentes escenarios de extracción descritos posteriormente.
3. Realizar los cálculos descritos por las operacionalizaciones de las métricas en el modelo en tiempo de ejecución.

4. Almacenar los resultados de las métricas en un sistema de almacenamiento permanente para su posterior consulta.

5.3.2. Motor de análisis

El motor de análisis se encarga de tomar los datos calculados por el motor de medición y tratarlos (tomar grupos de datos relevantes, por ejemplo seleccionar aquellas medidas sobre disponibilidad en rangos de 5 minutos si así ha sido expresado en el SLA) para comparar los resultados obtenidos con los establecidos por los acuerdos de nivel de servicio, de manera que se pueda comprobar, si las expectativas de calidad del Modelo de Requisitos de Monitorización están dentro de los umbrales deseados y así reportar éstos resultados para que puedan ser comprensibles y útiles por parte de la persona que demanda la monitorización o por un software de reconfiguración dinámica autónomo que permita la mejora del sistema.

5.4. Métodos de recolección de datos

Como se ha visto en secciones anteriores, una parte vital del proceso de monitorización es la utilización de datos crudos de bajo nivel procedentes del servicio a ser monitorizado, con el fin de realizar los cálculos que permitan calcular las métricas. Para la obtención de dichos valores, se han establecido cuatro escenarios de captura de datos con el fin de recoger información de calidad desde los servicios explotando diferentes mecanismos, para de ésta forma darle a la infraestructura un alto nivel de flexibilidad, interoperabilidad, extensibilidad y mantenibilidad.

Además, en el capítulo 3, se han estudiado diferentes herramientas comerciales especializadas en la extracción de datos útiles para la monitorización de los servicios en la nube. En ésta propuesta, nosotros tratamos de explotar entre otras formas datos provistos por herramientas de terceros que tratan de aprovechar recursos disponibles para conseguir datos de mayor nivel en forma de métricas.

Por otra parte y como se ha descrito anteriormente, el middleware de monitorización y análisis es el encargado de recolectar la información desde los servicios desplegados en la nube. Para ello, hemos establecido diferentes escenarios de captura de datos que permiten capturar información de bajo nivel de los servicios explotando distintos mecanismos de captura de información de calidad, lo que hace que la infraestructura propuesta sea flexible y extensible. Los mecanismos de captura de información se muestran en la Figura 5-7.

Además en las siguientes subsecciones, daremos una explicación de cada una de las formas de recuperación de información, y en capítulos posteriores, trataremos de mostrar cada una de estas formas, llevadas a plataformas específicas a fin de que se pueda comprobar su factibilidad y determinar que ésta es una propuesta que cumple con las características de flexibilidad e interoperabilidad aquí ofrecidas.

5.4.1. Obtención de datos utilizando herramientas de la plataforma

El primer escenario (Figura 5-7 (a)) se basa en la idea de utilizar mecanismos propios de la plataforma, éstos extraen información de calidad de los servicios, a través de librerías u otros servicios disponibles, proveyendo acceso directo a datos de calidad de servicios a través de contadores y que se corresponden directamente con las métricas especificadas en el Modelo de Requisitos de Monitorización. Generalmente, estos datos o parámetros provistos por las plataformas constituyen datos de bajo nivel, los mismos que muchas veces deben ser combinados con otros.

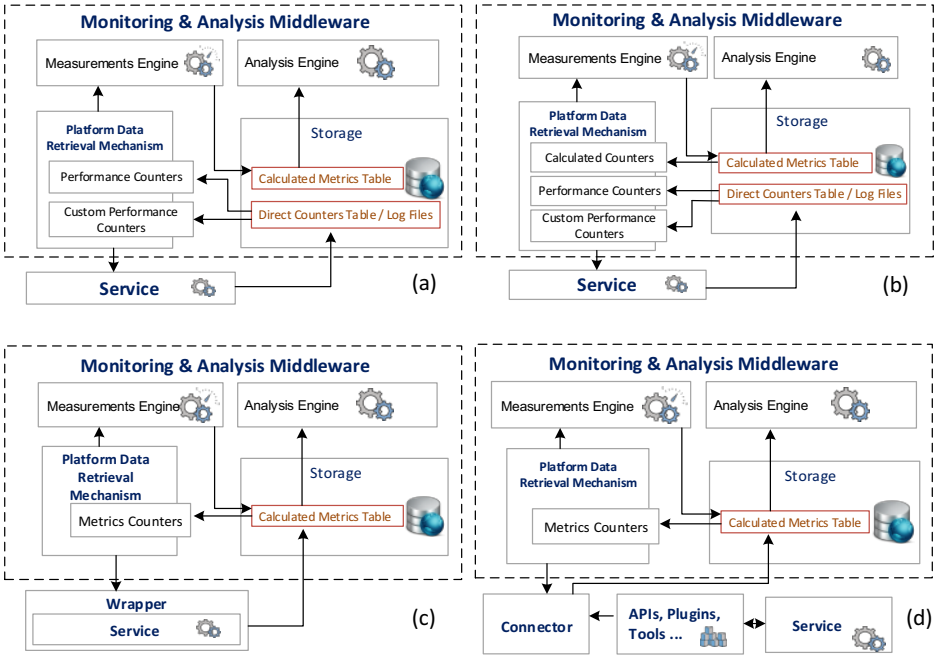


Figura 5-7 Escenarios de Captura de Información (Cedillo, et al., 2015)

Como ejemplo, existe la librería *Diagnostics* en *Azure*, que extrae y almacena datos de calidad de los servicios, por medio de contadores de rendimiento (*Performance Counters)* provistos por la plataforma a través de sus herramientas y

librerías. Estos contadores almacenan información cada cierto período de tiempo, el mismo que puede ser especificado por el usuario. Los contadores pueden ser propios de la plataforma o personalizados. Los valores de monitorización de este escenario son almacenados en la tabla *WADPerformanceCounters,* definida por Azure.

5.4.2. Obtención de datos a partir de datos existentes

El segundo escenario (Figura 5-7 (b)) ocurre ante la necesidad de realizar cálculos en base a los contadores del primer escenario e incluso de otros datos obtenidos haciendo uso de los otros escenarios, a fin de calcular una métrica. En términos de calidad, podríamos ver este escenario como la definición y aplicación de métricas indirectas (mediante la agregación de otras métricas directas o indirectas). Los cálculos son realizados por el motor de medición presentado en la sección anterior. Este método de recolección de información da lugar a los *Calculated Counters* que se muestran en la Figura 5-7 (b) y que son almacenados en la tabla *Calculated Metrics*. En conclusión, el segundo escenario está conformado por métricas indirectas calculadas a partir de datos obtenidos por otros escenarios.

5.4.3. Implementación de envoltorios para recolección de datos

El tercer escenario, permite la programación de envoltorios de servicios, extendiendo los servicios con funcionalidades que permitan generar o extraer información de monitorización de los servicios en la nube que no sea posible obtener utilizando los dos primeros escenarios. En la Figura 5-7 (c) se muestra el escenario de captura de información haciendo uso de envoltorios. Una vez que la información de monitorización es extraída, se la puede considerar como un contador del primer escenario (si su valor corresponde a la métrica a monitorizarse) o del segundo (si se realizará cálculos sobre dicha información). Este escenario a su vez, abre la posibilidad de extraer contadores muy apegados al dominio del servicio y que, al no ser de carácter general, pueden ser programados de acuerdo a la necesidad específica del usuario, que de otra forma sería imposible obtenerlos directamente de la plataforma. El hecho de incluir nueva funcionalidad al servicio en este escenario, puede generar el inconveniente de un re-despliegue del servicio lo que ocasionaría un corte momentáneo en la provisión del mismo, sin embargo, como parte del proyecto al que pertenece ésta solución, se está también proponiendo una solución de reconfiguración dinámica de servicios, lo que provee las herramientas necesarias para que este tipo de cambios puedan realizarse de manera autónoma y automática, evitando la necesidad de detener el servicio.

5.4.4. Captura de datos desde soluciones de terceros

Finalmente, el cuarto escenario está relacionado con la utilización de herramientas de terceros (conectores, APIs, etc.) para recolectar información de los servicios (ver Figura 5-7 (d)). En este caso, los datos son capturados de dichas fuentes y mediante un servicio conector, la información es recuperada y almacenada en la tabla de métricas capturadas. Al igual que en el tercer escenario, se podrá recurrir al escenario (a) o (b) dependiendo del caso de cálculo de las métricas. Como ejemplo de este escenario, podemos citar a Amazon CloudWatch, que guarda archivos log con datos de monitorización y permite almacenar y acceder a archivos log de instancias Amazon Elastic Compute Cloud (EC2), y desde los cuales capturaremos la información de monitorización mediante la definición de conectores o APIs de conversión y los almacenaremos los datos en la tabla de métricas calculadas.

Cabe destacar que tanto el tercero como el cuarto escenario, necesitan ajustes de programación en el primer caso la extensión del servicio y en el segundo la programación de un conector de recuperación de datos. Estos escenarios constituyen una gran oportunidad ya que permiten adherir código de recuperación de datos específicos así como también herramientas especializadas de monitorización, que han sido estudiadas con detenimiento de acuerdo a diferentes RNF y que pueden ser de gran utilidad a la hora de integrar y de interoperar con un gran abanico de soluciones, ahorrándonos gran cantidad de esfuerzo a cambio de codificación que muchas veces puede incluso resultar trivial, ya que podría ser incluso vista como una "migración" o "recolección" de datos de monitorización ya obtenidos y procesados hacia la solución aquí propuesta.

Además, se hace notar que estos cuatro escenarios pueden funcionar simultáneamente, creando métricas como una combinación de datos procedentes de varias fuentes para establecer el valor de una métrica en particular.

5.5. Instanciación de la infraestructura a una plataforma específica

En esta sub-sección, se mostrará cómo el diseño de la infraestructura puede ser instanciado a distintas plataformas desde una visión general, en el Capítulo 6, se realizará dicha instanciación sobre una plataforma particular a modo de un prototipo, para de esta manera la viabilidad de esta solución sobre una plataforma real.

La Figura 5-8 muestra una visión de la arquitectura, la cual presenta cada componente y su dependencia con una plataforma específica. Cada componente ha sido etiquetado con un número que lo identifica dentro de esta descripción.

El primer artefacto etiquetado con (1) es el Modelo de Calidad SaaS, el cual contiene las características, sub-características, atributos y métricas que son usadas durante la configuración para el mapeo de los RNF con una clasificación estandarizada que permite una fácil y eficiente configuración de los parámetros de monitorización. A pesar de que este contiene ciertas instrucciones dependientes a varias plataformas, sin embargo estas instrucciones representan solamente una guía durante la configuración de la monitorización. De ahí, al Modelo de Calidad SaaS, lo hemos considerado independiente de la plataforma en nuestra solución. El segundo artefacto que interviene en la infraestructura es el Modelo de Requisitos de Monitorización, el cual contiene los RNF a ser monitorizados. Estos RNF han sido obtenidos desde el SLA y RNF adicionales. Este modelo por tanto es independiente de la plataforma, ya que no incluye especificidades de la misma sino métricas generales que han sido acordadas para ser consideradas para medir la calidad de los servicios, generalmente entre un cliente o un proveedor o por una de las partes en caso de los RNF adicionales. El Configurador de la Monitorización, ha sido representado con el número (3) en la Figura 5-8, este componente presenta una interface de interacción con el usuario, la misma que lo guía para obtener el Modelo de Calidad en Tiempo de Ejecución. A pesar de que este puede ser implementado como una aplicación separada, es fuertemente recomendado mantener este componente como parte de la infraestructura de manera dependiente de la plataforma, esto porque es más útil obtener los datos dependientes de la plataforma, debido a la posibilidad de utilizarlos directamente usando las librerías y herramientas provistas por la misma plataforma sobre la que se están monitorizando los servicios. El Configurador de la Monitorización utiliza como anteriormente ya se ha citado, el Modelo de Calidad SaaS y el Modelo de Requisitos de la Monitorización como entrada. Los datos contenidos en los modelos anteriores, son automáticamente cargados en el Configurador de la Monitorización. Las acciones de cargar los modelos en el configurador, probablemente constituyan un esfuerzo importante en la implementación dependiente de la plataforma. Sin embargo, esto se implementa una sola vez por plataforma y luego pueden ser ofrecidos y consumidos como librerías, de ahí que constituye un esfuerzo fácilmente reutilizable. La creación de estas librerías puede ser vista como una extensión a la infraestructura en trabajos futuros.

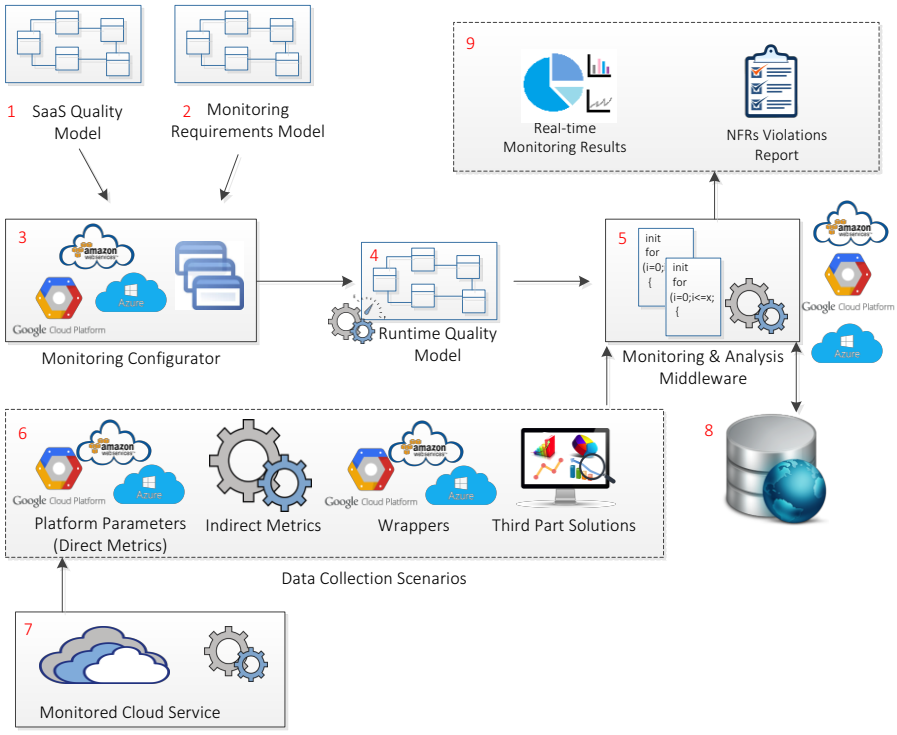


Figura 5-8 Componentes y su dependencia a una plataforma determinada

Dentro de la configuración de la monitorización y como resultado de ella se obtiene el Modelo de Calidad en Tiempo de Ejecución (4), el cual es la entrada del siguiente componente. El Modelo de Calidad en Tiempo de Ejecución contiene las directivas a ser aplicadas durante la actividad de monitorización. Este modelo tiene los elementos prescriptivos y descriptivos de todo modelo en tiempo de ejecución. Los detalles y clases correspondientes a los meta-modelos aquí utilizados han sido descritos en las sub-secciones anteriores. El Modelo de Calidad en Tiempo de Ejecución, se puede ver como independiente de la plataforma ya que no necesita ningún tipo de implementación adicional para ser creado, sin embargo contiene instrucciones dependientes de la plataforma, lo que hace que se apegue a la misma. Sin embargo su estructura no cambia dependiendo de la plataforma en la que sea utilizado. Luego el Middleware de Monitorización y Análisis (5) utiliza como entrada en Modelo de Calidad en Tiempo de Ejecución y calcula a partir de éste las métricas para medir los atributos especificados en el Modelo de Requisitos de Monitorización. Esta herramienta como se ha explicado anteriormente recoge la información y la calcula utilizando instrucciones propias de la plataforma de monitorización, por

lo que es dependiente de plataforma. Luego están los escenarios de recolección de datos desde distintas fuentes para alimentar el Middleware de Monitorización y Análisis, estas acciones se pueden hacer directamente de los servicios (información en tiempo real) o utilizando las bases de datos (8), las cuales almacenan la información así como también ciertas métricas calculadas. Finalmente, el Middleware de Monitorización y Análisis permiten las diferentes opciones de visualización para mostrar los resultados de la monitorización (9). Este último componente (9) puede ser construido usando diferentes librerías y es completamente independiente de la plataforma.

5.6. Conclusiones

En este capítulo, hemos presentado la infraestructura de monitorización que soporta el método propuesto en el capítulo anterior. Esta infraestructura permitirá el llevar a cabo todas y cada una de las actividades de monitorización planteadas, teniendo en cuenta los diferentes modelos que intervendrán en el proceso.

Se ha mostrado además, cada una de las estructuras de los modelos que se utilizarán, en los cuales tanto el Modelo de Requisitos de Monitorización, como el modelo de Calidad SaaS y el Modelo de Calidad en Tiempo de Ejecución, están basados y de acuerdo a estándares y lenguajes de dominio específico, lo que facilita y mejora grandemente una interacción totalmente automatizada y da a la infraestructura un alto nivel de formalidad, sin dejar espacio a ambigüedades tanto de requisitos como de clasificación y posterior selección de métricas y operacionalizaciones adecuadas, que se ajusten a los SLA y a tener una facilidad de selección de las formas más convenientes y apegadas a los contratos efectuados entre clientes y proveedores.

Por otra parte, se han descrito las formas previstas en esta infraestructura para recoger datos crudos desde los servicios; éstas nos permitirán interactuar de una manera sencilla con los mismos sin necesidad de adecuaciones mayores al middleware de monitorización. Ofreciendo así, un alto grado de interoperabilidad y de aprovechamiento de recursos ofrecidos por la misma plataforma de despliegue.

Finalmente, se ha mostrado cada componente y su dependencia con la plataforma, lo que guiará a los *stakeholders* en la implementación e instanciación de esta solución para una plataforma específica.

Como trabajo futuro en esta solución, se ve necesaria la implementación de métodos de auto-configuración que permitan recoger información teniendo en

cuenta ciertos aspectos como cuellos de botella, así como también se ve la necesidad de crear conectores genéricos que permitan ahorrar el mayor esfuerzo posible en recuperación de información para el cuarto escenario.

Capítulo 6
Instanciación de la Solución

Este capítulo ilustra la manera de instanciar tanto el método de monitorización de servicios en la nube como también la infraestructura de monitorización que lo soporta. Se ha escogido un caso de estudio abierto, planteado en el ámbito del proyecto SLA@SOI (McCarthy, 2011), y se han realizado las adecuaciones necesarias del mismo a fin de poder demostrar como la solución aquí planteada puede ser fácilmente utilizada en un problema real. De ahí, la sección 6.1 muestra una introducción al caso de estudio seleccionado; la sección 6.2 muestra la instanciación del método como un conjunto de actividades que conforman un proceso; la sección 6.3 presenta la instanciación de la infraestructura de monitorización en una plataforma específica; la sección 6.4 muestra la aplicación del método y la infraestructura bajo la selección de un conjunto de características de calidad. En la sección 6.5 se muestran las lecciones aprendidas tras la instanciación del método, infraestructura y aplicación práctica total de la solución y finalmente en la sección 6.6 se detallan las conclusiones del presente capítulo.

6.1. Introducción a un caso de estudio para la instanciación del método

El caso de estudio aquí presentado, es una adaptación del caso de estudio ORC (Open Reference Case) utilizado en el proyecto SLA@SOI (McCarthy, 2011), que será utilizado para la aplicación del método Cloud MoS@RT y la infraestructura de monitorización que lo soporta.

El ORC es una extensión del caso de estudio CoCoMe (Herold *et al.*, 2008), el cual provee una solución para supermercados, que maneja las ventas y el proceso de existencia de mercaderías.

El ORC resalta el uso del manejo del marco de trabajo SLA –incluyendo el proceso de negociación y renegociación del SLA- en el contexto de una solución de venta al por menor orientada a servicios que soporta las ventas en supermercados. El ORC incluye soporte IT para cadenas de reventa en general, cubriendo casas matrices (manejo central), tiendas (manejo local) y cajas. Algunos proveedores de tiendas, cada uno con un cierto número de tiendas son

conectadas a un simple proveedor de servicios, soportando las ventas de mercancía con un sistema de IT.

Este proveedor ofrece varios servicios, tales como el manejo de inventarios, pagos con tarjeta de crédito, tarjetas de clientes preferenciales y contabilidad. El software ORC puede ser operado sobre una infraestructura virtualizada usando desarrollo a medida que se adapta a tiendas de tamaño variable y con requisitos variables de acuerdo a la eficiencia del sistema.

El ORC extiende el Ejemplo Common Component Modeling Example (CoCoME), el cual fue introducido para comparar las facetas de algunos modelos bien conocidos. CoCoME representa un sistema de comercio que trata con varios aspectos de manejo de ventas en un supermercado, incluyendo la interacción con el cliente en caja (incluyendo en escaneado del producto y el pago), y registrando la venta en el inventario. El sistema de comercio CoCoME además permite realizar pedidos de mercaderías a mayoristas y generar reportes.

En CoCoMe, una empresa consiste de varias tiendas. Cada empresa tiene un servidor de empresa al cual todas las tiendas están conectadas. Un cliente de empresa es definido como parte del escenario general, permitiendo al gestor de la empresa generar algunas clases de reportes. Para brindar soporte al proceso de ventas, una tienda de venta al por menor opera un cierto número de cajas. Cada tienda tiene su propio servidor de tienda central el cual es conectado a cada caja como también al servidor de la empresa.

La caja es el lugar donde el cajero escanea la mercadería que el cliente quiere comprar y donde el pago es hecho (ya sea por tarjeta de crédito o en efectivo). Un número de componentes de hardware son asociados con una caja (el cajón de dinero, el escáner de código de barras). La unidad central de cada caja es la PC, la cual vincula todos los componentes entre sí y llama a los servicios provistos por el proveedor de la solución de venta al por menor.

La Figura 6-1 muestra a breves rasgos la arquitectura del caso de estudio presentado aquí y que será utilizado a lo largo de este capítulo con el fin de ilustrar la instanciación del método y la infraestructura. Como se puede observar en la Figura 6-1, existe una serie de servicios publicados para que clientes de diferentes cadenas de distribución de productos, puedan consumir estos servicios. En este caso, el proveedor de servicios ofrece a través de su arquitectura: servicios de inventario, servicios de información de tienda, servicio de órdenes, servicios de pago (pagos por débito, validación de tarjeta de crédito). Todo esto a nivel SaaS. Por otra parte, la solución SaaS está desplegada sobre un conjunto de servidores virtualizados y ofrecidos a través de IaaS (*Virtualized Infrastructure Provider*) a la empresa proveedora SaaS. Finalmente, la solución SaaS

se comunica con proveedores de servicios bancarios y de componentes adicionales de software; cabe señalar que, estas interacciones entre los diferentes proveedores de servicios y sus clientes, deben cumplir cláusulas de calidad, todas ellas especificadas a través de los SLA.

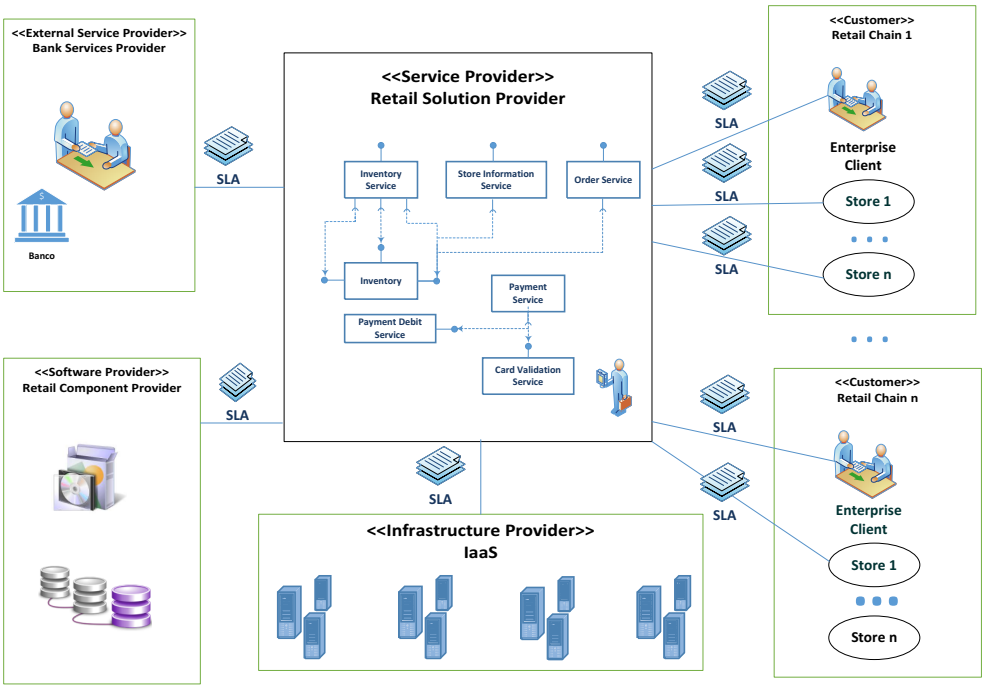


Figura 6-1 Distribución de los servicios ofrecidos en el caso de estudio ORC (Wieder *et al.*, 2011)

En el escenario promedio, el proceso de ventas es identificado como el proceso de negocio clave. Para realizar el proceso de ventas, servicios adicionales necesitan ser accedidos. Por ejemplo, las funciones ValidateCard y DebitAmount de la Figura 6-2, son provistas por el servicio externo de bancos y accedido vía el servicio ORC. De esta manera, Figura 6-2 además muestra el proceso de ventas del supermercado y sus sub-procesos. Los procesos comienzan cuando los productos son escaneados. Dentro del subproceso *Scan Good*, El código de barras de cada producto es escaneado y la operación *GetProductDetails* es llamada, todos estos procesos se encuentran en el servicio *InventoryService* presentado en la Figura 6-1. Si el código de barras del producto detectado es correcto, el nivel de stock es actualizado. Si no, el código del producto debe ser ingresado manualmente. Luego de ello, el pago por tarjeta de crédito es manejado automáticamente. La tarjeta es primero escaneada a través del método *Scan Card* del subproceso *HandlePayment* y luego validada usando el

CardValidationService. Si la tarjeta es válida, el pago es completado usando el método *DebitAmount*. Si no, la tarjeta es rechazada y el pago debe ser hecho manualmente al contado.

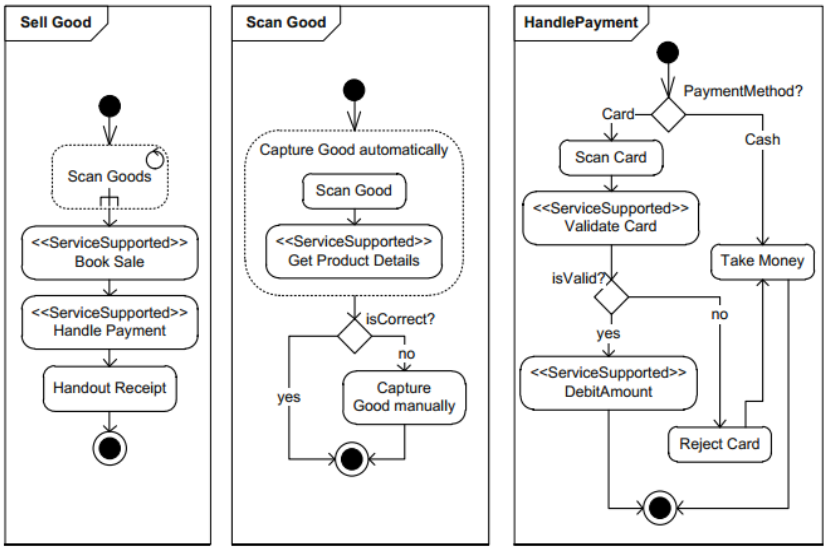


Figura 6-2 Escenarios correspondientes al proceso de ventas ORC (Wieder *et al.*, 2011)

La arquitectura interna del ORC es descrita en más detalle debajo, usando la sintaxis UML 2 y dando una vista estructural. La Figura 6-3 muestra una vista general del sistema de implementación, conteniendo los componentes heredados de CoCoMe y los servicios web adicionales.

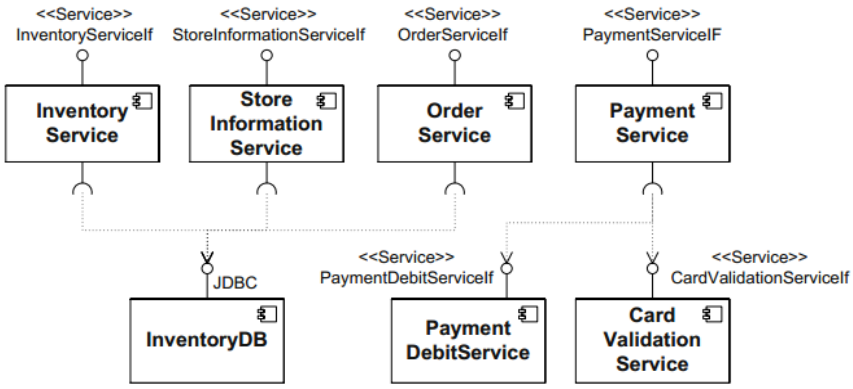


Figura 6-3 Arquitectura interna ORC (Wieder *et al.*, 2011)

Los 5 componentes *InventoryService*, *StoreInformationService*, *OrderService*, *PaymentDebitService* y *CardValidationService*, implementan interfaces de servicios web. *InventoryService*, *StoreInformationService* y *OrderService* requieren funcionalidad provista por el componente heredado CoCoMe *Inventory*.

Los servicios web y sus interfaces, los cuales forman el corazón de la implementación ORC, son descritos detalladamente en McCarthy, (2011); sin embargo y por motivos de ilustración, en este trabajo hemos descrito el servicio de inventario, sobre el cual realizaremos la monitorización y especificaremos los RNF.

Servicio de Inventario (*InventoryService*)

- *InventoryServiceInterface* define operaciones para tomar información del producto y cambiar los precios. Esta es usada para recuperar el precio de un producto, verificar su disponibilidad en el inventario y contabilizar si se ha removido desde el inventario después de la venta. La interface consiste en 5 operaciones:

- *getAllProductsWithOptionalStockItem*. Determina todos los productos en el portafolio de una tienda dada, el proveedor de cada uno de esos productos, y el número de cada producto en stock (si hubiere).
- *getProductsWithLowStock*. Determina los productos que están cercanos a agotarse, lo que significa que el número en stock es más bajo que el 10% del stock máximo. Este retorna una lista de productos y sus niveles de stock en una tienda dada.
- *getAllProducts*. Determina todos los productos con el portafolio de una tienda dada y el proveedor de cada uno de ellos. Este retorna una lista de todos los productos y sus proveedores.
- *changePrice*. actualiza el precio de ventas de un ítem en existencia. Este necesita el ítem y el nuevo precio como parámetros y retorna una instancia de *ProductWithStockItemTo*, el cual mantiene información del producto y el precio actualizado del ítem en stock, identificado por el parámetro *StockItemTo*.
- *getProductDetails*. Usa un código de barra para determinar los productos actualmente en existencia en la tienda. Este devuelve una instancia *ProductWithStockItemTo*, la cual contiene el producto ligado a un ítem en existencia en la tienda.

Para contextualizar los RNF ofrecidos sobre el servicio de inventarios presentado en este caso de estudio, las características de calidad consideradas en los SLA son las siguientes:

- La eficiencia, cuya cláusula del SLA está dada de la siguiente manera: *"El tiempo de respuesta de una petición no puede ser mayor a 500 ms"*
- La precisión del servicio, en éste caso la cláusula del SLA ofrece: *"El número de respuestas correctas será del 99.5%"*

Además, desde el punto de vista del proveedor de servicios, la carga del sistema inducida por las necesidades de volumen de cierto cliente a ser especificadas es limitada. Este conocimiento de volumen esperado por el cliente es requerido para la provisión de la infraestructura de servicios. Por esta razón, una característica importante de calidad es la carga del sistema, la cual es medida en invocaciones por segundo; de esta manera pueden existir un conjunto de RNF que pueden ser expresados en el SLA a través de cláusulas, las mismas que deben ser cumplidas por parte del proveedor, caso contrario, éste deberá sujetarse a una serie de penalizaciones que están también estipuladas en el SLA. Por ejemplo, se puede incluir en el SLA una cláusula que diga que en caso de incumplimiento, el proveedor del servicio deberá compensar al cliente con un 10% del valor total de su facturación, siempre que éste sea expresamente solicitado en un plazo máximo de 30 días posterior a la detección de la falta.

Normalmente, no existen métodos utilizados por los proveedores que alerten al cliente acerca de estos incumplimientos del SLA, de ahí que es importante para el usuario el contar con una herramienta que pueda indicarle los problemas durante la provisión de los servicios. Así también, para el proveedor es de vital importancia conocer cómo está proveyendo los servicios a sus clientes, dado que de esta manera puede mejorar la calidad de su servicio, evitando así descontentos por parte de sus usuarios y creándose una reputación buena que demuestre la provisión de soluciones óptimas, que den confianza y atraigan a nuevos usuarios, así como también manteniendo a sus usuarios fidelizados y satisfechos con sus servicios.

Los primeros pasos que realizaremos en la instanciación de la solución aquí presentada serán la instanciación del método, sus pasos: construcción del Modelo de Requisitos de Monitorización, configuración de la monitorización, obtención del Modelo de Calidad en Tiempo de Ejecución con los valores que abarquen este caso de estudio y la construcción del middleware, considerando el servicio a ser monitorizado en la plataforma Microsoft Azure. De ahí, aplicaremos ciertos escenarios de recuperación de información y finalmente la monitorización de los servicios y sus resultados.

6.2. Instanciación del Método

Teniendo como base el proceso de monitorización presentado por Cedillo *et al. (*2014), el mismo que a lo largo de las demás etapas de este trabajo se ha ido refinando, hasta llegar a involucrar a todos artefactos necesarios para monitorización de los servicios cloud a nivel de SaaS, en esta sub-sección se ha instanciado este proceso de una forma particular al dominio del caso de estudio introducido en la sección anterior.

En la Figura 6-4 mostraremos el proceso involucrando cada uno de los artefactos que formarán parte del mismo y serán relativos a este dominio.

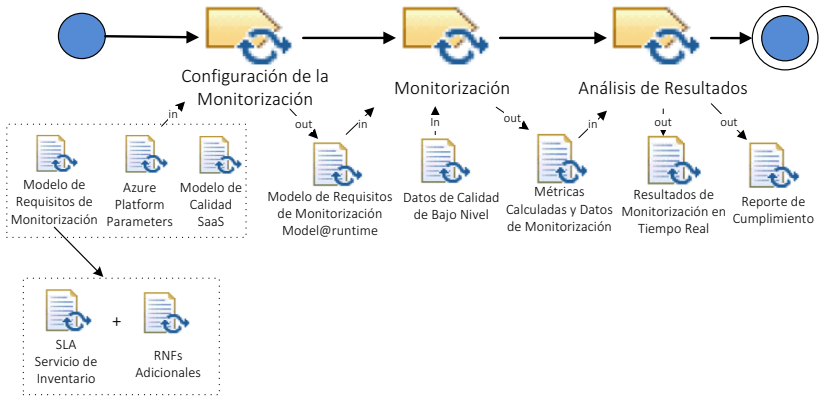


Figura 6-4 Proceso de Monitorización Cloud MoS@RT instanciado al caso de estudio ORC

6.2.1. Instanciación de la configuración de la monitorización

En primer lugar, se deben tener en cuenta los requisitos no funcionales a ser monitorizados, de tal manera que se pueda con éstos conformar el Modelo de Requisitos de Monitorización. Según lo planteado anteriormente, los requisitos de monitorización a ser monitorizados con fines de ilustración son:

- La eficiencia, cuya cláusula del SLA está dada de la siguiente manera: *"El tiempo de respuesta de una petición no puede ser mayor a 500 ms"*
- La precisión del servicio, en éste caso la cláusula del SLA ofrece: *"El número de respuestas correctas será del 99.5%"*

Aquí, el Planificador de la Monitorización, (ver sección 4.2.1.1 Establecimiento de los requisitos de monitorización), es el encargado de establecer los requisitos de monitorización en un formato "comprensible" a la infraestructura, este es el Modelo de Requisitos de Monitorización, el mismo que

como ya se ha comentado anteriormente, está expresado en una extensión del lenguaje WSLA.

6.2.1.1. Construcción del modelo de requisitos de monitorización

A continuación, se presenta un fragmento del SLA, instanciado para este caso de estudio.

Este fragmento es una representación en XML del modelo de requisitos de monitorización, expresada en la extensión del WSLA plantaeada en esta solución.

Cabecera del SLA:

```
<?xml version="1.0"?>
<wsla:SLA xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:wsla="http://www.ibm.com/wsla" name="OpenReferenceCaseExample" >
```

Declaración de las partes que intervienen:

Proveedor de Servicios

```
<Parties>
 <ServiceProvider name="CoCoMeORCProvider">
  <Contact>
   <Street>Tarongers Ave CP 46022 </Street>
   <City>Valencia Capital, Valencia, Spain </City>
  </Contact>
    <Action name="notification" partyName="AuditorandServices"
        xsi:type="WSDLSOAPActionDescriptionType">

    <WSDLFile>notification.wsdl</WSDLFile>
       <SOAPBindingName>SOAPNotificationBinding</SOAPBindingName>
       <SOAPOperationName>notification</SOAPOperationName>
  </Action>
 </ServiceProvider>
```

Consumidor del Servicio

```
 <ServiceCustomer name="SuperMarketA">
  <Contact>
   <Street> Ave. Primero de Mayo </Street>
   <City>Cuenca, Ecuador </City>
  </Contact>
  <Action name="notificationCustomer" partyName="AuditorandServicesMeasurements"
            xsi:type="WSDLSOAPOperationDescriptionType">
            <WSDLFile>notificationCustomer.wsdl</WSDLFile>
       <SOAPBindingName>SOAPNotificationBinding</SOAPBindingName>
       <SOAPOperationName>notification</SOAPOperationName>
  </Action>
  </ServiceCustomer>
```

Partes de Auditoría del Servicio

```
<SupportingParty name="AuditorandServicesMeasurements"
  role="Amsterdam Street CP 46023"
  sponsor="CoCoMeORCProvider">
 <Contact>
  <Street>Amsterdam Street CP 46023 </Street>
  <City>Zurich, Switzerland</City>
 </Contact>
 <Action name="monitoringUpdate"
  partyName="AuditorandServicesMeasurements"
  xsi:type="WSDLSOAPOperationDescriptionType">
  <WSDLFile>monitoringUpdate.wsdl</WSDLFile>
  <SOAPBindingName>SOAPNotificationBinding</SOAPBindingName>
  <SOAPOperationName>monitoringUpdate</SOAPOperationName>
 </Action>
</SupportingParty>
<SupportingParty name="XYZAuditing"
  role="ConditionEvaluationService"
  sponsor="CoCoMeORCProvider">
 <Contact>
  <Street>France Avenue</Street>
  <City>Zurich, Switzerland</City>
 </Contact>
 <Action name="monitoringUpdate"
  partyName="AuditorandServicesMeasurements"
  xsi:type="WSDLSOAPOperationDescriptionType">
  <WSDLFile>monitoringUpdate.wsdl</WSDLFile>
  <SOAPBindingName>SOAPNotificationBinding</SOAPBindingName>
  <SOAPOperationName>reqInfoMonitor</SOAPOperationName>
 </Action>
</SupportingParty>
</Parties>
```

Definición del Servicio

```
<!-- The definition of the service in terms of the service parameters and their measurement. -->
<ServiceDefinition name="InventoryService">
 <Schedule name="FiveMinutesSchedule">
  <Period>
   <Start>2014-12-01T14:00:00.000-05:00</Start>
   <End>2014-12-31T14:00:00.000-05:00</End>
  </Period>
  <Interval>
   <Minutes>
    5</Minutes>
   </Interval>
 </Schedule>
 <Schedule name="hourlySchedule">
  <Period>
   <Start>2014-12-01T14:00:00.000-05:00</Start>
   <End>2014-12-31T14:00:00.000-05:00</End>
  </Period>
```

```
<Interval>
 <Minutes>
  60</Minutes>
 </Interval>
</Schedule>
```

Operaciones que intervienen la provisión del servicio

```
<Operation name="NewItemInventory">
 <SLAParameter name="ResponseTime" type="Float" unit="Milliseconds">
  <Metric>ResponseTime</Metric>
  <Communication>
   <Source>CoCoMeORCProvider</Source>
   <Pull>XYZAuditing</Pull>
   <Push>XYZAuditing</Push>
  </Communication>
 </SLAParameter>

 <SLAParameter name="ServiceAccuracy" type="Float" unit="Percentage">
  <Metric>SA_ServiceAccuracy</Metric>
  <Communication>
   <Source>CoCoMeORCProvider</Source>
   <Pull>XYZAuditing</Pull>
   <Push>XYZAuditing</Push>
  </Communication>
 </SLAParameter>

<Metric name="ResponseTime" type="float" unit="milliseconds">
  <Schedule>hourlySchedule</Schedule>
  <Source>AuditorandServiceMeasurements</Source>
   <Metric>ExecutionTime</Metric>
</Metric>

<Metric name="SA_ServiceAccuracy" type="float" unit="milliseconds">

<Schedule>hourlySchedule</Schedule>
   <Source>AuditorandServiceMeasurements</Source>
    <Function type="Division" resultType="float">
     <Operand> <Metric>NumberofCorrectResponses</Metric> </Operand>
     <Operand> <Metric>NumberofTotalResponses</Metric> </Operand>
   </Function>
</Metric>
</ServiceDefinition>
```

Obligaciones del Proveedor de Servicios

```
<Obligations>
        <ServiceLevelObjective name= "ConditionalSLOForResponseTime">
         <Obliged> CoCoMeORCProvider </Obliged>
         <Validity>
                <Start>2014-11-30T14:00:00.000-05:00</Start>
                <End>2014-12-31T14:00:00.000-05:00</End>
         </Validity>
         <Expression>
```

```
            <Implies>
                <Predicate xsi:type= “Less”>
                    <SLA Parameter>ResponseTime</SLA Parameter>
                    <Value>500</Value> <!--500ms>
                </Predicate>
            </Implies>
          </Expression>
        </ServiceLevelObjective>
        <ServiceLevelObjective name= “ConditionalSLOServiceAccuracy>
          <Obliged> CoCoMeORCProvider </Obliged>
          <Validity>
                <Start>2014-11-30T14:00:00.000-05:00</Start>
                <End>2014-12-31T14:00:00.000-05:00</End>
          </Validity>
          <Expression>
            <Implies>
                <Predicate xsi:type= “Greater”>
                    <SLA Parameter>ServiceAccurracy</SLA Parameter>
                    <Value>9.995</Value> <!--Mayor al 99.5% de respuestas correctas>
                </Predicate>
            </Implies>
          </Expression>
        </ServiceLevelObjective>
</Obligations>
```

Acciones que garantizan la correcta provisión del servicio

```
<ActionGuarantee name= “GuaranteeResponseTime”>
  <Obligued > XYZAuditing </Obligued>
  <Expression>
    <Predicate xsi:type= “Violation”>
          <ServiceLevelObjective>ConditionalSLOForResponseTime</ServiceLevelObjective>
    </Predicate>
  </Expression>
  <EvaluationEvent>NewValue</EvaluationEvent>
  <QualifiedAction>
          <Party> CoCoMeORCProvider </Party>
          <Action actionName="penaltee1" xsi:type="Notification">
          <NotificationType>Violation</NotificationType>
          <CausingGuarantee>ResponseTimeGuarantee </CausingGuarantee>
          <SLAParameter>ResponseTime</SLAParameter> </Action>
  </QualifiedAction>
  <ExecutionModality>Always</ExecutionModality>
</ActionGuarantee>
<ActionGuarantee name= “GuaranteeServiceAccuracy”>
  <Obligued > XYZAuditing </Obligued>
  <Expression>
    <Predicate xsi:type= “Violation”>
          <ServiceLevelObjective>ConditionalSLOForServiceAccuracy</ServiceLevelObjective>
    </Predicate>
  </Expression>
  <EvaluationEvent>NewValue2</EvaluationEvent>
  <QualifiedAction>
```

```
        <Party> CoCoMeORCProvider </Party>
        <Action actionName="penaltee2" xsi:type="Notification">
        <NotificationType>Violation</NotificationType>
        <CausingGuarantee>ServiceAccuracyGuarantee </CausingGuarantee>
        <SLAParameter>ServiceAccuracy</SLAParameter> </Action>
 </QualifiedAction>
 <ExecutionModality>Always</ExecutionModality>
</ActionGuarantee>
</wsla:SLA>
```

6.2.1.2. Modelo de calidad SaaS

El Modelo de Calidad SaaS, servirá de soporte durante la configuración de la monitorización, siendo este un artefacto de entrada a esta tarea. A continuación, se presenta un extracto de este modelo para con él realizar el mapeo de las características, sub-características, atributos y métricas de se utilizarán para la formulación del Modelo de Calidad en Tiempo de Ejecución.

Tabla 6-1. Fragmento de un Modelo de Calidad SaaS

Caracte-rísticas	Sub-Carac.	Atributos	Métricas	Operacionalizaciones
Reliability		Maturity		
		Availability	Robustness of a Service (ROS)	Available Time for Invoking SaaS/Total Time for Operating SaaS
			Metric of Availability	(Agreed Service Time - Outage Downtime)/Agreed Service Time
				Uptime/Agreed Service Time
		Fault Tolerance	Defective Operations Per Million (DPM)	((Operations Attempted - Operations Successful)/Operations Attempted)*10^6
				(Operations Failed / Operations Attempted)*10^6
				(Oprations Failed / (Operations Successful+Operations Failed))*10^6
		Service Stability	Coverage of Fault Tolerance (CFT)	Number of Faults Without Become Failures / Total Faults Occured
			Coverage of Failure Recovery (CFR)	Number of Failures Remedied / Total Number of Failures

Caracte-rísticas	Sub-Carac.	Atributos	Métricas	Operacionalizaciones
		Service Accuracy	Service Accuracy (SA)	Number of Correct Responses / Total Number of Requests
Performance Efficiency	Time Behavior	Response Time	Latency	Request Execution Time + Request Response Time
			Request Execution Time	Execution Time
			Time Behavior (TB)	Execution Time / Total Service Invocation Time
		Data Exchange Workload		
	Resource Utilization	Elasticity in Resources	Coverage of Scalability (COS)	Sum(Amount of Allocated Resources of ist Request / Total Amount of Requested Resources of ist Request)/Total Requests for Extending Resources Used
		Optimization in the use of resources		
	Capacity	Concurrent Users		
		Throughput of Services	Throughput	Number of Successful Transactions per Hour

6.2.1.3. Parámetros de la plataforma Azure

Dentro del proceso de monitorización, como artefacto de entrada a la configuración de la monitorización, se tiene una lista de parámetros de la plataforma, métricas indirectas, envoltorios y soluciones de terceros, los mismos que serán listados con la finalidad de poder elaborar el modelo en tiempo de ejecución. Éstos representan las fuentes de datos que se mapearán con los operandos de las métricas cuyas operacionalizaciones han sido seleccionadas, a fin de que, en tareas posteriores realizar los cálculos necesarios para medir la calidad de los atributos. Como ejemplo para esta instanciación, se han recogido en las tablas presentadas a continuación (Tabla 6-2 y Tabla 6-3) los siguientes parámetros, de los cuales se deberán seleccionar los más adecuados al momento del mapeo, teniendo en cuenta los RNF que están descritos en el Modelo de Requisitos de Monitorización.

En este caso, se seleccionarán los más aproximados a la función de medición, se han puesto algunos adicionales con la finalidad de entender cómo esto se

presentará al usuario para realizar los mapeos necesarios. Se debe tener en cuenta que la operacionalización seleccionada debe ser lo más cercana posible a la pactada en el SLA o descrita por el proveedor en los RNF adicionales.

Para el RNF 1, los contadores más cercanos podrían ser:

$$Execution\ Time = Request\ Execution\ Time$$

Tabla 6-2. Conjunto de Parámetros de Bajo Nivel Elegibles RNF1

Parámetros de Recolección de Datos
\ASP.NET\Request Execution Time
\Web Service(*)\Service Uptime
\ASP.NET\Requests Current
\ASP.NET Applications(*)\Sessions Active
\CUSTOM\Total Time Running Application

Para el RNF 2, los contadores más cercanos podrían ser:

$$SA = \frac{Number\ of\ Correct\ Responses}{Number\ of\ Total\ Responses}$$

Tabla 6-3. Conjunto de Parámetros de Bajo Nivel Elegibles RNF2

Parámetros de Recolección de Datos
\ASP.NET Applications(*)\Errors Total
\ASP.NET Applications(*)\Requests Total
\ASP.NET Applications(*)\Requests Succeeded
\ASP.NET Applications(*)\Sessions Active
\ASP.NET Applications(*)\Requests Failed

6.2.1.4. Configuración de la monitorización

En esta sección se presenta la configuración de la monitorización a manera de proceso, en las próximas subsecciones se presentará la instanciación de este componente en una plataforma específica, como parte de la infraestructura de monitorización.

Teniendo al proceso en consideración y habiendo ya presentado los artefactos de entrada y su estructura, queda por realizar el mapeo de los RNF expresados en el Modelo de Requisitos de Monitorización de la sección 6.2.1.1 con los parámetros que obtienen los datos crudos de los servicios presentados

en la sección 6.2.1.3 y haciendo uso del Modelo de Requisitos de Monitorización de la sección 6.2.1.2. con la finalidad de obtener el Modelo de Calidad en Tiempo de Ejecución. Este mapeo se ejecutará de la siguiente manera:

En el caso del RNF 1, la Figura 6-5 muestra el mapeo entre el RNF expresado en el Modelo de Requisitos de Monitorización y el parámetro que se obtiene de la plataforma. Este ejemplo pertenece al primer escenario de captura de datos de monitorización, en el que se ha encontrado una métrica que permite la operacionalización del RNF1 de manera directa.

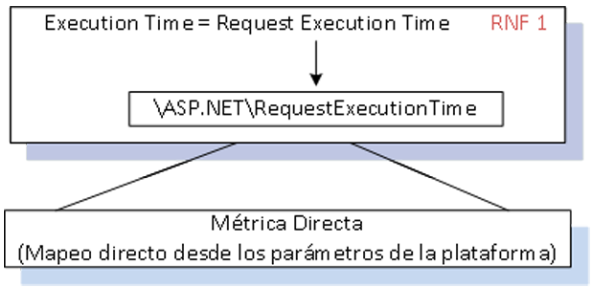


Figura 6-5. Mapeo del RNF1 a realizarse durante la configuración de la monitorización

Por otra parte, en el caso del RNF 2, La Figura 6-6 muestra el mapeo entre el RNF 2, expresado en el Modelo de Requisitos de Monitorización y los datos obtenidos desde los servicios. En este ejemplo, se trata del segundo escenario de recolección de datos, en el cual se realiza el cálculo de la métrica, a partir de dos métricas directas obtenidas desde parámetros propios de la plataforma.

Una vez que el mapeo es realizado, estos datos pasan a formar parte del Modelo de Calidad en Tiempo de Ejecución.

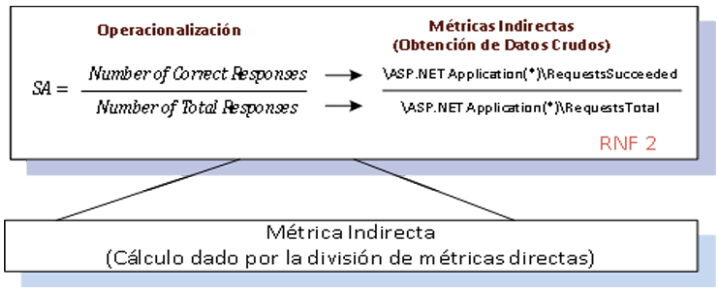


Figura 6-6. Mapeo del RNF2 a realizarse durante la configuración de la monitorización

6.2.1.5. Obtención del modelo de calidad en tiempo de ejecución

Teniendo en cuenta que el Modelo de Calidad en Tiempo de Ejecución, contiene un subconjunto de clases del Modelo de Calidad SaaS, sumadas a las

partes prescriptiva y descriptiva propias de un modelo en tiempo de ejecución y de acuerdo al ejemplo que se aplica a lo largo de la instanciación presentada en este capítulo, la parte modular de modelo en tiempo de ejecución estará dada por las siguientes clases y los siguientes datos relacionados, los cuales han sido mostrados en un formato más legible presentado a continuación, en vista de la dificultad que conlleva leer el modelo en el formato que genera el lenguaje propio de modelado, así también se presentan las clases más importantes y su contenido, proveniente del mapeo realizado en los pasos anteriores. En este ejemplo se utiliza una nomenclatura bastante simple que permita mostrar el funcionamiento principal del modelo de calidad en tiempo de ejecución. Cabe destacar que, en caso de renegociación del SLA o cualquier cambio dentro de los requisitos, este modelo cambiará sus valores. Dado que la infraestructura consume este modelo, la misma al enfrentarse a los cambios de los RNF no necesitará actualización alguna en su código. El modelo completo en tiempo de ejecución puede ser revisado en el Anexo II, sin embargo aquí se mostrarán las partes esenciales para un mejor entendimiento.

Clase QualityRuntimeModel
name: M@RT_InventoryServiceOCR
description: M@RT del servicio de inventarios ORC-CoCoMe

Clase ConfigurationFile
address: "https://cocome.servicebus.cocome.net/Cocome
contract: "Service.InventoryService"
binding: "ws2015HttpRelayBinding"
securityKey: "3463kjfgksj3847837639"
storageConnection: "storeCocomeMonitor"

Para el NFR1:

Clase NFR
name: "ResponseTime <500ms"
description: "The response time is lower than 500ms"
nfrReference: "1"
Clase Characteristic
name: "Efficiency"
Clase Characteristic (Sub-Characteristic)
name: "Time Behaviour"
Clase Attribute
name: "ResponseTime"
definition: "Definition of Response Time"

Clase Metric
name: "Response Execution Time"
description: "Definition of Response Execution Time"
Clase Operationalization
description: "Execution Time"
level: SaaS
pers: serviceProvider
Clase Unit
name: milliseconds

Para el NFR2:

Clase NFR
name: "Service Accuracy >99.5%"
description: "The correct answers should be >99.5%"
nfrReference: "2"
Clase Characteristic
name: "Reliabililty"
Clase Attribute
name: "Service Accuracy"
definition: "Accuracy of the service"
Clase Metric
name: "Service Accuracy"
description: "Definition of the service accuracy"
Clase Operationalization
description: "NumberofCorrectResponces/TotalNumberofRequests"
level: SaaS
pers: serviceProvider
Clase Unit
name: percentage

6.2.2. Instanciación de la actividad de monitorización

Una vez que se ha realizado la configuración de la monitorización, se obtiene como resultado el Modelo de Calidad en Tiempo de Ejecución. Éste constituye un artefacto de entrada para la actividad de monitorización, en la que se realizará automáticamente las operaciones y los cálculos necesarios para calcular las métricas contenidas en el Modelo de Calidad en Tiempo de Ejecución.

Esta tarea está constituida por un middleware de monitorización que se encarga además de recoger los datos del servicio especificado, en este caso del *Inventory Service* para ser monitorizado. Sobre los datos recogidos del servicio se

realizarán las operaciones descritas en las diferentes operacionalizaciones de las métricas. Éstas son dependientes de la plataforma.

El objetivo de esta actividad será transformar los datos de bajo nivel en métricas de alto nivel. Estos datos serán almacenados y/o reportados al Motor de Análisis, el cual se encargará de comparar las medidas obtenidas con los umbrales fijados en los RNF.

Al ser éste un proceso automático, se especificará en detalle en la parte de la instanciación de la infraestructura. La instanciación de este componente, se verá reflejada en un prototipo construido para demostrar la fiabilidad de este método. El prototipo ha sido realizado dentro del trabajo de fin de carrera de Jimenez (2015), realizado en la Universidad Politécnica de Valencia.

6.2.3. Instanciación de la actividad de presentación de resultados

La actividad de presentación de resultados, se encarga de mostrar cada uno de los resultados obtenidos tras el proceso de monitorización, a manera de un informe de cumplimiento o en forma de un *dashboard* que muestra los resultados provenientes de los datos monitorizados, una vez que estos han sido procesados y constituyen resultados válidos de la monitorización de los RNF propuestos.
Esta actividad es así mismo automática y se instanciará como parte de un trabajo futuro.
Para realizar esta actividad, se contará con herramientas y librerías ya existentes, las mismas que serán consumidas a fin de presentar los resultados de monitorización a manera de cuadros estadísticos, gráficos en tiempo real o diagramas de radar.

6.3. Instanciación de la infraestructura de monitorización

En esta sección se muestra la implementación de la infraestructura de monitorización, construida a través de un prototipo que ha sido presentado en el trabajo de Jimenez (2015). Este prototipo ha sido construido en la forma de una aplicación web para la parte del configurador que, junto al middleware de monitorización y análisis será desplegado a manera de un servicio en la nube, en la plataforma Microsoft Azure. En esta sub-sección, primero se expondrán los requisitos de las herramientas, luego se realizará un análisis de la plataforma seleccionada para la implementación y su influencia al momento de la construcción del prototipo. Luego se mostrará el diseño y la realización de la aplicación web y sus características. Finalmente, se mostrará la implementación del middleware de monitorización, y se aplicará todo al caso de estudio ORC-CoCoMe presentado en las sub-secciones anteriores.

6.3.1. Implementación del configurador como un servicio desplegado en la nube

El configurador de la monitorización se ha construido en el prototipo a manera de una aplicación web, cuyo objetivo es asistir al usuario durante la tarea de configuración de la monitorización. Ésta da soporte a las diferentes sub-tareas de la configuración de la monitorización presentada anteriormente, cuyo artefacto de salida será el modelo en tiempo de ejecución que será luego consumido por el middleware de monitorización y análisis.

En esta sección se ha tomado en cuenta el configurador de la monitorización a manera de una aplicación desarrollada con .NET, y disponible en la nube a manera de servicio. Se ha tenido en consideración esta característica dada la necesidad de contar con herramientas que no necesiten ser instaladas ni configuradas y que funcionen en armonía con los servicios, tratando de sacar el máximo provecho de la plataforma a través de contadores disponibles.

Los objetivos de esta herramienta serán:

- Utilizar los RNF especificados en el SLA y los que son de interés al usuario de la herramienta de monitorización para integrarlos en un Modelo de Requisitos de Monitorización.
- Se capaz de contener un modelo de calidad SaaS previamente establecido que ayude al usuario a mapear los RNF para obtener un Modelo de Calidad en Tiempo de Ejecución que se enmarque dentro de un estándar de calidad, a fin de que sea fácilmente configurada la monitorización.
- Seleccionar de entre las métricas correspondientes a los atributos del modelo de calidad la más apropiada y aproximada a aquella especificada en el Modelo de Requisitos de Monitorización.
- Ser capaz operacionalizar las métricas seleccionadas teniendo como operandos los parámetros y datos crudos recogidos desde los servicios.
- Generar el modelo en tiempo de ejecución a ser consumido por el Middleware de Monitorización y Análisis.

6.3.1.1. Patrón de diseño

El configurador de la monitorización ha sido implementado por (Jimenez, 2015) siguiendo el patrón de diseño de interfaces de usuario del Asistente (Wizard) (Jimenez, 2015), este patrón tiene como objetivo dividir una tarea en varias sub-tareas más pequeñas para que sea más sencillo completar la tarea principal.

El autor del prototipo ha tomado como guía el diagrama del proceso de configuración de la monitorización de Cloud MoS@RT y ha seguido su carácter lineal, atravesando cada una de sus tareas. El proceso debe ser implementado de una manera fácil y bastante sencilla para el usuario, de tal manera que permita sin problemas obtener el Modelo de Calidad en Tiempo de Ejecución. De igual manera, debido a la complejidad de las acciones a realizar y de los conceptos empleados, es necesario que el usuario se guía a través de un "asistente" que permita indicarle las acciones que debe tomar.

En versiones preliminares se han mostrado ciertas particularidades del configurador de la monitorización, hasta llegar a una versión más terminada y comprensible al usuario. En la Figura 6.7 se presenta una versión inicial de un configurador de la monitorización, la misma que fue descartada por su complejidad y falta de asistencia al momento de realizar las configuraciones por parte del usuario.

Como indica la Figura 6.7, se debe cargar el XML que contiene el Modelo de Requisitos de Monitorización, el mismo que en una versión previa se vio representado por el SLA. Además, se tenía en consideración los tres actores del SLA (proveedor, cliente y partes de soporte), y luego se desplegaba la lista de operaciones del servicio, los parámetros que contiene el SLA y las métricas a ser utilizadas las mismas que eran mapeadas con los parámetros provenientes de los escenarios de recolección de datos. El problema en esta interface es que era muy poco intuitiva y el usuario debía conocer exactamente las acciones a tomar para poder realizar la configuración de la monitorización.

Luego, se realizó un conjunto de interfaces para medir la percepción de facilidad de uso, utilidad e intención de uso futura, en este prototipo de interfaces, se evaluó la configuración de la monitorización (el proceso de evaluación y los resultados se muestran en el Capítulo 7). Estas interfaces fueron el preámbulo que nos llevó al diseño final del prototipo de la infraestructura. Así como también, nos sirvieron para realizar experimentos controlados sobre la percepción de usuario. En estas interfaces, lo que interesaba era medir las características antes mencionadas, por lo que por temas de acceso a la plataforma, se optó por un diseño preliminar de interfaces realizado en una macro. Luego este diseño fue implementado como un servicio a modo de una aplicación Web. Esta aplicación Web final del configurador se muestra en la Figura 6-8. Este diseño final presenta el habitual diseño de interface presente en aplicaciones que utilizan el patrón antes mencionado, en el cual los formularios a rellenar utilizan la parte central de la pantalla y en la parte inferior se encuentran los botones de avance y retroceso en el proceso, para poder navegar entre tareas.

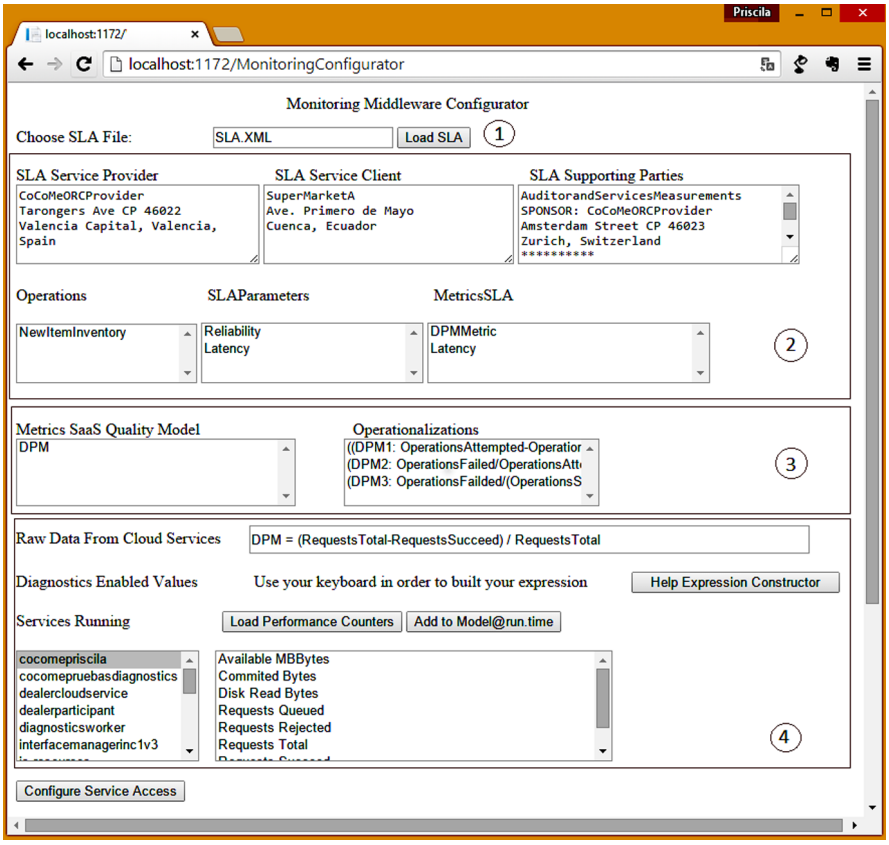


Figura 6-7. Prototipo inicial del configurador de la monitorización

El conjunto de pasos a seguir a fin de lograr la implementación del configurador de la monitorización están definidos siguiendo los pasos presentados en el proceso de monitorización y teniendo en cuenta todos los detalles necesarios para lograr la generación del Modelo de Calidad en Tiempo de Ejecución. El flujo de la aplicación se ve ilustrado a través del diagrama presentado en la Figura 6-9.

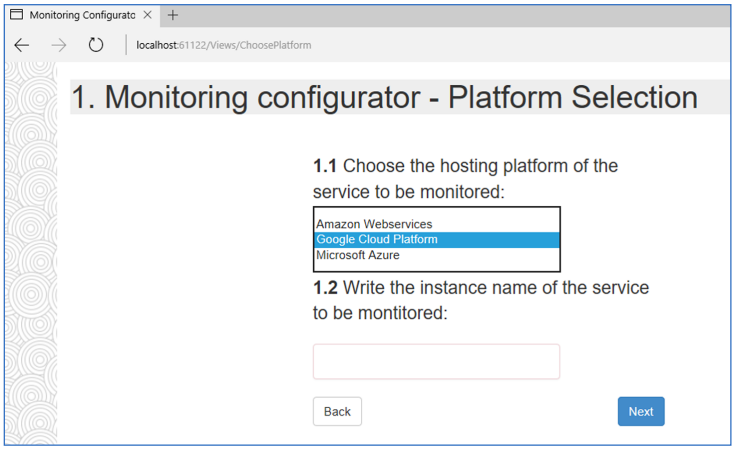


Figura 6-8. Prototipo de la interface con el patrón asistente (Jimenez, 2015)

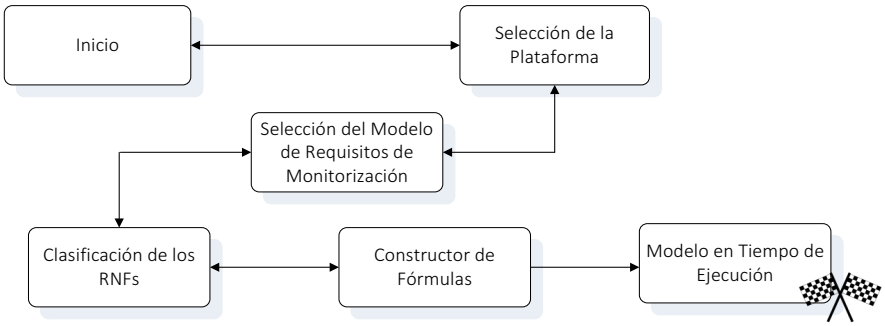


Figura 6-9. Flujo del Configurador de la Monitorización

6.3.1.2. Selección de la plataforma

En la interface de selección de la plataforma, el usuario debe introducir los datos del servicio a monitorizar. Para ello, seleccionará la plataforma en la que el servicio está desplegado de entre las soportadas por la aplicación, en este prototipo se da soporte a Microsoft Azure. A continuación, se deben introducir las credenciales propias del servicio (nombre, identificador del despliegue y cadena de conexión), valores que servirán para reflejarlos en el Modelo de Calidad en Tiempo de Ejecución, correspondientes a la conexión causal de la herramienta de monitorización con el servicio a ser monitorizado. El identificador del despliegue determina únicamente el servicio dentro de la plataforma, mientras que la cadena de conexión incluye la información necesaria para conectarse al recurso de almacenamiento en el cual los datos de monitorización serán almacenados. Recurso de almacenamiento que será utilizado por el Middleware para realizar los cálculos posteriores cuando se

presentan funciones de agregación o necesidad de datos históricos útiles para los reportes.

En la Figura 6-10 se presenta la interface de selección de la plataforma.

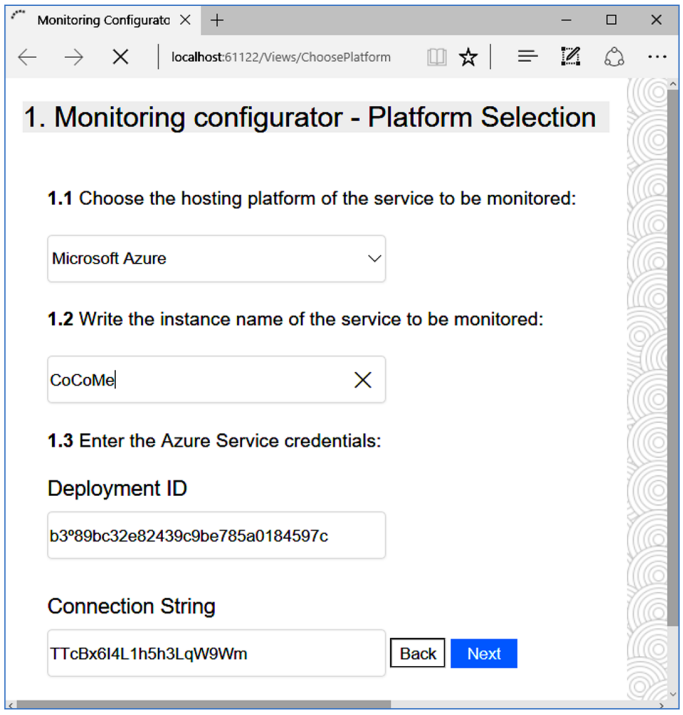


Figura 6-10. Pantalla de Selección de la Plataforma (Jimenez, 2015)

6.3.1.3. Selección del modelo de requisitos de monitorización

El modelo de requisitos de monitorización es uno de los artefactos utilizados en el proceso de configuración de la monitorización, éste es realizado por el rol del Planificador de la Monitorización. En secciones anteriores se presentó su construcción. Como se había discutido anteriormente, el Modelo de Requisitos de Monitorización contiene la información relativa a los requisitos no funcionales a monitorizar que será necesaria para establecer los mapeos y de ahí las métricas que permitirán realizar las mediciones de estos requisitos.

La Figura 6-11 muestra la captura del Modelo en la cual se ha optado por la selección de este modelo en formato XMI (XML Metadata Interchange), este es un formato estándar ISO/IEC 19509:2014 que tiene como objetivo principal "permitir el fácil intercambio de metadatos entre aplicaciones del proceso de vida del desarrollo (como herramientas de modelado basadas en Unified

Modeling Language (UML) y repositorios de metadatos o *frameworks* basados en Meta Object Facility (MOF) en entornos distribuidos y heterogéneos".

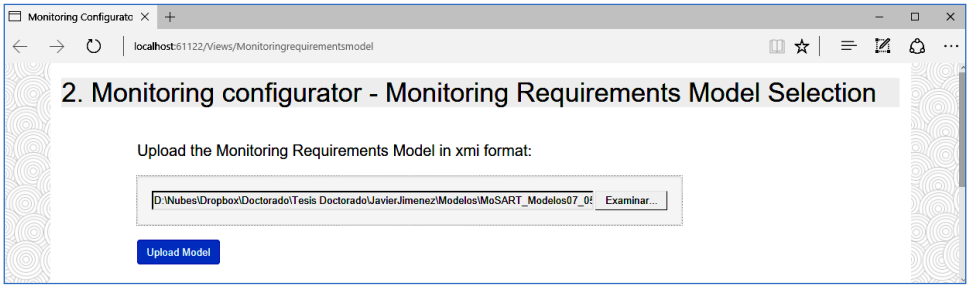


Figura 6-11. Pantalla de Selección del Modelo de Requisitos de Monitorización (Jimenez, 2015)

El empleo de XMI en esta solución tiene como objetivo el conservar la universalidad del modelo para su uso desde cualquier tipo de aplicación, de ahí que se lo ha considerado como contenedor de información de configuración. La dificultad de utilizar XMI fue la manera de consumir los datos programando un "*parser*" que permita rescatar la información desde el modelo, sin embargo una vez realizado el algoritmo que permita el consumo del modelo, la complejidad se vio considerablemente minimizada en los pasos siguientes del prototipado de esta solución.

Una vez que el usuario ha escogido el Modelo de Requisitos de Monitorización, y lo ha seleccionado en el explorador, en la parte inferior de la pantalla se presenta la información de los requisitos de monitorización, la misma que está conformada por:

- **NFR Name**: Nombre del requisito no funcional.
- **Threshold**: Umbral del requisito no funcional, el mismo que puede describir la condición de >,< o = del dato obtenido con respecto a la condición esperada.
- **Metric Name**: Nombre de la métrica o métricas seleccionadas para medir el RNF seleccionado.
- **Metric Formula**: Función de medición en lenguaje natural que expresa la manera en la que se medirá la métrica. Una métrica puede medirse con una o más funciones de medición.

La interface con el modelo de calidad cargado al configurador se presenta la Figura 6-12, en la cual se muestra cada una de las partes antes descritas.

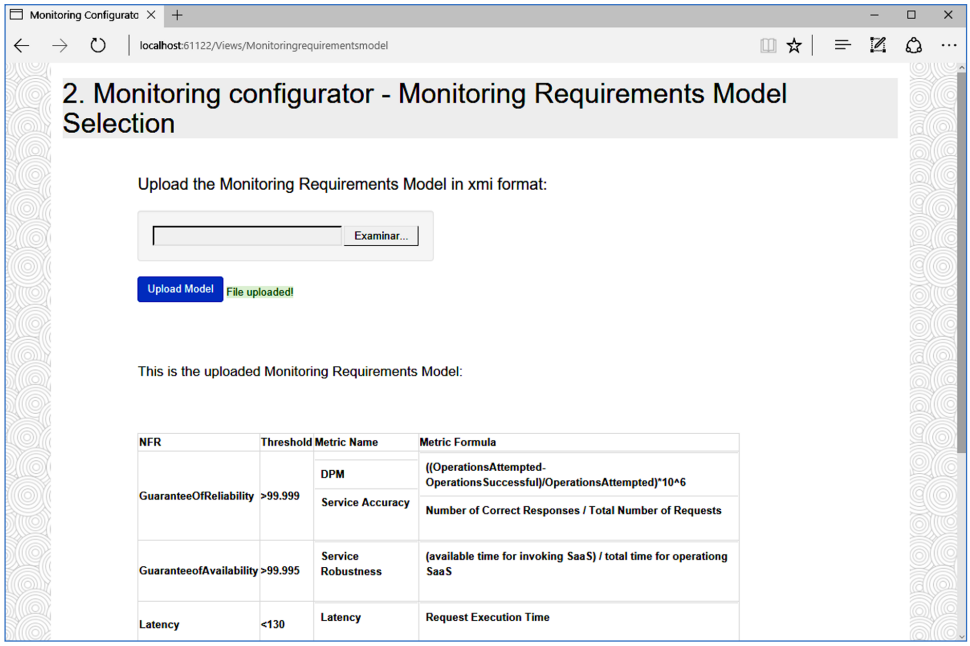


Figura 6-12. Pantalla de Selección del Modelo de Requisitos de Monitorización (Jimenez, 2015)

6.3.1.4. Clasificación de los requisitos no funcionales

La clasificación de los RNF deberá ser realizada con la finalidad de poder mapear estos requisitos con los parámetros de la plataforma. Esta tarea se verá soportada por un Modelo de Calidad SaaS, el mismo que ha sido descrito en el Capítulo 5. Para ello, el usuario debe seleccionar uno por uno los RNF, haciendo clic sobre una tabla situada bajo la explicación del paso 3.1 de la Figura 6-13 en la interface del prototipo aquí presentado. Aquí, se procederá a realizar la clasificación dentro de las ramificaciones del modelo de Calidad SaaS, tal y como indica la segunda tarea del proceso de monitorización presentado en este trabajo. Para ello, primero se selecciona la característica a la que pertenece la el RNF y que puede ser encontrado bajo el título Characteristic y recorriendo el árbol del Modelo de Calidad SaaS que ha sido construido conforme el SQuaRE (ISO/IEC, 2011).

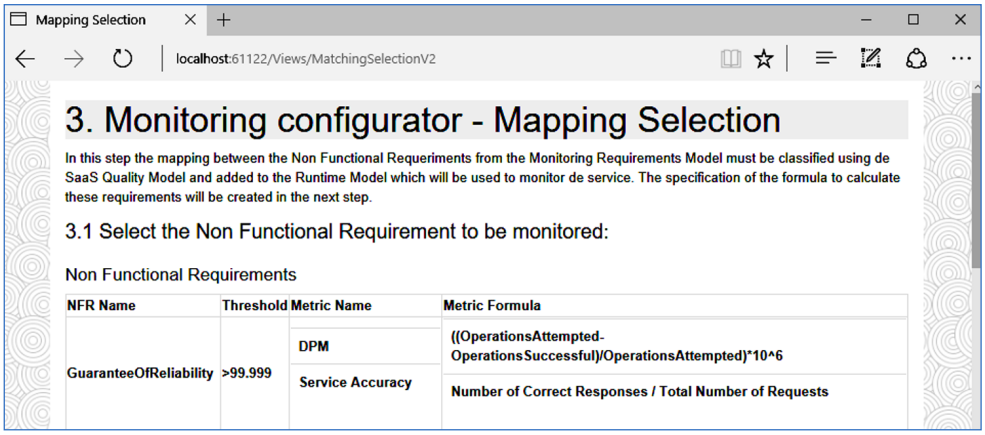


Figura 6-13. Mapeo de las Métricas en el Prototipo del Config. de la Monitorización (Jimenez, 2015)

Para el mapeo de los RNF y la construcción posterior del Modelo de Calidad en Tiempo de Ejecución, se utiliza la interface mostrada en la Figura 6-14. En ésta, el modelo de Calidad SaaS se ha cargado como parte de los datos de la aplicación de una manera dinámica al igual que el Modelo de Requisitos de Monitorización, todo esto desde su respectiva fuente XMI, por lo que podría ampliarse o modificarse sin que la implementación se vea afectada. Se ha utilizado un extracto de un Modelo de Calidad SaaS, y como parte de un trabajo paralelo, se está desarrollando un Modelo de Calidad SaaS más elaborado y detallado. Sin embargo y para ejemplificación, el modelo de Calidad SaaS aquí utilizado, contiene las características, sub-características, atributos, métricas y operacionalizaciones necesarias y suficientes para los casos descritos en esta instanciación.

Cabe destacar, que es necesario que el usuario que utilice esta solución, debe tener un conocimiento teórico sobre el modelo de calidad y en general en temas de Cloud Computing y servicios, de manera que éste le permita entender cómo se realizará la monitorización de los servicios. Y qué es lo que quiere monitorizar, siendo capaz de encontrar la clasificación más apegada a los RNF y utilizando todos los conceptos de calidad de software llevados al ámbito de servicios en la nube con modelo de servicio SaaS.

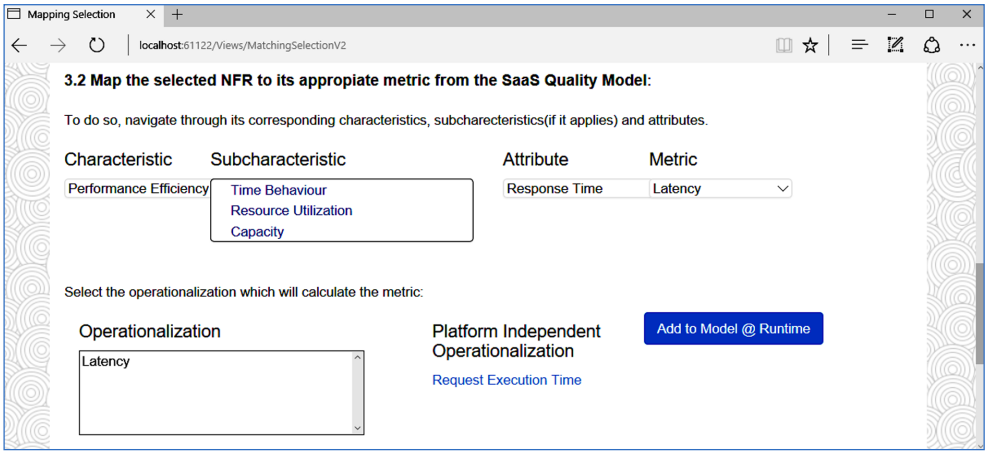


Figura 6-14. Asociación de Métricas. Mapeo Utilizando el Modelo de Calidad SaaS.

Una vez que se ha realizado la clasificación y selección de métricas, se debe seleccionar la operacionalización. Esta operacionalización describe la forma de medir la métrica de manera general, en términos independientes de la plataforma. Si la clasificación es correcta, se la añadirá al Modelo de Calidad en Tiempo de Ejecución. Este prototipo así también, permite la eliminación de las acciones antes realizadas en caso de haber algún error.

Finalmente, cuando se ha terminado ya la clasificación de todos los requisitos no funcionales, se procede al siguiente paso. En la tabla (ver Figura 6-15) situada en la sección del modelo en tiempo de ejecución, se podrá ver la siguiente información:

- **#NFR:** Número de requisito no funcional. Identifica al requisito no funcional del Modelo de Requisitos de Monitorización, de forma única.
- **Attribute Name:** Nombre del atributo bajo el cual se ha clasificado el RNF. (No es necesaria la información pertinente a las características y sub-características). No debe haber dos atributos con el mismo nombre en el modelo de calidad SaaS.
- **Metric Name**: Métrica producto de la clasificación del RNF.
- **Operationalization Name:** Nombre de la operacionalización, en términos independientes de la plataforma, que ha sido elegida para medir la métrica.

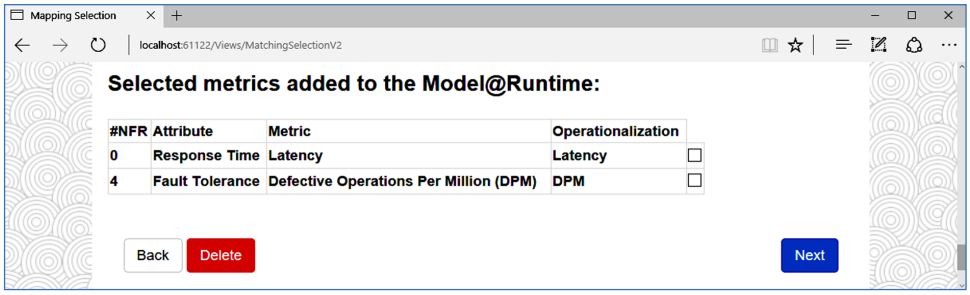


Figura 6-15. Asociación de Métricas. Mapeo para el Modelo de Calidad en Tiempo de Ejecución (Jimenez, 2015)

6.3.1.5. Constructor de funciones de medición

El constructor de funciones de medición soporta la asociación de la métrica con las fuentes de información, en este caso, los escenarios de recolección de datos. Una vez clasificada y seleccionada una operacionalización independiente de la plataforma, que "equivalga" a la presentada en el Modelo de Requisitos de Monitorización, el usuario debe mapearla a los datos que emanen de los servicios a monitorizarse.

Para el caso del presente prototipo, se utilizarán para los ejemplos descritos los *Performance Counters* de *Azure Dignostics* más apropiados, caso contrario los contadores personalizados que se obtienen ya sea de métricas indirectas, envoltorios o soluciones de terceros. Esta función de medición será la que utilice el motor de medición para extraer la información de los servicios y realizar las operaciones necesarias para medir los atributos de calidad.

De esta forma, el configurador da soporte a los escenarios de extracción antes mencionados.

Para realizar esta tarea, se debe seleccionar cada RNF de la tabla haciendo clic sobre el *Checkbox* de la fila del RNF deseado. Y luego pulsar el botón *Edit*. Una vez realizado esto se obtiene una pantalla para construir las funciones de medición necesarias, esta interface está mostrada en la Figura 6-16.

La pantalla servirá para construir la operacionalización (en caso de no existir) o seleccionarla de entre las existentes. Esto se realizará mediante la selección de los *Performance Counters* de *Microsoft Azure*, los términos de la calculadora (símbolos y números) y los *Custom Counters* (contadores personalizados provenientes ya sea de métricas indirectas, envoltorios o soluciones de terceros), que en este caso estarán disponibles para construir la función de medición que mapee la operacionalización independiente de la plataforma con la que es dependiente y será utilizada para los cálculos.

Para añadir un contador (ya sea un Performance Counter o un Custom Counter) se debe seleccionar la opción del menú haciendo clic en el título de manera que aparezca la vista con las opciones de los contadores. En ella se deberá elegir un contador de la lista haciendo clic sobre él y seleccionar un tipo de extracción (Extraction Type): En el prototipo puede escogerse entre media (se realizará una media de los valores obtenidos entre cálculos de la métrica) y total (se escogerá el último valor obtenido).

Seleccionar el ratio de extracción (Extraction Rate): Este es un valor numérico (en segundos) en el cuál se extrae un dato del contador seleccionado del servicio.

Pulsar Add To Formula: El contador aparecerá en el cuadro de texto de la función de medición y podrá elegirse otro ítem para continuar con la construcción de la función de medición siguiendo el mismo proceso (por cuestiones de visualización y espacio se ha recortado el marco de la ventana de lasFigura 6-16 y 6-17)

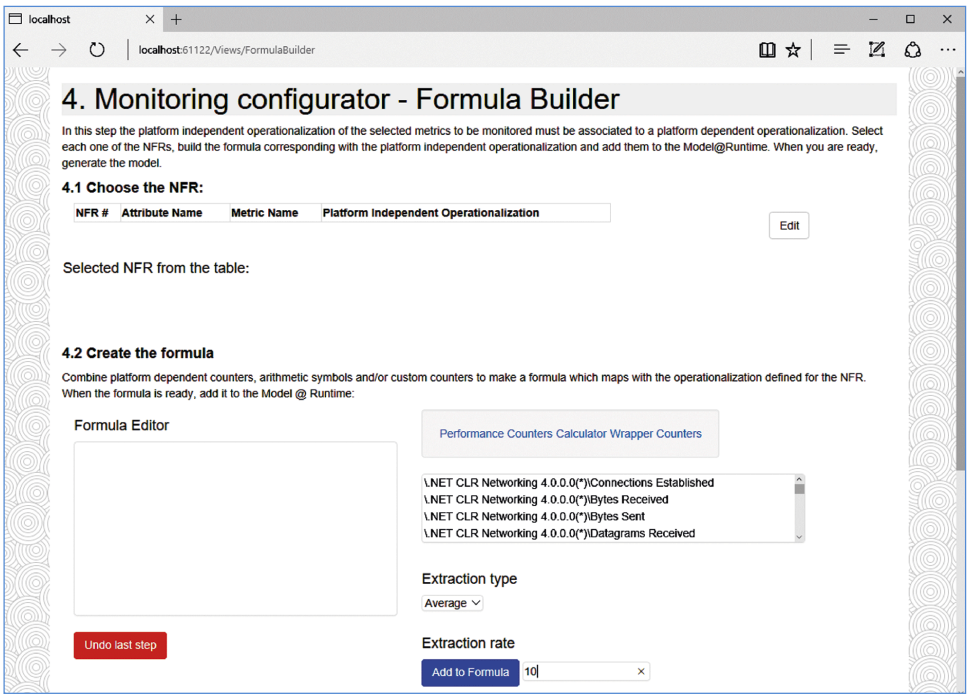


Figura 6-16. Instanciación del Constructor de Funciones de Medición (Jimenez, 2015)

En caso de producirse errores al introducir nuevos elementos en la formula, puede deshacerse el último elemento añadido pulsando el botón *Undo last step*. En caso de que la función de medición ya esté completada, puede añadirse al

Modelo en tiempo de ejecución pulsando *Add to Model@Runtime*. Al pulsar, una nueva fila aparecerá en la tabla localizada en el punto 4.3 de la Figura 6-17, con la siguiente información:

- **#NFR**: Número de requisito no funcional. Identifica al requisito no funcional de forma única.
- **Attribute Name**: Nombre del atributo en el cuál este RNF se ha clasificado.
- **Metric Name**: Nombre de la métrica en la cual se ha clasificado el RNF.
- **Platform Dependent Operationalization**: Función de medición construida en los pasos anteriores con elementos dependientes de la plataforma.

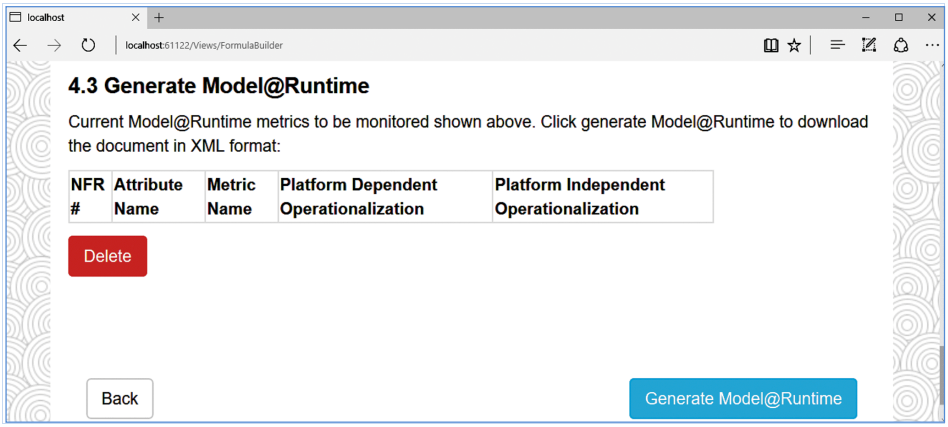


Figura 6-17. Constructor de Fórmulas. Agregación al Modelo de Calidad en Tiempo de Ejecución (Jimenez, 2015)

Este proceso debe repetirse con todos los RNF que se deseen monitorizar. Una vez que todas las operacionalizaciones dependientes de la plataforma se hayan mapeado y cargado en el Modelo de Calidad en Tiempo de Ejecución, éste se generará de forma definitiva y será enviado al Middleware de Monitorización y Análisis.

6.3.2. Implementación del middleware de configuración y análisis

El Middleware de Configuración y Análisis forma parte de la infraestructura de monitorización. Esta herramienta es la encargada de:

1. Recibir el modelo en tiempo de ejecución y configurar el servicio para su monitorización.
2. Extraer los datos de monitorización utilizando los diferentes escenarios de extracción de información descritos en la sección 5.4.

3. Realizar los cálculos descritos por las operacionalizaciones del as métricas contenidas en el modelo en tiempo de ejecución.
4. Almacenar los resultados de las métricas en un medio de almacenamiento permanente para su posterior tratamiento y consulta.

En esta sección se ha instanciado un prototipo del middleware de monitorización en la plataforma Microsoft Azure para la monitorización de servicios desplegados en esta misma plataforma y se ha contemplado los tres primeros escenarios de extracción de datos. El cuarto escenario se ha planteado, mas su instanciación será parte de trabajos futuro ya que se deben definir los conectores para la extracción de la información.

A continuación, se procederá a exponer los requisitos y características del middleware así como su implementación y operación, la misma que ha sido desarrollada y presentada en (Jimenez, 2015).

6.3.2.1. Requisitos y características del middleware de monitorización

Los requisitos de monitorización del middleware, van en consonancia a las brechas de investigación encontradas en las otras herramientas de este tipo, realzando aquellas ventajas o requisitos que aportará a la solución el uso de los modelos en tiempo de ejecución.

En el trabajo de Jimenez (2015), se ha tenido en cuenta un refinamiento iterativo del middleware, teniendo en cuenta los siguientes RNF:

- **Extensibilidad:** El middleware de monitorización y análisis debe ser capaz de adaptarse fácilmente a diferentes plataformas en cuanto a su diseño. Y además, debe ser fácilmente extensible a nuevos requisitos. Esto es conseguido a través del uso de los modelos en tiempo de ejecución.
- **Interoperabilidad:** El middleware de monitorización y análisis debe ser capaz de interactuar con otras soluciones de monitorización, lo cual es conseguido a través del tercer y cuarto escenario de recolección de datos.
- **Flexibilidad:** En cuanto a su operación, debe ser posible además, adaptar de forma fácil y dinámica nuevos RNF ya sea modificando los existentes o agregando nuevos RNF. Así como también cambiando de manera sencilla los parámetros de monitorización como por ejemplo el tiempo de recolección de información, tipo de los RNF entre otros.

Además, el prototipo aquí presentado, ofrece los siguientes requisitos funcionales:

- **Extracción de Datos:** El monitor debe ser capaz de extraer los datos correspondientes a los escenarios de una manera efectiva y eficiente.
- **Cálculo de Métricas:** El monitor debe ser capaz de calcular las funciones de medición expresadas a través las operacionalizaciones seleccionadas en el modelo en tiempo de ejecución.
- **Almacenamiento de Medidas:** El monitor debe ser capaz de almacenar las medidas calculadas en una base de datos, de manera que la información de monitorización pueda ser utilizada posteriormente. Esta información puede ser utilizada para calcular otras métricas, para funciones de agregación usadas en reportes o para la generación de reportes en general.
- **Actualización en Tiempo de Ejecución:** El monitor debe ser capaz de actualizar los parámetros de su configuración en cualquier momento de su ejecución, recibiendo un nuevo modelo en tiempo de ejecución que ajuste los valores a nuevos requisitos de monitorización.

6.3.2.2. Arquitectura

El prototipo de ésta solución desarrollado en el trabajo de Jimenez (2015) presenta la arquitectura mostrada en la Figura 6-18.

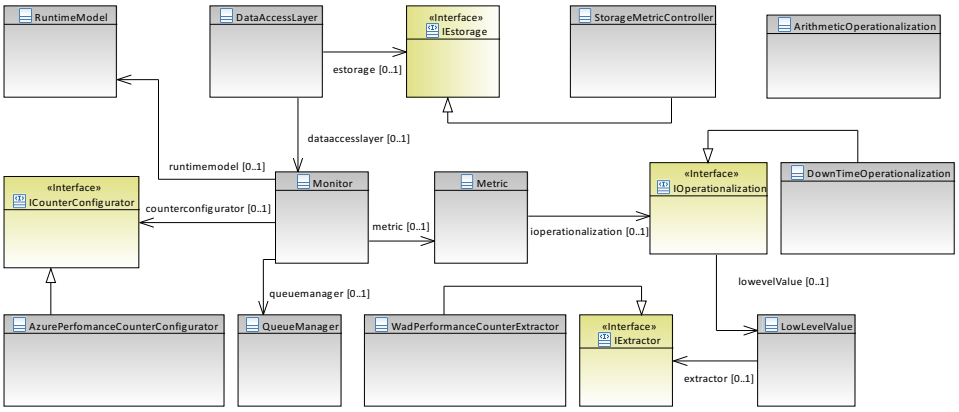


Figura 6-18. Constructor de Fórmulas. Agregación al Modelo de Calidad en Tiempo de Ejecución (Jimenez, 2015)

En cuanto a la descripción de los componentes aquí presentados, se tomará la explicación presentada por Jimenez (2015); de cada una de las clases que intervienen en el mismo.

Monitor: Es la clase encargada del ciclo de vida del middleware. Se ocupa de recoger los datos desde los servicios, realizar las operaciones para determinar el valor de las métricas, guardar los datos y realizar las actualizaciones cuando se oportuno. Esta clase es la central del middleware.

Metric: Representación de una métrica medible a través de una operacionalización. Una vez que hemos conseguido el valor de ésta, representará la salida del proceso de medición, la misma que contiene el valor de la medida en cada ciclo de recogida de datos.

IOperationalization: Interfaz que define los métodos para el cálculo de las operacionalizaciones. Es necesario realizar una especialización entre operacionalizaciones ya que no todas tendrán un carácter aritmético (aunque si serán las más generales) y es probable que otras requieran de cálculos más complejos. Por ejemplo en el caso del *Downtime* (tiempo de servicio inaccesible), no existe una métrica válida expresada por el software Diagnostics y para su cálculo es necesario el uso de condicionales y guardar valores históricos para realizar comparaciones y funciones de agregación. Para facilitar el acople de este tipo de implementaciones se ofrece esta interfaz.

Arithmetic Operationalization: Constituye la implementación de la interfaz IOperationalization. Esta clase es la encargada de realizar cualquier cálculo de métricas cuya operacionalización pueda ser expresada en una fórmula aritmética también llamada función de medición. Esta implementación ayudada por la librería NCalc de .NET, es la encargada realizar los cálculos definidos por las fórmulas de la operacionalización y de encargar al Extractor la recolección de los valores necesarios para dicho cálculo. El módulo NCalc permite el uso de fórmulas que en un formato propio de la librería, permitan calcular dinámicamente, cualquier tipo de función de medición con variables no asignadas estáticamente, por lo que añade flexibilidad a la hora de crear y aplicar la función de medición del modelo en tiempo de ejecución.

DowntimeOperationalization: Esta clase representa la implementación de la interfaz IOperationalization, sirve para realizar una medida personalizada como las que se mencionaban en la descripción de la interfaz. Tomando como ejemplo la operacionalización del atributo de disponibilidad no podía realizarse de forma aritmética debido al tipo de datos sobre el estado del servicio disponible en Azure Diagnostics. Por lo que para este caso ha sido necesario realizar una labor de detección de intervalos en los cuáles el servicio está caído (detectado por el reinicio del contador de Uptime) y a partir de ahí calcular el tiempo aproximado en el cuál el servicio ha estado caído (la diferencia entre la marca de tiempo del

último contador de Uptime mayor que cero y el primer contador de Uptime vuelto a recibir).

LowLevelValue: Representación de los datos de bajo nivel obtenidos por los medios de recolección de datos de los servicios. Cada uno de estos valores son empleados por las funciones de medición de las operacionalizaciones para llevar a cabo los cálculos. Lleva asociada una fuente de datos del cual son extraídos periódicamente.

IExtractor : Interfaz que define los métodos de recolección de datos procedentes de los extractores de datos crudos de monitorización. Esta interfaz posibilita el uso de distintos extractores de datos para datos procedentes de distintas fuentes. De esta manera distintos tipos de extractores pueden trabajar en conjunción con distintos tipos de datos, que pueden utilizarse durante el mismo proceso de monitorización o incluso en una misma métrica formada por datos procedentes de distintas fuentes.

IExtractor: Interfaz que define el comportamiento de un extractor de datos de monitorización de una fuente de datos. Utilizado para desacoplar de la implementación el uso de uno o varios adaptadores. Establece dos tipos de muestreo de datos, como totales o como media. Esto es así dado que el elevado número de muestras de datos que puede existir en la fuente no es necesario para obtener medidas fieles ya que si se ha definido un tiempo de recolección de estas métricas muy reducido en los contadores acumulativos obtendremos muchos valores casi idénticos por lo que es recomendable solamente trabajar con totales o también este elevado número de muestras carece sentido a tan baja granularidad y es preferible trabajar con medias en intervalos cortos de tiempo (3-30s) que aportarán mayor significado y relevancia a la métrica.

WADPerformanceCounterExtractor: Implementación de IExtractor. Esta implementación toma como fuente de datos los contadores procedentes de Microsoft Azure Diagnostics localizados en la tabla WADPerfomanceCounters del servicio a monitorizar. Para su acceso es necesaria la obtención de la ConnectionString del servicio. Esta clase regula la obtención de datos, en caso de que se pidieran datos al extractor con mayor frecuencia de la que Diagnostics obtiene valores nuevos, este retornaría un valor vacío de manera que el resto del programa ignoraría esta medición hasta el siguiente ciclo de obtención de datos donde volvería comprobarse si existen valores nuevos y omitiendo cuando no existan procediendo de esta manera hasta obtener nuevas mediciones.

DataAccessLayer: Esta clase es la encargada de centralizar el acceso a la capa de datos de guardado de las métricas calculadas, de manera que se desacople la

fuente de datos utilizada para almacenar estos valores de la implementación y esta sea intercambiable dinámicamente.

IStorage: IStorage es la interfaz que define los posibles métodos de acceso a la base de datos de guardado de las métricas calculadas.

StorageMetricsController: Implementación de IStorage en la cual se implementa un adaptador para el almacenamiento de Azure, en el que se guardan las métricas calculadas.

QueueManager: QueueManager proporciona acceso a la API del sistema de colas de Microsoft Azure mediante el cual se transportan los modelos en tiempo de ejecución que determinarán la configuración de la monitorización.

ICounterConfigurator: Interfaz encargada de modelar los métodos disponibles para la realización de la configuración dinámica de contadores de rendimiento de la plataforma a monitorizar, en el caso en que estos estén disponibles de manera programática (como lo es en el caso de Windows Azure). En esencia se podrán configurar los tipos de contadores de datos crudos que se quieren recuperar y el período de extracción en el cual se desea recuperarlos.

AzurePerformanceCounterConfigurator: Instancia de ICounterConfigurator, encargada de la configuración de los Performance Counters disponibles en la plataforma Microsoft Azure. Ésta se encargará de activar y desactivar la extracción de datos según las necesidades marcadas por los modelos en tiempo de ejecución recibidos así como determinar el período de extracción de estos datos.

6.3.2.3. Descripción del funcionamiento

Para describir el funcionamiento de este middleware, se ha tomado así mismo el aporte en la instanciación de la infraestructura, propuesto por Jimenez (2015).

En el middleware de monitorización se encuentra desplegado y ejecutándose en una instancia *Worker Role* en la plataforma de Microsoft Azure. Una vez que éste se ejecuta, se crea un objeto *Monitor* el cual carga las configuraciones del Modelo de Calidad en Tiempo de Ejecución para poder iniciar el proceso de monitorización. Todo esto cuando el Configurador de la Monitorización ha generado el Modelo de Calidad en Tiempo de Ejecución, y lo ha enviado a través del sistema de colas de Microsoft Azure para que pueda ser recuperado por la clase *Queue Manager*.

Con el modelo en tiempo de ejecución en el monitor, se instancia un objeto *RuntimeModel* con el cual se accede a todos los datos de la configuración. Las operacionalizaciones de las métricas indicarán qué *Performance Counters* son

necesarios para poder realizar las mediciones. De manera que el monitor procederá a utilizar la instancia apropiada de la interface *ICounterConfigurator,* que en este caso es Microsoft Azure, por lo que se empleará *AzurePerformanceCounter Configurator* para realizar la activación en el servicio correspondiente.

El Middleware de Monitorización procederá a realizar los cálculos de las métricas presentes en el modelo en tiempo de ejecución. Para ello emplea la estructura de la función de medición mostrada en la Figura 6-19, la misma que es compatible con NCalc. Así cualquier fórmula aritmética descrita puede ser calculada dinámicamente. Primero se dan valores a los parámetros de la formula utilizando la instancia del *IExtractor* apropiada, dependiendo del escenario de captura de información, en el caso de ser un contador de Azure se utiliza el *WADPerformanceCounterExtractor*, el cual busca los valores recientes para presentar ese valor. Una vez que se han calculado los valores, NCalc calcula el valor de las métricas, el cual será almacenado en la base de datos destinada a tal efecto por la instancia *IStorage* que se haya seleccionado en la configuración, en este caso se trata de la tabla *Calculated Metrics*, que se creó en el Almacenamiento de Azure.

Este proceso se repetirá de manera cíclica hasta que el monitor reciba una actualización del modelo en tiempo de ejecución, momento en el cual reconfigurará el servicio y procederá a realizar de nuevo el bucle del proceso.

File Edit Selection Find View Goto Tools Project Preferences Help

FormulaOperationalization.html

```
<name>Defective Operations Per Million (DPM) </name>
<operationalization xsi:type="IndirectMetric">
    <name>DPM</name>
    <function>
        <formula>([36]-[35])/[36]</formula>
        <operands>
            <DirectMetric>
                <name>\ASP.NET Applications(*)\Requests Total </name>
                <identifier>[36]</identifier>
                <extractionRate>4</extractionRate>
                <extractionType>0</extractionType>
            </DirectMetric>
            <DirectMetric>
                <name>\ASP.NET Applications(*)\RequestsSucceeded </name>
                <identifier>[35]</identifier>
                <extractionRate>4</extractionRate>
                <extractionType>0</extractionType>
            </DirectMetric>
            <DirectMetric>
                <name>\ASP.NET Applications(*)\Requests Total </name>
                <identifier>[36]</identifier>
                <extractionRate>4</extractionRate>
                <extractionType>0</extractionType>
            </DirectMetric>
        </operands>
    </function>_
</operationalization>
```

Figura 6-19. Operacionalización de la función de medición representada usando XML (Jimenez, 2015) .

La ventaja de utilizar este mecanismo es que así como ahora se realiza una actualización desde el nuevo requisito hacia la infraestructura, como trabajo futuro se puede implementar una fácil reconfiguración de los parámetros de monitorización, basados por ejemplo, en la carga que experimente la nube o en características propias que se necesiten modificar desde los servicios hacia la infraestructura de monitorización, para sacar el mayor provecho posible del uso de la tecnología de los modelos en tiempo de ejecución.

6.3.3. Aplicación de la instanciación del método al caso de estudio CoCoMe

En esta sección se presentará el uso de la infraestructura con el caso presentado en la sección 6.2. El contexto en este caso ha sido ya introducido y explicado y queda por presentar el uso de la infraestructura construida sobre el caso de estudio.

Para la aplicación de la infraestructura se ha tomado uno de los RNF presentados anteriormente y se ha agregado por cuestiones de ilustración el cálculo de la confiabilidad para mostrar un escenario diferente de recolección de datos, teniendo en cuenta que éste ejemplo también ha sido utilizado en el trabajo de Jimenez (2015).

Por tanto, los requisitos no funcionales a ser monitorizados serán:

1. La eficiencia (*Eficiency*) será medida a través de la latencia, para ello se usará la métrica del tiempo de ejecución de la petición y se ha establecido que será de máximo 130 milisegundos y será calculada con la siguiente función de medición:

$$Execution\ Time = Request\ Execution\ Time$$

2. La confiabilidad (*Reliability*) será medida a través de la operaciones defectuosas por millón. En este caso, el servicio tendrá un máximo de 10 operaciones defectuosas por millón (99.999% de confiabilidad de servicio). Este requisito será calculado utilizando la métrica correspondiente (*Defective Operations per Million-DPM*)

$$DPM = \frac{Operations\ Attempted\ - OperationsSuccessful}{Operations\ Attempted} * 10^6$$

6.3.3.1. Introducción de los datos

En el primer paso de la interfaz, *Platform Selection* mostrada en la Figura 6-20, dentro del paso 1.1, se selecciona la plataforma en la cual se encuentra desplegado el servicio a ser monitorizado, para este caso concreto Microsoft

Azure. Una vez seleccionada aparecen dos campos más que permiten la configuración de la plataforma Azure con respecto al servicio.

Dentro del paso 1.2 se introduce el nombre del servicio, en nuestro caso el servicio es la aplicación web CoCoMe. En el caso del *Deployment ID* y el *Connection String* estos son parámetros propios del servicio que deben ser consultados en el panel de control del servicio CoCoMe, el primero es un identificador del servicio y el segundo una clave pública que concede acceso a los datos del servicio, necesarios para llevar a cabo la monitorización. La configuración finalizada puede observarse en la Figura 6-20.

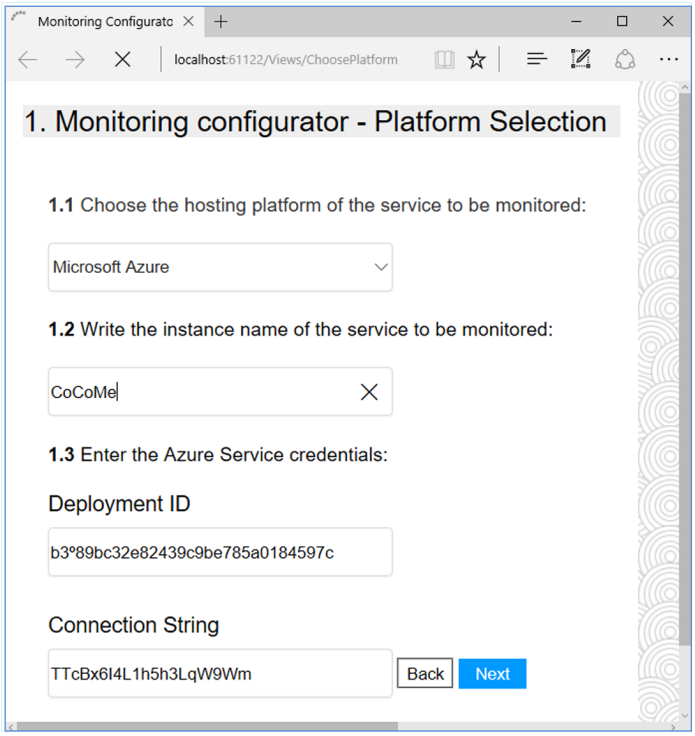


Figura 6-20. Caso de Estudio: Selección de la Plataforma y Configuración (Jimenez, 2015)

6.3.3.2. Selección del Modelo de Requisitos de Monitorización

En la segunda pantalla, tras pulsar el botón Next, encontramos la interfaz que permite la carga del Modelo de Requisitos de Monitorización, cuya construcción fue presentada con detalle en la sección 6.2.1.1. Este archivo debe estar en formato XMI, como se ha indicado. La Figura 6-21 muestra la interfaz una vez que ha realizado la carga de los RNF contenidos en el Modelo de Requisitos de Monitorización.

Cuando el Modelo de Requisitos de Monitorización ha sido correctamente cargado, se habilita el botón Next que permitirá llegar al siguiente paso.

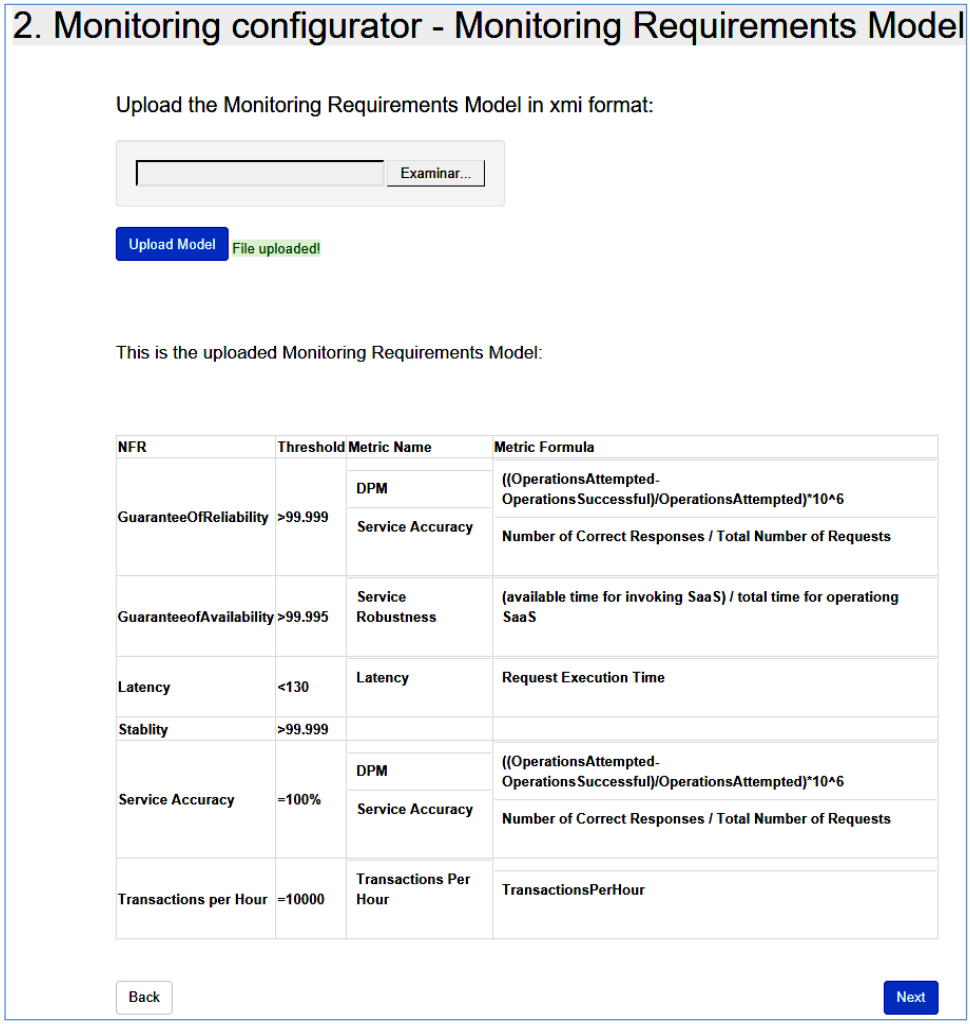


Figura 6-21. Caso de Estudio: Carga del Modelo de Requisitos de Monitorización (Jimenez, 2015)

6.3.3.3. Clasificación y asociación de métricas

Una vez cargado el Modelo de Requisitos de Monitorización, se deben asociar los RNF haciendo uso del Modelo de Calidad SaaS. Para este ejemplo se tomará la *Latencia* y la *Confiabilidad*.

Mapeo de la Confiabilidad: Para realizar este paso, se debe seleccionar de la tabla presentada en la interfaz, el RNF correspondiente a la confiabilidad y luego realizar la asociación, en este caso la correspondiente característica será

"Reliability", acto seguido y según el Modelo de Requisitos de Monitorización se debe seleccionar el atributo *fault tolerance* (tolerancia a fallos) y la operacionalización correspondiente que en este caso sería DPM. En la Figura 6-22 se presenta gráficamente lo antes mencionado y la manera en la cual en la instanciación de la infraestructura, puntualmente del Configurador de la Monitorización se realiza este paso.

Mapeo de la Latencia: De igual manera se procede con el RNF correspondiente a la Latencia. En este caso tendremos como característica en el Modelo de Calidad SaaS a la Eficiencia de Funcionamiento *(Performance Efficiency)*, la sub-característica correspondiente será el Comportamiento Temporal (*Time Behaviour)*, el atributo Tiempo de Respuesta (*Response Time)*, la métrica *Latency* y finalmente la operacionalización con el mismo nombre (*Latency)*. La Figura 6-23 muestra la clasificación y mapeo de este RNF.

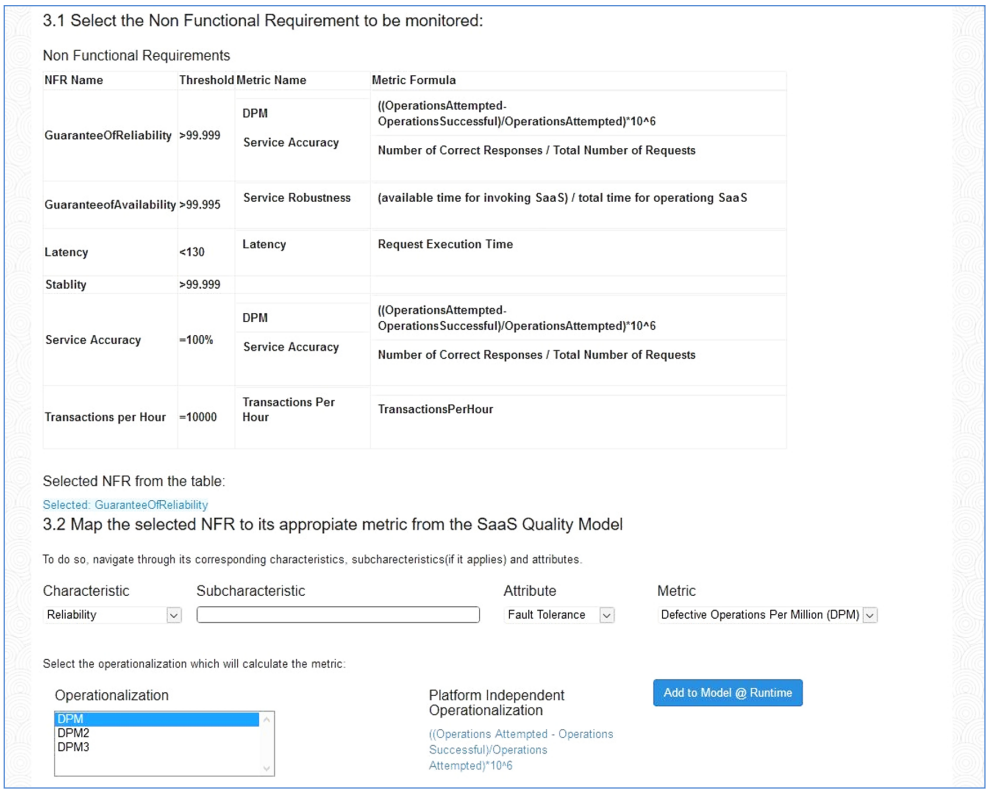


Figura 6-22. Caso de Estudio: Mapeo correspondiente a la *Confiabilidad* (Jimenez, 2015)

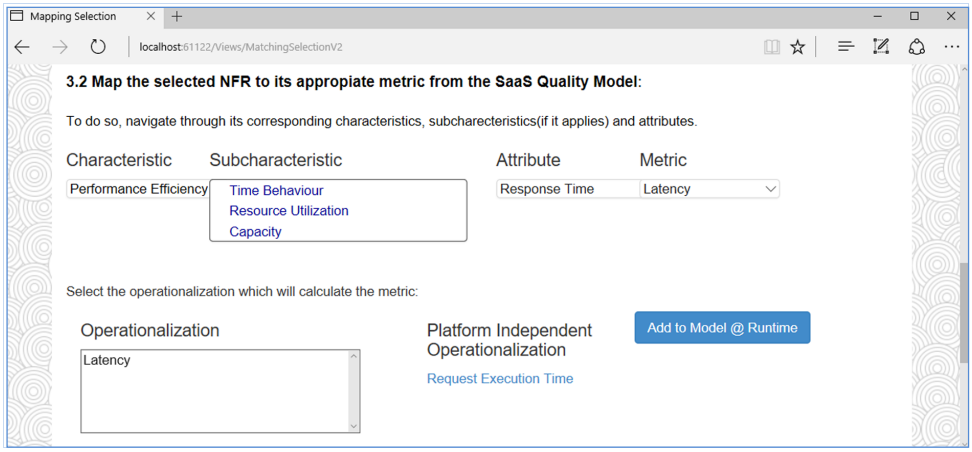


Figura 6-23. Caso de Estudio: Mapeo correspondiente a la *Latencia* (Jimenez, 2015)

Una vez clasificados y correctamente mapeados los RNF y las métricas procedentes del Modelo de Calidad SaaS (ver Figura 6-24), los mismos que hasta el momento son independientes de la plataforma, se deberá realizar el mapeo con las fuentes de datos crudos a fin de poder calcular las funciones de medición. Para ello se debe seleccionar uno por uno los atributos, métricas y operacionalizaciones a fin de realizar dicho mapeo.

Figura 6-24. Caso de Estudio: Métricas añadidas al Modelo de Calidad en Tiempo de Ejecución(Jimenez, 2015)

6.3.3.4. Construcción de las métricas dependientes de la plataforma

Una vez mapeadas las métricas independientes de la plataforma, éstas deben transformarse en métricas dependientes de la plataforma para ser incluidas en el Modelo de Calidad en Tiempo de Ejecución. Para ello, con la guía de las operacionalizaciones SaaS, se deben construir las funciones de medición de cada RNF.

Para el primer caso, se ve en la Figura 6-25 el mapeo para la confiabilidad, el mismo que pertenece al segundo escenario de extracción de datos, en donde la métrica es calculada indirectamente a partir de una función de medición en la que intervienen varios *Performance Counters* de *Microsoft Azure*.

Una vez añadida esta fórmula, se agrega al Modelo de Calidad en Tiempo de Ejecución.

Para el segundo caso, se procede a seleccionar la latencia, ésta tiene una correspondencia directa con los parámetros propios de la plataforma, en este caso corresponde al primer escenario de recolección de datos. Por tanto, se procede a seleccionar el Performance Counter correspondiente y se agrega al Modelo de Calidad en Tiempo de Ejecución. En la interfaz este paso está ilustrado en la Figura 6-26.

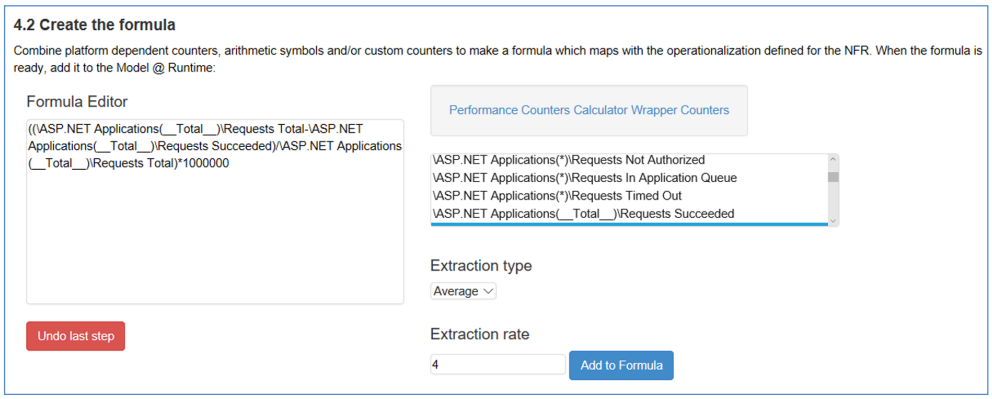


Figura 6-25. Caso de Estudio: Construcción de la Función de Medición Dependiente de la Plataforma para la Confiabilidad (Jimenez, 2015)

Aquí se debe considerar una tasa de extracción de cada 3 minutos, como se ha mencionado anteriormente, en trabajos futuros se tiene planificado hacer que la tasa de extracción de datos sea calculada en base a ciertos aspectos de la plataforma (carga, cuellos de botella, horas de menor uso, almacenamiento). En este caso los modelos en tiempo de ejecución entrarían a jugar un rol muy importante en cuanto a la conexión causal que es uno de los principales fines que persigue esta tecnología, en sentido inverso (desde el sistema hacia la infraestructura de monitorización). Haciendo que la solución reaccione de acuerdo al estado de la provisión del servicio.

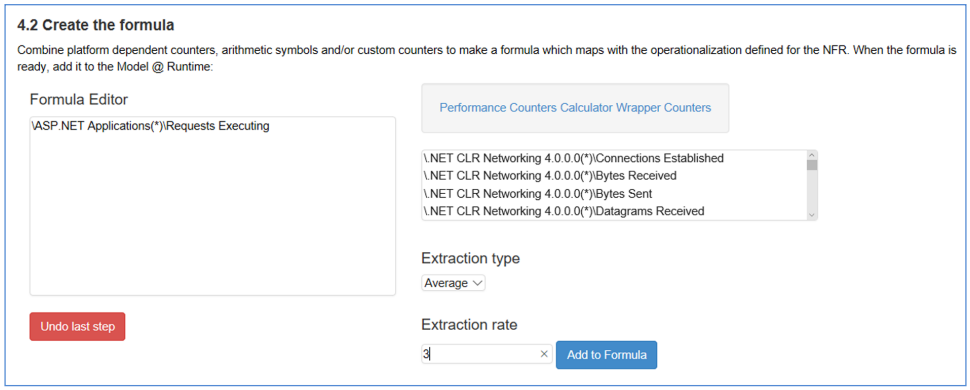


Figura 6-26. Caso de Estudio: Construcción de la Operacionalización de la Latencia(Jimenez, 2015)

Una vez que se configuran todas las funciones de medición y sus fuentes de información, enlazadas por medio de las operacionalizaciones de las métricas, se procede a la generación del Modelo de Calidad en Tiempo de Ejecución, para poder enviarlo al middleware de monitorización, quien continuará con el proceso de monitorización deseado. La Figura 6-27 muestra la última pantalla de la Configuración de la Monitorización, previa la generación del Modelo de Calidad en Tiempo de Ejecución.

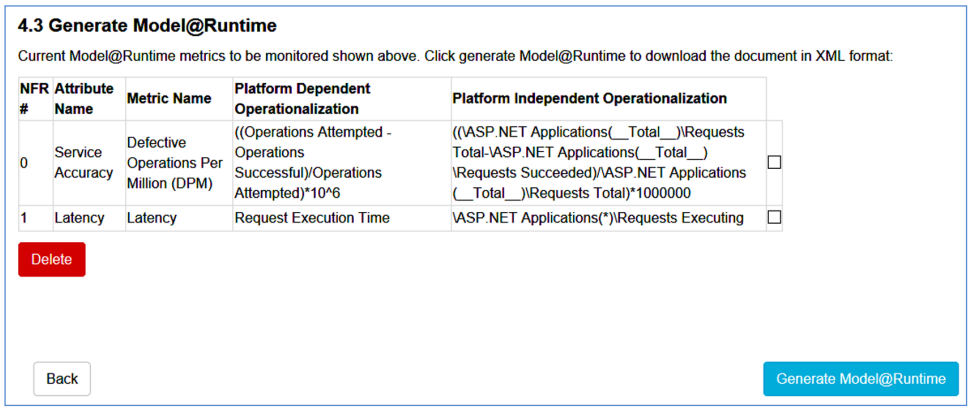


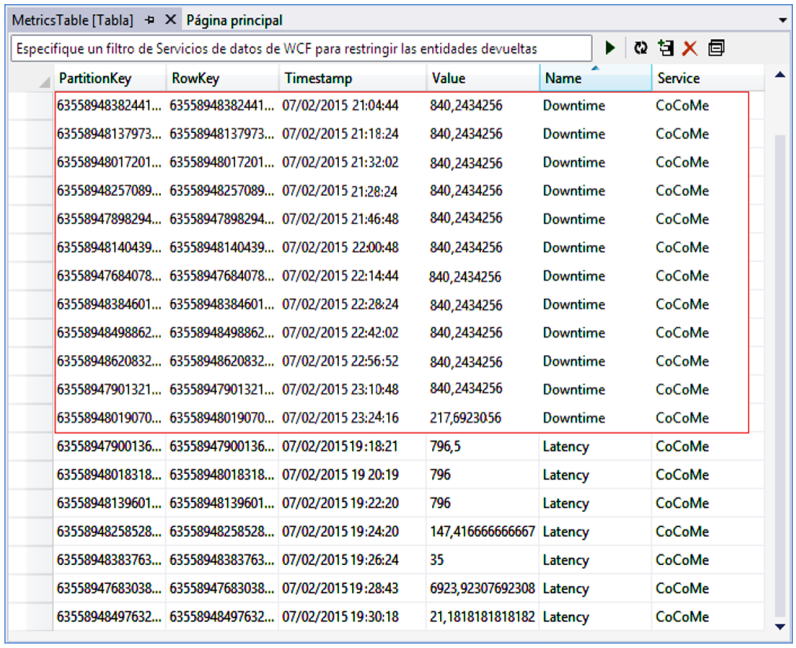
Figura 6-27. Caso de Estudio: Resultado de la Configuración (Jimenez, 2015)

En el Anexo II se muestra el resultado de la generación del Modelo de Calidad en Tiempo de Ejecución, el mismo que ha sido obtenido haciendo uso de la instanciación presentada en el trabajo de Jimenez (2015).

6.3.3.5. Utilización de middleware de monitorización y análisis en el caso de estudio.

Una vez que el Modelo de Calidad en Tiempo de Ejecución ha sido generado, éste será recibido por la instancia del Middleware que esté ejecutándose. Se analizará este modelo en XML y se convertirá la información recibida en instancias cuya información se utilizará para realizar la configuración. Se tomarán las métricas empleadas y usando las fórmulas se procederá a activar en el servicio la recogida de información de los Contadores de Rendimiento seleccionados, en este caso *Requests Succeeded, Requests Total, Request Execution Time* y *Requests Response Time*.

Cuando los datos se recojan, se procederá a aplicar las fórmulas y poblar la base de datos especificada con los valores de los atributos correspondientes. O a generar los valores y las visualizaciones, dependiendo de la configuración. Revisando las medidas de la Latencia y la Confiabilidad pueden observarse si se cumplen o no los requisitos ofrecidos en el SLA. La Figura 6-28 muestra los resultados de la monitorización obtenidos y contenidos en la base de datos mencionada.



PartitionKey	RowKey	Timestamp	Value	Name	Service
63558948382441...	63558948382441...	07/02/2015 21:04:44	840,2434256	Downtime	CoCoMe
63558948137973...	63558948137973...	07/02/2015 21:18:24	840,2434256	Downtime	CoCoMe
63558948017201...	63558948017201...	07/02/2015 21:32:02	840,2434256	Downtime	CoCoMe
63558948257089...	63558948257089...	07/02/2015 21:28:24	840,2434256	Downtime	CoCoMe
63558947898294...	63558947898294...	07/02/2015 21:46:48	840,2434256	Downtime	CoCoMe
63558948140439...	63558948140439...	07/02/2015 22:00:48	840,2434256	Downtime	CoCoMe
63558947684078...	63558947684078...	07/02/2015 22:14:44	840,2434256	Downtime	CoCoMe
63558948384601...	63558948384601...	07/02/2015 22:28:24	840,2434256	Downtime	CoCoMe
63558948498862...	63558948498862...	07/02/2015 22:42:02	840,2434256	Downtime	CoCoMe
63558948620832...	63558948620832...	07/02/2015 22:56:52	840,2434256	Downtime	CoCoMe
63558947901321...	63558947901321...	07/02/2015 23:10:48	840,2434256	Downtime	CoCoMe
63558948019070...	63558948019070...	07/02/2015 23:24:16	217,6923056	Downtime	CoCoMe
63558947900136...	63558947900136...	07/02/2015 19:18:21	796,5	Latency	CoCoMe
63558948018318...	63558948018318...	07/02/2015 19 20:19	796	Latency	CoCoMe
63558948139601...	63558948139601...	07/02/2015 19:22:20	796	Latency	CoCoMe
63558948258528...	63558948258528...	07/02/2015 19:24:20	147,416666666667	Latency	CoCoMe
63558948383763...	63558948383763...	07/02/2015 19:26:24	35	Latency	CoCoMe
63558947683038...	63558947683038...	07/02/2015 19:28:43	6923,92307692308	Latency	CoCoMe
63558948497632...	63558948497632...	07/02/2015 19:30:18	21,1818181818182	Latency	CoCoMe

Figura 6-28. Caso de Estudio: Resultados de la Monitorización (Jimenez, 2015)

El Middleware continuará recogiendo datos y calculando las métricas indicadas por este modelo hasta que un nuevo Modelo de Calidad en Tiempo de Ejecución sea recibido y se actualicen los parámetros.

En este caso de estudio se han seleccionado los dos primeros escenarios de recolección de datos, ya que estos ejemplifican la parte sustancial de la obtención de los datos desde los servicios y la necesidad de contar con las fuentes de datos. El tercer y cuarto escenario, cambiarán únicamente en la manera en la que se emana la información del servicio, en el tercer escenario por ejemplo se obtendrá el dato directamente desde el servicio a través de una programación adicional que permita obtener datos deseables que de otra manera serían imposibles obtener, por ejemplo en el caso de necesitar el número de clics sobre un botón para ciertas métricas se implementará un contador dentro del servicio, y para el cuarto escenario se tendrán fuentes emanadas por otras herramientas, las mismas que por medio de conectores permitirán extraer los datos hacia nuestra herramienta, una vez extraídos la operación tanto del tercer como del cuarto escenario serán las mismas que las presentadas en el ejemplo anterior.

6.4. Lecciones aprendidas de la instanciación de Cloud MoS@RT en Azure

Las lecciones aprendidas tras ejecutar las acciones presentadas en este capítulo indican que la solución aquí planteada es factible, fácil de instanciar en cualquier plataforma y que permite vislumbrar una serie de mejoras que pueden ser implementadas como parte de trabajo futuro. Sin embargo, durante la implementación del trabajo, se han encontrado una serie de dificultades, las mismas que según Jimenez (2015) y la autora de esta tesis, quienes han venido trabajando conjuntamente a fin de conseguir esta solución, se han ido venciendo.

Una de las fortalezas más grandes y quizás la más importante de este trabajo es el uso de los modelos en tiempo de ejecución, ya que este provee un alto grado de flexibilidad a la solución, así como permite tener escalabilidad y debido a las características propias de los modelos en tiempo de ejecución, se abre la posibilidad de reaccionar no solamente a nuevos requisitos sino también a entender y mejorar el proceso de monitorización teniendo en cuenta el estado de los servicios en determinados momentos de su ejecución.

Dentro de las dificultades está el arduo trabajo de quien realizó la implementación de la solución al momento de consumir los modelos en formato XMI. Sin embargo, la ventaja es que una vez conseguido esto, la solución puede constituir un API o una librería para futuras instanciaciones de esta solución. En cuanto a cuestiones específicas del desarrollo y sus dificultades, se puede

encontrar una mayor explicación a las cuestiones programáticas en Jimenez (2015).

Otra de las limitaciones ha sido la curva de aprendizaje y entendimiento de la plataforma y su manera de operar. Sin embargo se ha contado con acceso a la Plataforma Azure, gracias a un premio conseguido por el grupo de investigación, lo que ha motivado al uso de ésta plataforma al momento de la instanciación. Siendo una de las plataformas más utilizadas y sofisticadas del mercado, lo que ha hecho que se cuente con una tecnología precisa y actual para poder desarrollar la solución de forma óptima.

6.5. Conclusiones

Como conclusiones se tiene que la instanciación de Cloud MoS@RT, salvo ciertas dificultades fácilmente vencidas y muy dependientes de la plataforma mas no de la solución, ha sido factible y satisfactoriamente realizada en el trabajo de Jimenez (2015), lo que indica que el uso de Modelos en Tiempo de Ejecución realmente posibilita la fácil y dinámica configuración de nuevos RNF, así como su modificación. Además, abre una puerta hacia la posibilidad de ajustar este tipo de soluciones a posibles estados de los servicios para así mejorar el desempeño y no ocasionar cargas innecesarias al momento de la monitorización. Esto se prevé como uno de los trabajos futuros más inmediatos dado su nivel de innovación.

Por otro lado, la selección de la plataforma ha demostrado que los escenarios de extracción de información cubren todas las posibilidades de interactuar con los servicios, otras plataformas e incluso otras soluciones de monitorización, lo que hace que la solución sea versátil, fácilmente adaptable e interoperable.

Capítulo 7
Evaluación Empírica de Cloud MoS@RT

Este capítulo describe una familia de cuatro cuasi-experimentos que tiene como objetivo evaluar la utilidad de Cloud MoS@RT, teniendo en cuenta la percepción del usuario al momento de realizar la configuración de la monitorización de los servicios. En la sección 7.1 se presenta una introducción a la evaluación empírica de la solución propuesta, en la sección 7.2 se muestran los estudios empíricos empleados en el dominio de este trabajo. En la sección 7.3 se muestran los modelos teóricos de evaluación en Ingeniería del Software. La sección 7.4 presenta la adaptación del modelo de evaluación a ser utilizado en el método de monitorización. La sección 7.5 muestra una familia de cuasi-experimentos de la solución propuesta en esta tesis. La sección 7.6 presenta las amenazas a la validez y finalmente sección 7.7 presenta las conclusiones de este capítulo.

7.1. Introducción

Teniendo en cuenta el alto impacto que las percepciones de los usuarios tienen en la selección y adopción de una nueva solución, la aplicación del *Method Evaluation Model (MEM)* (Moody, 2001) puede ser de utilidad a la hora de evaluar la eficacia percibida de Cloud MoS@RT. El MEM fue originalmente propuesto como un medio para evaluar métodos de diseño de Sistemas de Información, pero sin embargo éste ha sido exitosamente aplicado en la evaluación de otro tipo de métodos (Abrahão *et al.*, 2011). Este modelo ha sido elegido debido a que integra variables del rendimiento actual de los usuarios y de su percepción como mecanismo para predecir la intención de uso y posible adopción en práctica de un método. MEM extiende el *Technology Acceptance Model (TAM)* (Davis, 1986), el cual ha sido validado empíricamente en numerosos estudios que demuestran su utilidad para analizar la facilidad de uso percibida, utilidad percibida e intención de uso de los participantes aplicando un método para predecir su posible aceptación.

Para utilizar el MEM en la evaluación de métodos de monitorización de calidad de servicios es necesario operacionalizar este modelo teórico para su uso en este tipo específico de métodos. Esta operacionalización consiste en definir los constructores del modelo en función de las variables relevantes para los

métodos de monitorización y redefinir los ítems del cuestionario que permitirán medir las variables de percepción.

De esta forma, este capítulo presenta la operacionalización realizada así como el diseño y ejecución de una familia de cuatro cuasi-experimentos realizada para proporcionar evidencias sobre la utilidad de Cloud MoS@RT. El método ha sido evaluado ejecutando las actividades envueltas en el proceso de monitorización. Los usuarios utilizaron un prototipo del configurador de la monitorización, el cual es un componente de la infraestructura de monitorización de Cloud MoS@RT que requiere la interacción del usuario.

A continuación, discutimos los estudios empíricos realizados para evaluar los métodos de monitorización existentes, los modelos teóricos de evaluación en Ingeniería del Software, y presentamos la adaptación del MEM para métodos de monitorización de servicios. Por último, presentamos el diseño, ejecución y análisis de datos de la familia de cuasi-experimentos realizada.

7.2. Estudios empíricos para los métodos de monitorización existentes

Como una tecnología joven, la computación en la nube y sus herramientas de monitorización aún sufren la falta de un consenso sobre un criterio de evaluación apropiado. Es deseable tener herramientas de monitorización adecuadas, las cuales deben ser evaluadas para asegurar su eficacia y utilidad de cara a sus usuarios. En los últimos años, se han realizado algunos estudios empíricos que tienen como objetivo evaluar los métodos y herramientas de monitorización existentes. Estos trabajos son discutidos a continuación.

El método propuesto por Bodenstaff *et al.* (2011), permite el manejo y monitorización de dependencias entre servicios en una composición. Los autores han evaluado empíricamente su enfoque por medio de un cuasi-experimento con 34 participantes expertos en el desarrollo y manejo de servicios, de los cuales 11 pertenecen a la industria, 9 de ellos sin embargo también son miembros activos de la academia y 23 pertenecen únicamente a la academia. Los autores validaron la utilidad de su método pidiendo a los participantes que realicen tres composiciones de diferente complejidad a través de simulaciones utilizando MoDe4SLA. Sin embargo, el objetivo del experimento no está dirigido a monitorizar la calidad de servicios, sino se centra en obtener una buena composición de los mismos.

El proyecto SLA@SOI[1] ha propuesto un marco de trabajo para el manejo de los servicios en base a los SLA. En el contexto de este proyecto, se han presentado algunas evaluaciones para ilustrar la aplicabilidad de su solución, la misma que incluye un marco de trabajo para la monitorización de servicios denominado EVEREST, en dominios gubernamentales (Armellin *et al.*, 2011). Sin embargo, estas evaluaciones constituyen unicamente una prueba de concepto y los resultados están centrados en la solución completa sin centrarse en la evaluación del enfoque de monitorización.

Emeakaroha *et al.* (2012) presentaron una arquitectura para la monitorización de servicios llamada CASViD, la cual tiene como objetivo la detección de violaciones de los SLAs para aplicaciones desplegadas en la nube, su solución fue evaluada haciendo uso de una prueba de concepto. Los autores evaluaron dos aspectos: (i) la habilidad de la arquitectura para monitorizar aplicaciones en tiempo de ejecución para así detectar las violaciones del SLA y (ii) su capacidad de automáticamente determinar el intervalo de medición apropiado para una monitorización eficiente. Sin embargo, la evaluación se ha centrado en aspectos de rendimiento y no incluye las percepciones de los usuarios al momento de utilizar esta solución.

Finalmente, algunos trabajos reportan experiencias del uso de herramientas de monitorización para evaluar la eficiencia, latencia, rendimiento y otros atributos de calidad de bajo nivel (Meng *et al.*, 2013; Montes *et al.*, 2013; Povedano-Molina *et al.*, 2013)

Montes *et al. (*2013) propuso un enfoque para monitorización de servicios cloud llamado GMonE (*Global Monitoring systEm*), el cual constituye una herramienta de monitorización. Los autores evaluaron el desempeño, escalabilidad y sobrecarga de GMonE utilizando un banco de pruebas. Ellos probaron los beneficios de su enfoque en un ambiente cloud a gran escala, incluyendo el alto desempeño, baja sobrecarga, escalabilidad y elasticidad. Sin embargo, los autores no han realizado pruebas en diferentes escenarios y ambientes heterogéneos, observando el comportamiento de los servicios desde diferentes puntos de vista y presentando una manera flexible de cambiar los parámetros de monitorización.

Meng *et al. (*2013) ejecutaron experimentos por medio de simulaciones de un ambiente cloud con un sistema del mundo real y trazas de red. Los resultados muestran que su enfoque consigue bajar significativamente los costos, tener una alta escalabilidad y un mejor rendimiento multitenencia que otros. Sin embargo, su evaluación no incluye ambientes reales y usuarios para proveer retro-

[1] http://sla-at-soi.eu/

alimentación y contribuir hacia mejorar el enfoque en base a las percepciones de los usuarios.

El análisis de los estudios mencionados anteriormente ha permitido que se identifiquen algunas limitaciones en cuanto a las evaluaciones empíricas de las soluciones de monitorización, tales como (1) el bajo número de estudios empíricos que evalúen la experiencia de los usuarios utilizando la aproximación de monitorización; (2) la carencia de estudios que analicen la interacción entre la solución de monitorización y sus usuarios para la definición de las características de calidad a ser monitorizadas, y (3) el análisis de la intención de uso de una solución dada cuando los usuarios necesiten monitorizar sus servicios cloud.

7.3. Modelos teóricos de evaluación en Ingeniería del Software

Dentro de la Ingeniería del Software es importante tener en cuenta el rol de las personas en el proceso de adopción de una solución. Esto ha sido direccionado en el campo de las Ciencias Sociales a través del desarrollo de modelos teóricos que expliquen la aceptación de la tecnología, metodologías y métodos. En este campo, los modelos actuales de aceptación tecnológica tienen sus raíces en un número de diversas perspectivas teóricas. Tales modelos teóricos incorporan constructores para medir las reacciones psicológicas del usuario y factores organizacionales de una manera sistemática.

El *Technology Acceptance Model* (TAM) propuesto por Davis (1989) es un modelo teórico ampliamente usado desde la perspectiva del uso en general de un sistema.

Por otro lado, algunos modelos teóricos han sido empleados en el campo de la Ingeniería del Software (IS) para explicar la aceptación de metodologías y métodos de IS, por parte del desarrollador de software. Johnson *et al. (*1999) identificaron algunas creencias con respecto a la actitud, norma subjetiva, y constructores de control del comportamiento correspondientes a la *Theory of Planned Behavior (TPB)*. Sin embargo, las relaciones entre esos constructores no han sido empíricamente probadas.

Riemenschneider *et al. (*2002) examinaron cinco modelos teóricos de intenciones individuales para aceptar herramientas en el contexto de las metodologías del software. Ellos realizaron un estudio con 128 desarrolladores en una organización de gran tamaño. Los resultados mostraron que si una metodología no es recordada como útil por los desarrolladores, sus expectativas de un exitoso despliegue podrían verse severamente limitadas. Los autores

además explican por qué los desarrolladores de software aceptan o se resisten a ciertas metodologías.

El *Method Evaluation Model (MEM)* propuesto por Moody (2001) provee mecanismos para evaluar el rendimiento actual, la aceptación y la posible adopción de un método de sistemas de información en la práctica.

A continuación, se presentan los modelos teóricos más relevantes para el propósito de esta tesis.

7.3.1. Technology acceptance model (TAM)

El modelo TAM propuesto por Davis (1989) es uno de los más frecuentemente usados dentro del campo de los Sistemas de Información, e intenta predecir y explicar el comportamiento en el uso de la tecnología.

El TAM constituye una adaptación de la teoría *Theory of Reasoned Action* (TRA) propuesta por Fishbein *et al. (*1975). TAM usa TRA como una base teórica para especificar los vínculos causales entre dos pensamientos clave: utilidad percibida y facilidad de uso percibida y las actitudes, intenciones y comportamiento de los usuarios al momento de la adopción de una tecnología de computación. TAM es considerablemente menos general que TRA, está diseñado exclusivamente para el comportamiento en el uso del computador, pero éste incorpora hallazgos acumulados desde hace mucho tiempo en investigación de Ingeniería del Software.

De acuerdo a TRA, el rendimiento de una persona de una conducta específica está determinado por la intención de su comportamiento y la intención de su comportamiento está conjuntamente determinada por la actitud de la persona y las normas subjetivas concernientes al comportamiento en cuestión Davis, (1986). El TRA es un modelo general y como tal, este no especifica las creencias operativas para un comportamiento particular. Utilizando TRA se deberían primero identificar las creencias que son salientes para sujetos de acuerdo al comportamiento bajo investigación.

El TAM utiliza dos creencias de adaptadores potenciales, la facilidad de uso percibida y la utilidad percibida de la tecnología como los principales determinantes de las actitudes hacia una nueva tecnología. Estas actitudes influencian las intenciones y de ahí el comportamiento. En este modelo, el uso es modelado como una función directa de la intención. (Ver la Figura 7-1).

El significado de cada constructor de TAM es:

- Facilidad de Uso Percibida (PEOU): el grado al cual los usuarios esperan que el sistema objetivo sea libre de esfuerzo.

- Utilidad Percibida (PU): la probabilidad subjetiva del usuario de que utilizando una aplicación específica podría incrementar su rendimiento laboral en un contexto organizacional.
- Actitud (A): el deseo del usuario para usar el sistema. Tanto PU como PEOU predicen la actitud hacia usar el sistema.
- Intención de Comportamiento (IC): la medida de la resistencia a ejecutar un comportamiento específico. A y PU influencian al individuo de IC a usar el sistema
- Uso: el uso actual del sistema. Este es predicho por intenciones del comportamiento.

Se han hecho algunas réplicas del estudio original de Davis para proveer evidencia empírica sobre las relaciones que existen entre la facilidad de uso, la utilidad y el uso del sistema. Por ejemplo (Adams *et al.,* 1992; Hendrickson *et al.*, 1993; Segars *et al.*, 1993; Szajna, 1994). Por ejemplo Adams *et al.* (1992) replicó el estudio de Davis para demostrar la validez y confiabilidad de su instrumento y escala de medidas. Segars *et al. (*1993) reexaminaron posteriormente la réplica de Adams. Ellos fueron críticos en acerca del modelo de medición usado, y postularon un modelo diferente basado en tres constructores: utilidad, efectividad y facilidad de uso.

Además, un gran número de estudios experimentales han validado el TAM usando una variedad de sistemas específicos, tales como la predicción de uso de sitios Web (Fenech, 1998) y el éxito de aplicaciones comerciales existentes (Schubert *et al.*, 2002).

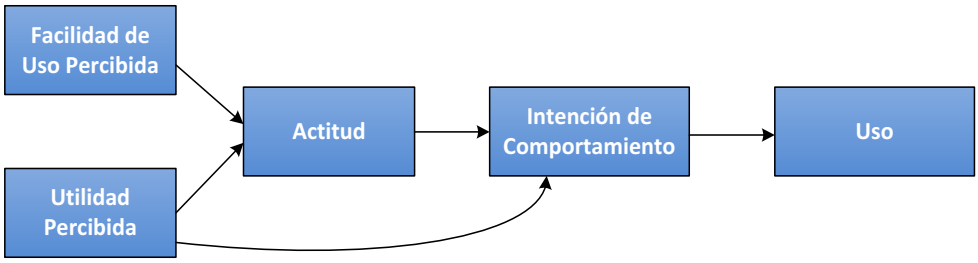


Figura 7-1 Technology Acceptance Model (TAM) simplificado (F. Davis, 1986)

7.3.2. Method Evaluation Model (MEM)

La principal contribución del MEM es que este incorpora dos diferentes aspectos del método de éxito: la eficiencia actual y el uso actual. La Figura 7-2

ilustra las diferentes partes de este método. Esto significa que la adopción de un método en práctica depende no solamente de si este es efectivo (éxito pragmático) (Abrahão et al., 2011), sino además de si los usuarios de un método lo perciben efectivo (éxito percibido). Ambos aspectos deben ser considerados para evaluar métodos de monitorización en la nube. La figura además muestra los constructores del modelo, así como también las relaciones causales a lo largo de los constructores del mismo.

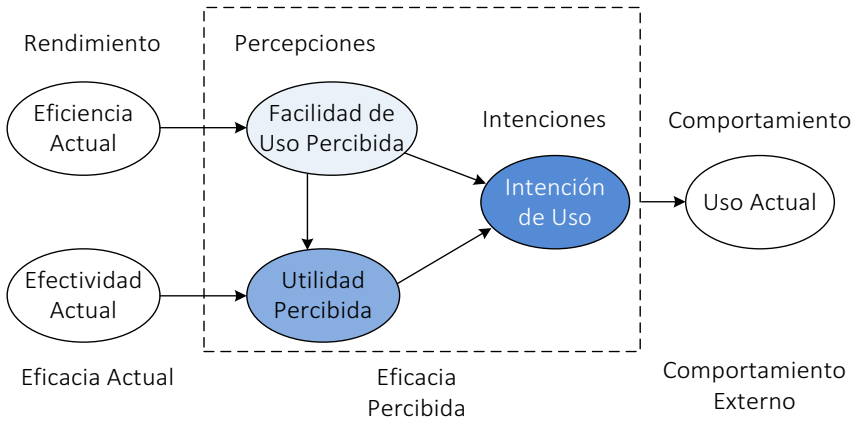


Figura 7-2 Method Evaluation Model – MEM (Fuente: Moody (2001)

En el MEM, la eficacia es definida como un constructor separado, el cual es diferente de la eficiencia y la efectividad. El constructor de la eficacia es derivado de la noción del éxito pragmático de Rescher, (1973), el cual es definido como la eficacia y la efectividad con la cual un método consigue sus obejtivos. La evaluación de la eficacia de un método requiere la medición tanto del esfuerzo requerido (eficiencia) y la calidad de los resultados (efectividad).

Los constructores del MEM están basados en el *Technology Acceptance Model (TAM),* un método bien conocido y validado empíricamente para la evaluación de tecnologías de la información. Los constructores del MEM son:

- *Eficacia Actual*: este constructor tiene dos variables basadas en el rendimiento del usuario:
  - *Eficiencia Actual*: es el esfuerzo requerido para aplicar un método.
  - *Efectividad Actual*: es el grado en el cual un método consigue sus objetivos. Este constructor está relacionado a la calidad de los artefactos obtenidos al momento de aplicar el método. Según Rescher (1973), todos los métodos intentan conseguir ciertos objetivos. Rescher define un método como "una colección de reglas y procedimientos designados para asistir

a la gente al momento de ejecutar una acción particular". Diferentes tipos de métodos son definidos para diferentes objetivos. Esto significa que variables específicas dependientes se pueden necesitar para definir cada clase de métodos para medir el rendimiento con respecto a sus objetivos específicos.

- *Eficacia Percibida*, la cual tiene dos variables basadas en la percepción:
  - *Facilidad de Uso Percibida*: es el grado en el cual una persona cree que usando un método en particular puede estar libre de esfuerzo. La facilidad de uso percibida representa un juicio perceptivo del esfuerzo requerido para aprender y usar el método en cuestión.
  - *Utilidad Percibida*: es el grado en el cual una persona cree que usando un método particular podría mejorar su rendimiento en el trabajo. Esta variable representa un juicio perceptual de la efectividad del método. Existe una relación causal en el modelo el cual indica que la utilidad percibida puede estar determinada por la facilidad de uso percibida.
  - *Intención de Uso*: es el modo en el que una persona intenta usar un método particular. Esta variable representa un juico perceptual de la eficacia y efectividad de coste del método. Esta variable es usada para medir la probabilidad del método para ser adoptado en la práctica. Las relaciones causales sugeridas que perciben la facilidad de uso y la utilidad percibida directamente afectan la intención de usar el método.
- *Uso Actual*, el cual representa una variable basada en el comportamiento, definida como la intención de utilizar un método en la práctica. De acuerdo a la relación causal hipotética, el uso actual debe estar determinado por la intención del uso.

  La predicción de la intención de uso e posible aceptación está indicada para métodos recientemente propuestos mientras que la adopción en la práctica (uso actual) solo puede ser evaluada en métodos establecidos, que ya han sido adoptados en la industria (Abrahão *et al.*, 2011).

7.4. Adaptando el MEM para su uso en métodos de monitorización

El primer paso involucrado para adaptar el MEM es definir los objetivos específicos de los métodos de monitorización de servicios cloud. Los constructores generales del MEM pueden ser instanciados en variables dependientes concretas basadas en estos objetivos.

De los métodos de monitorización existentes se aprecian tres objetivos principales: (1) configurar los RNF a ser monitorizados y la manera en la que los datos crudos serán recogidos directamente de los servicios a ser monitorizados (2) recoger los datos siguiendo la estrategia de configuración selecionada y evaluar la calidad de los servicios y (3) analizar los datos medidos y compararlos con los umbrales establecidos. Este capítulo se centra en el primer objetivo (la configuración de los RNF a ser monitorizados mediante la utilización de un método específico de monitorización), ya que esta es una actividad común y esencial para la monitorización de servicios cloud en la cual interviene activamente el usuario.

La evaluación de la eficacia de los métodos de monitorización cloud involucra la medición del esfuerzo requerido para aplicar el método (entrada) y la calidad de los resultados de la monitorización obtenida (salida). El esfuerzo requerido para entender y/o aplicar el método (eficiencia actual) puede ser medido utilizando algunas medidas, tales como el tiempo o el esfuerzo cognitivo. La calidad del resultado del método (actual efectividad) puede ser medida evaluando los resultados de monitorización que son producidos usando el método de monitorización (p. ej., si la configuración de la monitorización fue correctamente efectuada). Las siguientes variables de rendimiento son así utilizadas para medir la eficiencia y efectividad de los usuarios con respecto a la configuración de los RNF a ser monitorizados:

- *Efectividad actual:* la proporción entre el número de RNFs correctamente configurados y el número total de RNFs a ser configurados.

$$Efectividad = \frac{\#\text{RNFs correctamente configurados}}{\text{Número total de RNFs a ser configurados}}$$

- *Eficiencia actual*: el tiempo empleado para configurar el número total de RNFs a ser monitorizados.

$$Eficiencia = \sum_{i=1}^{n} Tiempo\ utilizado\ en\ configurar\ el\ NFR_i$$

Con el objetivo de medir las variables basadas en percepción, se ha adaptado un instrumento de medición utilizado en el MEM. La Figura 7-2 muestra como el MEM ha sido operacionalizado para evaluar métodos de monitorización de servicios cloud. Para medir cada uno de los tres constructores (facilidad de uso percibida, utilidad percibida e intención de uso), se han definido conjuntos de preguntas basados en los ítems mostrados en la Tabla 7-1. La Figura 7-3 muestra

el modelo teórico propuesto para evaluar la calidad de los métodos de monitorización de servicios cloud. Básicamente, se han usado medidas basadas en el rendimiento como factores que influencian las variables basadas en la percepción (ver Figura 7.4).

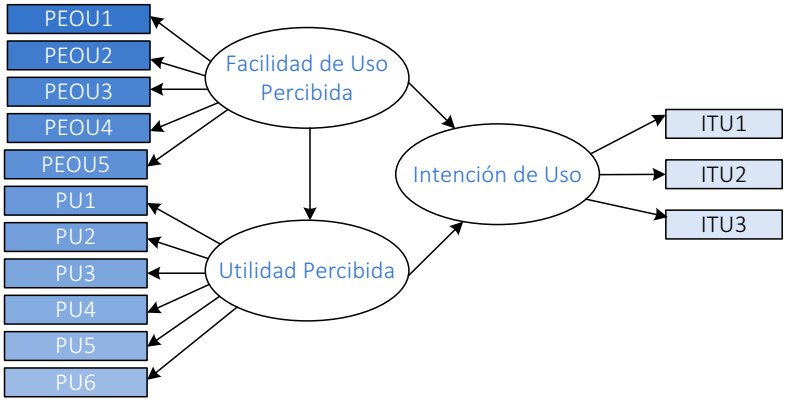


Figura 7-3. Distribución de preguntas del cuestionario aplicado al cuasi-experimento.

De acuerdo a nuestra adaptación de MEM, la probabilidad de que un método de monitorización de servicios cloud sea aceptado en la práctica puede ser predicho probando las siguientes hipótesis:

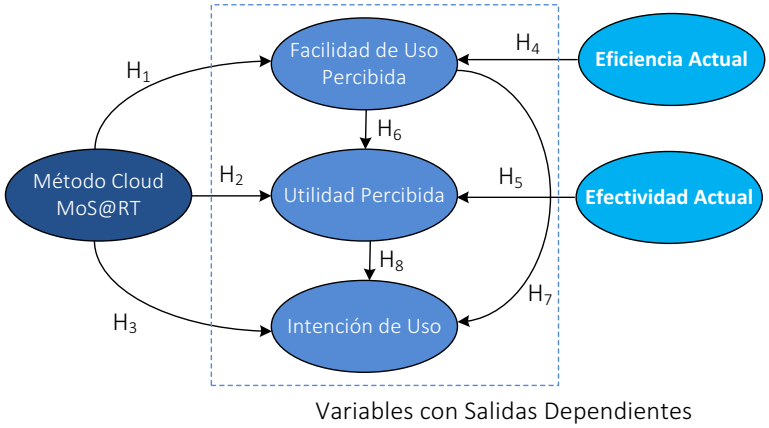


Figura 7-4. Modelo teórico para la evaluación de los métodos de monitorización cloud

- $H1_0$: El método de monitorización de servicios cloud se percibe como difícil de usar, $H1_0 = \neg H1_1$.
- $H2_0$: El método de monitorización de servicios cloud no se percibe como útil, $H2_0 = \neg H2_1$

- $H3_0$: No existe intención de utilizar este método de monitorización de servicios cloud en el futuro $H3_0=\neg H3_1$.

A continuación se presentan las hipótesis que muestran una relación directa entre el uso de un método particular de monitorización de servicios cloud y el rendimiento, percepciones e intenciones de los usuarios. El modelo de evaluación además propone un número de hipótesis que indican una relación causal entre las variables dependientes (tales como rendimiento teniendo un efecto sobre percepciones o percepciones que influencian intenciones):

- $H4_0$: La facilidad de uso percibida no puede verse determinada por la eficiencia, $H4_0=\neg H4_1$. La razón de esta hipótesis es que la eficiencia representa una medida basada en el rendimiento de la eficiencia actual, mientras que la facilidad de uso percibida representa una medida basada en la percepción. De acuerdo a MEM, la eficiencia mide el esfuerzo requerido para aplicar el método, el cual debería determinar las percepciones del esfuerzo requerido.
- $H5_0$: La percepción de la utilidad no está determinada por la efectividad. $H5_0=\neg H5_1$. : La razón de esta hipótesis es que la efectividad representa una medida basada en el rendimiento, mientras que la utilidad percibida representa una medida basada en la percepción de la efectividad. De acuerdo al MEM las percepciones de efectividad deberían estar determinadas por la efectividad actual.
- $H6_0$: La utilidad percibida no es determinada por la facilidad de uso percibida $H6_0=\neg H6_1$ Esta hipótesis es tomada desde el TAM, en el cual la facilidad de uso percibida se encuentra que no tiene una influencia directa sobre la utilidad percibida.
- $H7_0$: La intención de uso no es determinada por la facilidad de uso percibida $H7_0=\neg H7_1$ Esta hipótesis es tomada desde el TAM en el cual se encuentra que la facilidad de uso percibida tiene influencia sobre la intención de uso.
- $H8_0$: La intención de uso no está determinada por la utilidad percibida. $H8_0=\neg H8_1$ Esta hipótesis es tomada del TAM, en la cual se encontró que la utilidad percibida tiene una influencia directa sobre la intención de uso.

El modelo de evaluación consecuentemente denota que los métodos de monitorización cloud serán aceptados en la práctica en las bases de las percepciones de su facilidad de uso y utilidad. Es importante examinar y medir la correlación entre intención de uso y uso actual cuando empleamos modelos basados en TAM. Sin embargo, las medidas de uso actual impactan en la práctica (como opuestas al impacto potencial definido por la intención de uso). Esto

podría ser evaluado utilizando encuestas de práctica (no puede ser usado en la evaluación de nuevos métodos sino solamente para métodos ya establecidos).

La Tabla 7-1 muestra los ítems definidos para medir las variables basadas en la percepción. Estos ítems fueron combinados en un cuestionario con 14 preguntas. Los ítems fueron formulados utilizando una escala de 5 puntos de Likert, con el formato de preguntas opuestas. Varios ítems con el mismo grupo de constructores fueron aleatorizados para prevenir errores de respuesta sistemática. La facilidad de uso percibida se mide utilizando cuatro ítems del cuestionario. Además, para asegurar el balance de los ítems, aproximadamente la mitad de las preguntas fueron negadas, para evitar respuestas monótonas. El instrumento de medición se encuentra disponible en http://goo.gl/forms/JxMEh4TY5t.

Tabla 7-1. Cuestionario para medir las variables de percepción

Pregunta	Declaración Positiva (5 Puntos)
PEOU1	El método para monitorizar la calidad de los servicios cloud sencillo y fácil de seguir
PEOU2	En general, el método para monitorizar la calidad de los servicios cIoud es fácil de entender.
PEOU3	Los pasos usados para configurar la monitorización de la calidad de los servicios cloud son claros y fáciles de entender.
PEOU4	El método para monitorizar la calidad de los servicios cloud es fácil de aprender.
PEOU5	Considero que sería fácil adquirir destrezas en el uso de este método.
PU1	Considero que este método podría reducir el tiempo y esfuerzo requerido para monitorizar la calidad de servicios cloud.
PU2	En general, considero que el método para monitorizar la calidad de servicios cloud es útil.
PU3	Considero que el proceso para configurar los RNF de este método es útil para monitorizar la calidad de servicios cloud.
PU4	Pienso que el método es lo suficientemente expresivo para definir como la medición de los requisitos de monitorización debería ser ejecutada.
PU5	El uso de este método podría mejorar mi rendimiento cuando monitorizo la calidad de servicios cloud.
PU6	En general, pienso que este método podría permitirme monitorizar la calidad de servicios cloud de una manera adecuada.
ITU1	Si necesitaría utilizar un método de monitorización de servicios cloud en el futuro, consideraría este método.
ITU2	De ser necesario, utilizaría este método en el futuro.
ITU3	Recomendaría el uso de este método para monitorizar la calidad de servicios cloud.

7.5. Evaluando la utilidad percibida de Cloud MoS@RT en la práctica: una familia de cuasi-experimentos

En el campo de la Ingeniería del Software, como en cualquier otra disciplina técnica, se emplean estudios empíricos para evaluar los procesos, métodos, herramientas o cualquier otro tipo de tecnología creadas para al desarrollo, mantenimiento o aseguramiento de la calidad del software (Basili *et al.*, 1986; Basili, 1993). Un estudio empírico consiste en un acto u operación que permite descubrir algo que no se conoce o poner a prueba una hipótesis (Basili, 1993). Como se ha discutido en el Capítulo 1, las estrategias de investigación incluyen experimentos controlados, estudios cualitativos (encuestas o entrevistas) o casos de estudio (Wohlin *et al.*, 2012). Sin embargo, con la finalidad de mejorar la precisión y validez de los resultados, es necesario realizar réplicas de experimentos (Shull *et al.*, 2008). En el trabajo de Basili *et al. (*1999) se extiende el concepto de réplicas hacia familias de experimentos. Siendo una familia de experimentos aquella que permite construir el conocimiento necesario para extraer conclusiones significativas a través de varios experimentos similares.

En esta sección, se presenta la familia de experimentos llevada a cabo para validar empíricamente la fase de la configuración de la monitorización del método Cloud MoS@RT. En vista de que actualmente no existe un estándar o método de monitorización para servicios cloud ampliamente aceptado, nosotros no podemos evaluar Cloud MoS@RT con respecto a otro método que sirva de control. De ahí, hemos decidido llevar a cabo un cuasi-experimento con el objetivo de evaluar el método y probarlo empíricamente.

Un cuasi-experimento es una investigación empírica, similar a un experimento, en la cual la asignación de tratamiento a sujetos no puede estar basada en la aleatoriedad, pero emerge desde las características de los sujetos u objetos en sí mismo (Wohlin, 2007).

El cuasi-experimento fue diseñado de acuerdo al proceso experimental propuesto por Wohlin (2007). Y en este caso será utilizado para probar la eficacia percibida de Cloud MoS@RT. Para este fin, nuestro método de monitorización y la infraestructura del mismo fueron aplicados para predecir la probabilidad de aceptación de la actividad de la configuración de monitorización como parte del método en la práctica (la tarea de configuración de la monitorización). Un estudio basado en las percepciones de los usuarios, puede ayudarnos a entender las necesidades de usuarios para refinar la actividad de configuración, la misma que es la parte de nuestro método que necesita contar con la interacción del usuario.

De acuerdo al paradigma *Goal-Question Metric (GQM)* propuesta Basili *et al.* (1994) la meta de este experimento ha sido definida de la siguiente manera:

Evaluar:	la fase de configuración de Cloud MoS@RT
Con el propósito de:	evaluar el método propuesto con respecto a su eficacia percibida.
Desde el punto de vista del:	investigador
En el contexto de:	un grupo de alumnos y profesionales en Ciencias de la Computación.

Las preguntas de investigación son:

RQ1: ¿Cloud MoS@RT es percibido como fácil de usar y útil? De ser así, ¿las percepciones de los usuarios son el resultado de su rendimiento cuando utilizan el método para configurar los requisitos de calidad a ser monitorizados?

RQ2: ¿Existe una intención de uso de Cloud MoS@RT en el futuro? De ser así, ¿tales intenciones de uso es el resultado de las percepciones de los participantes?

Estas preguntas de investigación pueden ser evaluadas a través de la prueba de varias hipótesis (ver Figura 7-13). En particular, la primera pregunta de investigación puede ser estudiada mediante las siguientes hipótesis:

- $H1_0$: Cloud MoS@RT es percibido como difícil de usar, $H1_0=\neg H1_1$.
- $H2_0$: Cloud MoS@RT no es percibido como un método útil, $H2_0=\neg H2_1$
- $H4_0$: La facilidad de uso percibida no puede verse determinada por la eficiencia actual, $H4_0=\neg H4_1$.
- $H5_0$: La percepción de la utilidad no está determinada por la efectividad actual. $H5_0=\neg H5_1$.

Por otra parte, la segunda pregunta de investigación puede ser estudiada a través de la formulación de las siguientes hipótesis:

- $H3_0$: No existe intención de utilizar Cloud MoS@RT en el futuro $H3_0=\neg H3_1$.
- $H6_0$: La utilidad percibida no es determinada por la facilidad de uso percibida $H6_0=\neg H6_1$.
- $H7_0$: La intención de uso no es determinada por la facilidad de uso percibida $H7_0=\neg H7_1$.
- $H8_0$: La intención de uso no está determinada por la utilidad percibida. $H8_0=\neg H8_1$.

7.5.1. Planificación del cuasi-experimento

7.5.1.1. Selección del contexto

El contexto está determinado por el método de monitorización a ser evaluado, la selección del servicio cloud a ser evaluado y la selección de los participantes.

El método de monitorización a ser evaluado es el presentado en este trabajo. Aquí, nos centraremos en la actividad de la configuración, la cual es usada para genera el Modelo de Calidad en Tiempo de Ejecución.

Esta actividad es importante debido a la necesidad de la interacción del usuario con la infraestructura de monitorización. Los sujetos de ahí, ejecutarán el rol de Configurador de la Monitorización (ver Figura 4-2) el cual incluye las siguientes tareas: (i) selección de los atributos de calidad, (ii) selección de métricas, (iii) mapeo de métricas, y (iv) generación del modelo de monitorización.

El configurador de la monitorización utilizará el Modelo de Requisitos de Monitorización, provisto por el Planificador de la Monitorización (ver Figura 4-2), el cual incluye los RNF contenidos en el SLA y los RNF adicionales a ser monitorizados. Además, los participantes utilizarán un modelo de calidad que incluirá las características, sub-características, atributos de calidad y métricas incluidas en el Modelo de Calidad SaaS con la finalidad de realizar el mapeo de los RNF que se requieren monitorizar con el modelo de calidad, para que de esta manera sea más sencilla y normalizada la selección de las métricas que se utilizarán a lo largo de la monitorización.

Cuando la configuración de la monitorización sea completada, los participantes generarán el Modelo en tiempo de Ejecución, el cual será usado por el Middleware de Monitorización y Análisis para automáticamente monitorizar la calidad de los servicios.

El **servicio cloud** a ser evaluado pertenece a un sitio de subastas en línea. Un sitio de subastas en línea es un proceso para comprar y vender artículos o servicios ofreciéndolos a través de pujas que luego venden el ítem al mejor postor a través de Internet. Estos sitios permiten a miles de usuarios realizar pujas por tanto, deben cumplir ciertos niveles de calidad, tales como una alta disponibilidad, elasticidad, o precisión. Los sitios de subastas permiten pujar a los usuarios haciendo alusión a diferentes formatos, siendo las más populares las pujas ascendentes y descendentes. El servicio a ser evaluado en este experimento está relacionado a un sitio de pujas ascendentes. Estos sitios han proliferado en

Internet y representan un claro ejemplo de qué es lo que se quiere mostrar con respecto a los requisitos de calidad que pueden ser monitorizados utilizando nuestro enfoque. El proceso es el siguiente:

1) El usuario debe registrarse y comprar créditos. Este proceso se realiza utilizando un carrito de compras, muy común de sitios de comercio electrónico.
2) El usuario interesado en un producto en particular usa sus créditos para enviar una puja.
3) El usuario espera que un contador propio de la puja llegue a 0.
4) Si existe otro usuario quien además está interesado en el producto y desea comprarlo, este puja y el contador vuelve a reiniciarse.
5) Finalmente, si nadie más puja en la subasta, esta termina y el producto es adjudicado al último usuario en realizar una puja, quien es el ganador.

Muchos sitios que hoy en día se han popularizado siguen este modelo de negocios, tal es el caso por ejemplo de MadBid, QuiBids, DealDah, los cuales se especializan principalmente en subastas de productos electrónicos, joyería, productos de hogar, coches, etc. De esta manera, existen un conjunto de servicios cloud que son ofrecidos para este tipo de dominios.

De los servicios que el sitio ofrece (p. ej. servicio de inventarios, servicio de subastas, servicio de pagos, servicio de manejo de incidentes), para este experimento se ha escogido el **Servicio de Subastas,** el cual es el más importante y crítico. Los altos niveles de calidad que éste demanda (p. ej. disponibilidad, confiabilidad, baja latencia, elasticidad, precisión), proporciona a este experimento un interesante problema a ser resuelto. La configuración de la monitorización consistirá en la configuración de tres RNF.

El Modelo de Requisitos de Monitorización es provisto por el Planificador de la Monitorización y especifica los RNF y las métricas a ser consideradas. Por ejemplo, el servicio debería tener un máximo de 10 operaciones defectuosas por millón, lo cual representa una confiabilidad de 99.999% y puede ser medida utilizando la siguiente función de medición:

$$DPM = \frac{Operaciones\ Intentadas - Operaciones\ exitosas}{Operaciones\ intentadas} * 10^6$$

Finalmente, 58 participantes fueron seleccionados, todos ellos estudiantes de la Universidad Politécnica de Valencia. Se ha tomado una muestra consistente en dos grupos de 37 participantes el grupo de mañana y el de la tarde con 21 participantes. Los estudiantes tienen un muy buen conocimiento de modelos de

calidad, métricas y métodos de evaluación. El experimento estuvo organizado como una parte obligatoria del curso de Calidad.

7.5.1.2. Tareas experimentales

El cuasi-experimento consistió en tres tareas:

Tarea 1: La categorización de tres RNF a ser monitorizados y la selección de métricas y operacionalizaciones. Para cada RNF, los participantes tiene que utilizar el Modelo de Calidad SaaS para seleccionar las características de calidad apropiadas, atributos y operacionalización de una métrica independiente de la plataforma la cual les permita evaluar un RNF particular. La metra seleccionada tiene que ser equivalente a la métrica expresada en el Modelo de Requisitos de Monitorización, la cual describe los RNF a ser monitorizados. Las métricas y operacionalizaciones envueltas en esta tarea son independientes de la plataforma y la tarea es soportada por un prototipo del configurador de la monitorización.

Tarea 2: La selección de la función de medición dependiente de la plataforma más apropiada para cada métrica seleccionada en la tarea 1. Esto permite que el mapeo sea hecho entre la definición genérica de la métrica (función de medición) y los contadores de bajo nivel provistos por la plataforma, entonces permite datos crudos a ser recogidos desde el servicio y provistos por la plataforma, permitiendo que datos crudos sean recogidos desde el servicio y su monitorización en tiempo de ejecución. Aquí, una lista de contadores de la plataforma tomados desde Azure fueron mostrados en el configurador, permitiendo a los participantes construir la función de medición con la cual evaluar cada RNF a ser monitorizado.

Tarea 3: La modificación de un RNF debido a una renegociación de un SLA. Los participantes tuvieron que analizar la modificación de un RNF para determinar los cambios en la categorización de los atributos y métricas (Tarea 1) y necesitan métricas dependientes de la plataforma y contadores de bajo nivel obtenidos desde la plataforma (Tarea 2).

7.5.1.3. Variables

La Tabla 7-2 muestra las variables dependientes de interés basadas en la percepción, de acuerdo al MEM, las cuales fueron usadas para evaluar Cloud MoS@RT en la práctica.

Tabla 7-2. Variables dependientes basadas en la percepción

Variable	Descripción
Facilidad de Uso Percibida (PEOU)	El grado en el cual los participantes creen que al aprender y usar Cloud MoS@RT estarán libres de esfuerzo.
Utilidad Percibida (PU)	El grado en el cual los participantes creen que usando Cloud MoS@RT se incrementará su rendimiento.
Intención de Uso (ITU)	El grado en el cual los participantes piensan usar Cloud MoS@RT de necesitar un método de monitorización de calidad de servicios cloud. Esto representa un juicio de la eficacia del método y puede ser utilizado para predecir la aceptación del método en práctica.

Estas variables son medidas usando un cuestionario con una escala de Likert con un con un conjunto de 14 preguntas cerradas (ej. 5 para facilidad de uso percibida, 6 para utilidad percibida y 3 para intención de uso futura). Las preguntas cerradas fueron formuladas utilizando una escala de Likert de 5 puntos. El valor agregado para cada variable subjetiva fue calculado como la media aritmética de las respuestas a las preguntas asociadas con cada variable dependiente subjetiva.

En la Tabla 7-3 se muestran las variables basadas en el rendimiento de interés y la función de medición usada para determinar sus valores.

Tabla 7-3. Variables dependientes basadas en el rendimiento.

Variable	Descripción
Efectividad	$\frac{\sum_{i=1}^{n} Tarea_i \text{ ejecutada correctamente}}{n}$
Eficiencia	$\sum_{i=1}^{n} Tiempo \text{ ejecutando la tarea}_i$

Como se mencionó previamente, el experimento consistió en la ejecución de tres tareas. En la primera y segunda tarea los participantes configuraron tres NFRs. Cada NFR puntuó como un tercio del valor total de cada tarea. La efectividad de la primera y segunda tarea es de ahí la suma de las acciones correctamente ejecutadas con cada NFR. La efectividad de la tercera tarea puede ir de 0-1 significando que la suma total de la efectividad es la suma de las actividades correctamente ejecutadas para el número total de tareas. Finalmente, como el MEM sugiere, la eficiencia fue medida como el tiempo total gastado en la configuración de cada NFR en cada tarea ejecutada por los participantes.

7.5.1.4. Material experimental

El material experimental se compone del conjunto de documentos que son necesarios para realizar las tareas experimentales y el cuestionario parte medir la percepción del usuario una vez que se ha realizado el experimento.

Este material incluye todo lo referente a las presentaciones con los conceptos, explicación de tareas y ejemplos que son utilizadas para el entrenamiento de los participantes.

La documentación utilizada, en su totalidad ha sido incluida en los anexos de este trabajo y es explicada a continuación:

1. Un folleto que contiene la descripción del dominio, en este caso, el sitio de subastas con el servicio a ser monitorizado, los RNF a ser monitorizados, y las tres tareas a ser ejecutadas por los participantes. Se solicitó a los participantes que escriban la hora exacta de inicio y fin de las tareas a resolver.
2. Un anexo detallado como soporte, el cual describe cada RNF a ser monitorizado.
3. Un anexo con un fragmento del Modelo de Calidad SaaS.
4. Un fragmento de la lista de contadores provistos por la plataforma Azure.
5. Una guía del método de monitorización para ser utilizada durante el experimento como material de referencia.
6. El configurador de la monitorización
7. El cuestionario, el mismo que contiene preguntas cerradas para analizar las variables subjetivas y algunas respuestas abiertas para permitir a los participantes expresar su opinión sobre el método y la infraestructura que lo soporta.

Los datos fueron recolectados usando el configurador de la monitorización. El tiempo gastado en cada tarea y los datos obtenidos como resultado de la Tarea 3 fueron recolectados utilizando el folleto descrito en el numeral uno. Finalmente, el cuestionario fue recolectado en línea.

Todos los documentos fueron creados en español, ya que éste es el idioma nativo de los participantes en todos los experimentos. Todo el material (incluyendo las tareas experimentales y las diapositivas están disponibles para su descarga en http://users.dsic.upv.es/~icedillo/EjercicioCalidad); finalmente, todo el material de entrenamiento puede ser encontrado en http://users.dsic.upv.es/~icedillo/Entrenamiento.

7.5.2. Operación y ejecución de los cuasi-experimentos

Cuatro experimentos individuales fueron realizados, en cada uno de los cuales, tres sesiones de entrenamiento de 120 minutos fueron ejecutadas antes de la sesión experimental con el objetivo de presentar los conceptos de computación cloud y el método de monitorización a los participantes.

El entrenamiento incluyó el uso del prototipo del Configurador de la Monitorización y las tareas envueltas en el proceso de configuración. Como un ejemplo se realizó una sesión de entrenamiento utilizando el Caso de Referencia Abierta (ORC) propuesto en Wieder *et al.* (2011), el cual fue utilizado como un demostrador para mostrar los resultados del proyecto de investigación europeo SLA@SOI. El ORC es una extensión de la implementación de CoCoMe propuesta por Herold *et al.* (2008), el cual provee una solución orientada a servicios que trata de un sistema de distribución de un supermercado para manejar las ventas y los procesos de inventarios (Herold *et al.*, 2008). El conjunto de servicios fue desplegado como SaaS sobre la plataforma Microsoft Azure. En el entrenamiento, los participantes usaron el configurador para la configuración de dos RNF (eficiencia y precisión del servicio) para el servicio de Inventarios.

La Figura 7-5 resume la familia de cuasi-experimentos, en la que se muestran en cada rectángulo los detalles asociados a cada uno de ellos.

1er. Cuasi-experimento	2do. Cuasi-experimento	3er. Cuasi-experimento	4to. Cuasi-experimento
Universitat Politècnica de València (UPV1)	Universidad Nacional de Asunción (UNA)	Universidad de Cuenca (UC)	Universitat Politècnica de València (UPV2)
58 Estudiantes de Ingeniería Informática	14 Estudiantes de Máster	10 Estudiantes de Ingeniería de Sistemas	60 Estudiantes de Ingeniería Informática
	Replicación	Replicación	Replicación

Factor Principal: Método MoS@RT vs Variable Neutral
Variables dependientes: Eficacia, eficiencia, facilidad de uso percibida, utilidad percibida e intención de uso.

Figura 7-5. Resumen de la familia de experimentos

El cuasi-experimento original (UPV1) se replicó con el fin de obtener evidencias sobre las variables a ser medidas y dar una primera aproximación hacia la búsqueda de la intención de uso de la solución creada. El segundo cuasi-experimento (UNA) es una réplica diferenciada del experimento original en el cual se cambió el perfil de los sujetos, haciéndolos con estudiantes de máster quienes ya se han desarrollado en el ámbito profesional y en diferentes entornos,

éste fue realizado en la Universidad Nacional de Asunción en Paraguay. Por otro lado el tercer cuasi-experimento (UC) se realizó sobre un grupo de estudiantes pequeño, con un entorno completamente diferente y con un pensum de estudios bastante encaminado a la Ingeniería del Software, estos alumnos estudian en la Universidad de Cuenca en Ecuador. Finalmente, el cuarto cuasi-experimento ha sido realizado nuevamente con estudiantes de la Universitat Politècnica de València en España, con un perfil similar a los estudiantes del primer cuasi-experimento.

Los resultados de cada uno de los cuasi-experimentos individuales fueron recogidos utilizando boletines, un prototipo de configuración y un cuestionario post-experimento, los que se han detallado en la sección 7.5.1.4 Material Experimental.

7.5.3. Ejecución y análisis de los experimentos individuales

En esta sección se van a describir las características propias de cada uno de los experimentos que forman parte de la familia de experimentos.

7.5.3.1. Cuasi-experimento original (UPV1)

En esta sub-sección se describen los detalles relativos al experimento original. Para evitar redundancias, sólo se discutirán algunas aclaraciones del experimento original en relación a la información presentada en la sección 7.5.1.

El primer cuasi-experimento mencionado fue realizado en la Universidad Politécnica de Valencia en España, donde, después de la sesión de entrenamiento, la ejecución del cuasi-experimento fue realizada con 58 participantes. La ejecución fue controlada, no existieron interacciones significativas entre participantes. El experimento fue conducido en dos sesiones (una sesión con el grupo de la mañana y otra con el de la tarde). Cada sesión tuvo una duración de 90 minutos. Sin embargo, se permitió a los participantes finalizar el experimento incluso si el tiempo llegaba a su fin con el objetivo de mitigar el efecto techo. Las personas encargadas de realizar el experimento clarificaron cualquier duda o pregunta a lo largo de las sesiones experimentales. Después de las tareas experimentales, los participantes llenaron el cuestionario mostrado en la Tabla 7-1.

Para el análisis de los resultados, se usaron pruebas, estadística descriptiva y *box plots* para analizar los datos recogidos. Ya que ambos grupos de participantes tenían perfiles similares, combinamos los datos en un grupo. Los datos fueron analizados de acuerdo a las hipótesis establecidas. Los resultados fueron obtenidos usando SPSS v20 con un $\alpha = 0.05$.

Análisis de las Percepciones de Usuario

La Figura 7-6 muestra los diagramas de caja para cada variable de percepción en las cuales podemos ver que la media para cada variable es mayor que el valor neutro de la escala Likert, que es el 3.

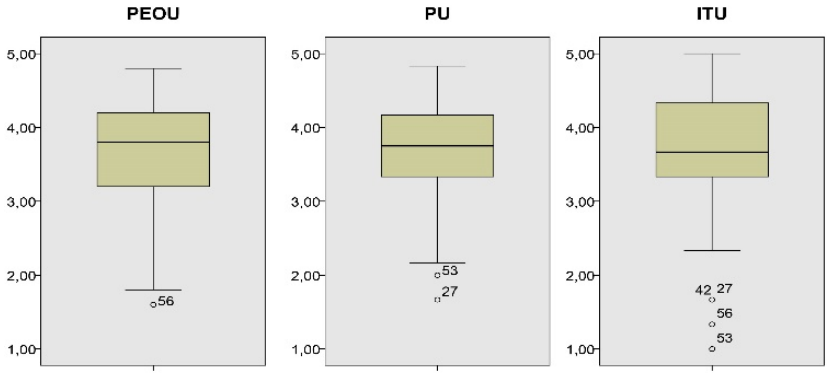


Figura 7-6 Diagrama de cajas para las variables PEOU, PU e ITU. Cuasi-Experimento 1- UPV1

Los diagramas de caja muestran algunos puntos de datos anómalos (participantes con id=27, 42, 53 y 56), los cuales corresponden a participantes que no asistieron a las sesiones del entrenamiento. Se han removido estos participantes para los análisis subsiguientes ya que no han seguido el protocolo experimental. Luego, se ha aplicado la prueba de Shapiro-Wilk para comprobar si los datos estaban distribuidos normalmente para seleccionar que test podría usarse para chequear las hipótesis H1, H2 y H3.

La Tabla 7-4 muestra los resultados de la prueba Shapiro-Wilk para las variables estudiadas. Se han aplicado pruebas para verificar las hipótesis comparando si la media de las respuestas a las preguntas relacionadas con una variable dada, fueron significativamente más altas que el valor neutro de la escala Likert.

Para las variables PU e ITU, las cuales tienen una distribución normal ($p>0.05$) se ha probado la hipótesis aplicando el t-test one-tailed mientras que para la variable PEOU, que no tiene una distribución normal ($p<0.05$), se ha aplicado el test Wilcoxon one-tailed one-sample con un valor de prueba igual a 3, ya que este valor corresponde al valor neutral de la escala Likert del cuestionario. Esos resultados nos permiten rechazar las hipótesis nulas $H1_0$, $H2_0$ y $H3_0$, lo que significa que los participantes perciben que el método Cloud MoS@RT es fácil de usar, útil y que ellos considerarían este método en el futuro si tuvieran que monitorizar la calidad de servicios cloud.

Tabla 7-4. Prueba de Shapiro Wilk para las variables subjetivas (UPV1)

Var	Min	Max	Mean	Std. Dev.	Std. E.	1-T. p-value	Shapiro-Wilk test p-value
PEOU	2.00	4.80	3.710	0.6978	0.09496	<0.001**	0.012*
PU	2.83	4.83	3.833	0.5005	0.06811	<0.001	0.302
ITU	2.33	5.00	3.809	0.6364	0.08661	<0.001	0.102

*The variable does not approach a normal distribution
** Results of the one-tailed one-sample Wilcoxon test

Análisis del Rendimiento del Usuario

Se ha medido la efectividad y la eficiencia de los participantes cuando utilizan Cloud MoS@RT en la práctica. La Tabla 7-5 presenta los valores de estadística descriptiva para las variables basadas en el rendimiento. Note que los valores atípicos identificados previamente han sido eliminados de este análisis. La efectividad total fue en promedio del 91.26%, indicando que casi todos los participantes fueron capaces de ejecutar la configuración de los requisitos de monitorización planteados correctamente.

La eficiencia ha sido calculada como el esfuerzo requerido (en minutos) para aplicar el método (Moody, 2001). Los resultados muestran que la eficiencia de los participantes fue de 14 a 56 minutos. Además, se debe tener en cuenta que la eficiencia puede variar considerablemente cuando se configura la monitorización de un servicio. Esto depende de muchos factores tales como la experiencia, la calidad de las especificaciones y el uso de las herramientas. A pesar de esas limitaciones, el propósito de este experimento fue probar si las percepciones de los usuarios fueron resultado de su rendimiento. Estos resultados además proporcionan información para entender cómo los participantes han usado el método.

Tabla 7-5. Estadística Descriptiva para Variables Basadas en la Percepción del Usuario (UPV1)

Variable	Min	Max	Media	Desviación Std
Efectividad	0.33	1.00	0.9126	0.1454
Eficiencia	14.00	56.00	26.6379	8.57

Análisis de las Relaciones Causales

El objetivo de esta sección es validar la parte estructural del MEM en términos de las relaciones causales entre sus constructores, con la excepción del Uso Actual. Para esto, se ha utilizado análisis de regresión para evaluar la operacionalización del MEM realizada, ya que las hipótesis a ser probadas son

relaciones causales entre variables continuas. Para realizar este análisis, hemos utilizado los siguientes niveles de significancia sugeridos por Moody (2001):

Tabla 7-6. Niveles de significancia (Moody, 2001)

Valor de Significancia	Rango
No significativo	p>0.1
Baja significancia	p<0.1
Media significancia	p<0.05
Alta significancia	p<0.01
Muy alta significancia	p<0.001

Eficiencia vs. Facilidad de Uso Percibida

La hipótesis H4 ha sido probada para comprobar si las percepciones de Facilidad de Uso Percibida (PEOU) son determinadas por la Eficiencia de los participantes cuando se aplica el método. Para ejecutar este análisis se ha construido un modelo de regresión simple en el cual la eficiencia fue usada como la variable independiente (predictora) y PEOU como la variable dependiente (predicha). La ecuación de regresión resultante del análisis es la siguiente:

$$PEOU = 4.044 + (-0.17) * Efficiency \quad (9)$$

Tabla 7-7. Regresión Simple entre la Eficiencia Actual y la Facilidad de Uso Percibida

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	4.044	0.344		11.76	<0.001		
Eficiencia	-.017	0.012	-.178	-1.36	0.181	0.178	0.032

El modelo de regresión fue encontrado no significativo, con p>0.1. El R^2 muestra que la variable eficiencia permite explicar solamente el 3.2% de la varianza en PEOU, indicando que la eficiencia actual de los participantes no influencia sus percepciones de facilidad de uso. Estos resultados no nos permiten rechazar la hipótesis nula $H4_0$ y aceptar su hipótesis alternativa, significando que hemos corroborado que la facilidad de uso percibida (PEOU) no está determinada por la Eficiencia.

Efectividad vs Utilidad Percibida

La hipótesis H5 ha sido probada para verificar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Efectividad de los participantes. Similarmente, nosotros construimos un modelo de regresión simple en el cual la Efectividad fue usada como la variable independiente

mientras que la PU fue usada como variable dependiente. La ecuación obtenida desde el modelo es la siguiente:

$$PU = 2.808 + 1.123 * Effectiveness \quad (10)$$

Tabla 7-8. Regresión Simple entre la Efectividad Actual y la Utilidad Percibida.

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	2.808	0.417		6.738	<0.001		
Efectividad	1.123	0.451	0.326	2.490	0.016	0.326	0.106

El modelo de regresión presenta una significancia media con $p<0.05$. El R^2 muestra que la Efectividad es capaz de explicar el 10.6% de varianza de PU, indicando que ciertas percepciones con respecto a PU están determinadas por la efectividad de los participantes cuando aplican el método. Como se esperaba, el coeficiente de regresión para la efectividad fue positivo, lo que significa que a valor más alto para la efectividad más alto el valor de la Utilidad Percibida. Esos resultados nos permiten rechazar $H5_0$ y aceptar su hipótesis alternativa, lo que significa que empíricamente se ha probado que PU está determinado por la efectividad.

PEOU vs Utilidad Percibida

La hipótesis H6 ha sido probada para verificar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Facilidad de Uso Percibida (PEOU). Similarmente, se construyó un modelo de regresión en el cual la variable PEOU fue usada como variable independiente mientras que PU fue usada como la variable dependiente.

La ecuación obtenida del modelo de regresión es:

$$PU = 2.739 + 0.294*PEOU \quad (11)$$

Tabla 7-9. Regresión Simple entre la Facilidad de Uso Percibida y la Utilidad Percibida

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	2.739	0.343		7.993	<0.001		
PEOU	0.294	0.091	0.411	3.247	0.002	0.411	0.169

Se ha encontrado que el modelo de regresión es altamente significativo con $p<0.01$. R^2 muestra que la variable Facilidad de Uso Percibida es capaz de explicar el 16.9% de la varianza en PU, indicando que ciertas percepciones con respecto a PU están determinadas por PEOU. Estos resultados nos permiten

rechazar $H6_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado empíricamente que PU está determinada por PEOU.

Intención de Uso Vs Utilidad Percibida

La hipótesis H7 ha sido probada para verificar si las percepciones de la Intención de Uso (ITU) están actualmente determinadas por la utilidad percibida (PU) cuando se aplica el método. La ejecución de este análisis fue construido a partir de un modelo simple de regresión en el cual la variable PU fue usada como variable independiente mientras ITU fue usada como variable dependiente, la ecuación obtenida del modelo es como sigue:

$$ITU = 0.053 + 0.849 * PU \quad (12)$$

Tabla 7-10 Regresión Simple entre Utilidad Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	0.553	0.507		1.090	<0.001		
PU	0.849	0.131	0.7	6.473	0.000	0.668	0.446

Mediante el modelo de regresión se encontró una alta significancia, con $p<0.001$. R^2 muestra que la variable PU es capaz de explicar el 44.6% de la varianza en ITU, lo cual representa un alto valor dado que podrían existir otros factores que influencien la intención de los participantes de usar un método. Estos resultados permiten rechazar $H7_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado que ITU está determinado por PU.

Intención de Uso vs. Facilidad de Uso Percibida

La hipótesis H8 ha sido probada para verificar si la Intención de Uso (ITU) está actualmente determinada por la Facilidad de Uso Percibida (PEOU). Similarmente, se construyó un modelo de regresión simple en el cual la variable PEOU fue usada como una variable independiente e ITU como una variable dependiente. La ecuación obtenida desde el modelo es la siguiente:

$$ITU= 2.304+0.405 * PEOU \quad (13)$$

Tabla 7-11 Regresión Simple entre Facilidad de Uso Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	2.304	0.428		5.381	<0.001		
PEOU	0.405	0.113	0.444	3.574	0.001	0.444	0.197

A través de este análisis se encontró una alta significancia con $p<0.01$. El R^2 muestra que la variable PEOU permite explicar el 19.7% de la varianza en ITU, indicando que las intenciones de los participantes para usar el método en el futuro están determinadas en cierta medida por su percepción en la facilidad de uso del método. Estos resultados permiten rechazar $H8_0$ y aceptar la hipótesis alternativa, lo que significa que se ha corroborado empíricamente que ITU está determinada por PEOU.

7.5.3.2. Cuasi-experimento2 (UNA)

La primera replicación del cuasi-experimento mencionado fue realizada en la Universidad Nacional de Asunción (UNA) en Paraguay, con un grupo de 14 profesionales que participaban en un Máster profesional en Ingeniería el Software en esta universidad.

Al igual que en el cuasi-experimento original, después de las sesiones de entrenamiento, su ejecución se realizó según estuvo prevista. La ejecución fue controlada, no existieron interacciones significativas entre participantes. El experimento fue conducido en una sesión. La sesión tuvo una duración de 90 minutos. Sin embargo, se permitió a los participantes finalizar el experimento incluso si el tiempo llegaba a su fin, esto para mitigar el efecto techo. La persona encargada de dirigir el experimento estuvo presta a clarificar cualquier pregunta a lo largo de la sesión experimental. Como se ha previsto para todos los experimentos, después de la ejecución de todas las tareas, los participantes llenaron el cuestionario mostrado en la Tabla 7-1.

Para el análisis de los resultados, se usaron pruebas, estadística descriptiva y *box plots* para analizar los datos recogidos. Ya que ambos grupos de participantes tenían perfiles similares, combinamos los datos en un grupo. Los datos fueron analizados de acuerdo a las hipótesis establecidas. Los resultados fueron obtenidos usando SPSS v20 con un $\alpha = 0.05$.

Análisis de las Percepciones de Usuario

La Figura 7-7 muestra los *box plots* para cada tipo de variable PEOU, PU e ITU en las cuales nosotros podemos ver que la media para cada variable es mayor que el valor neutral de la escala de Likert que es 3.

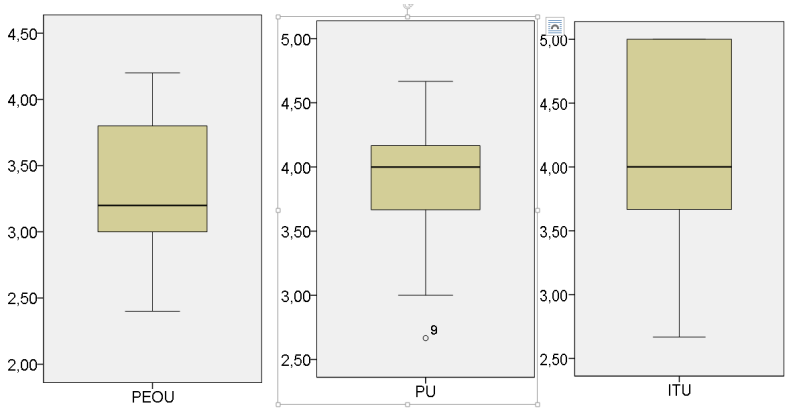


Figura 7-7 *Box-plots* para las variables PEOU, PU e ITU – Cuasi-Experimento 2 (UNA)

Los *box plots* muestran un punto de datos anómalos (participante con id=9), el cual corresponde a un participante que no estuvo durante las sesiones del entrenamiento. Se ha removido este participante para los análisis subsiguientes ya que no ha seguido el protocolo experimental. Luego, se ha aplicado la prueba de Shapiro-Wilk para comprobar si los datos estaban normalmente distribuidos para seleccionar que test podría usarse para verificar las hipótesis H1, H2 y H3.

La Tabla 7-4 muestra los resultados de la prueba Shapiro-Wilk para las variables que son estudiadas. Se han aplicado pruebas para verificar las hipótesis comparando si la media de las respuestas a las preguntas relacionadas con una variable dada, fueron significativamente más altas que el valor neutral Likert.

Las variables PU, PEOU e ITU han presentado una distribución normal ($p>0.05$) por lo que se ha probado la hipótesis aplicando el t-test. Esos resultados nos permiten rechazar las hipótesis nulas $H1_{0,}$ $H2_0$ y $H3_0$, lo que significa que los participantes perciben que Cloud MoS@RT es fácil de usar, útil y que ellos muestran una intención de uso futura cuando necesiten monitorizar servicios cloud.

Tabla 7-12. Prueba de Shapiro Wilk. Cuasi-Experimento 2 (UNA)

Var	Min	Max	Mean	Std. Dev.	Std. E.	1-T. p-value	Shapiro-Wilk test p-value
PEOU	2.40	4.20	3.3143	0.5067	0.13541	0.037	0.725
PU	2.67	4.67	3.9286	0.5536	0.14796	0.000	0.069
ITU	2.67	5.00	4.1429	0.7814	0.20882	0.000	0.107

Análisis del Rendimiento del Usuario

Nosotros medimos la efectividad de los participantes y su eficiencia cuando aplican Cloud MoS@RT en la práctica. Tabla 7-13 presenta los valores de estadística descriptiva para las variables basadas en el desempeño. Note que los valores atípicos identificados previamente han sido eliminados de este análisis. La efectividad total fue en promedio del 83.73%, indicando que casi todos los participantes fueron capaces de ejecutar la configuración del conjunto de los RNF correctamente.

Nosotros calculamos la eficiencia como el esfuerzo requerido (en minutos) para aplicar un método (Moody, 2001). Los resultados muestran que la eficiencia de los participantes fue de 45 a 90 minutos. Además se debe tener en cuenta que la eficiencia puede variar considerablemente cuando se configura la monitorización de un servicio. Esto depende de muchos factores tales como la experiencia, la calidad de las especificaciones y el uso de las herramientas. A pesar de esas limitaciones, el propósito de este experimento fue probar si las percepciones de los usuarios estuvieron dirigidas por su rendimiento. Estos resultados además proveen ciertas bases para entender cómo la gente usa el método.

Tabla 7-13. Estadística Descriptiva para Variables Basadas en la Percepción del Usuario. Cuasi-Experimento 2 (UNA)

Variable	Min	Max	Media	Desviación Std
Efectividad	0.28	1.00	0.8373	0.24016
Eficiencia	45.00	92.00	63.071	4.30112

Análisis de las Relaciones Causales

Al igual que en el primer cuasi-experimento, el objetivo de esta sección es validar la parte estructural del MEM en términos de las relaciones causales entre sus constructores, con la excepción del uso actual. Para esto, hemos seleccionado el uso de un análisis de regresión para evaluar nuestra operacionalización de MEM ya que las hipótesis a ser probadas son relaciones causales entre variables continuas. Nosotros hemos usado los niveles de significancia presentados por Moody (2001), los mismos que fueron mostrados con anterioridad en la Tabla 7-6

Eficiencia vs Facilidad de Uso Percibida

Nosotros probamos la hipótesis H4 para chequear si las percepciones de la Facilidad de Uso Percibida (PEOU) son determinadas por la eficiencia cuando

se aplica el método. Para ejecutar este análisis se ha construido un modelo de regresión simple en el cual la eficiencia fue usada como la variable independiente (predictor) y PEOU como la variable dependiente (predicho). La ecuación de regresión resultante del análisis es la siguiente:

$$PEOU = 3.159 + (0.02) * Efficiency \qquad (14)$$

Tabla 7-14. Regresión Simple entre la Eficiencia Actual y la Facilidad de Uso Percibida. Cuasi-Experimento 2 (UNA)

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	3.159	0.588		5.368	<0.001		
Eficiencia	0.002	0.009	0.078	0.272	0.791	0.078	0.006

El modelo de regresión fue encontrado no significativo, con p>0.1. El R^2 muestra que la variable eficiencia permite explicar solamente el 0.6% de la varianza en PEOU, indicando que la eficiencia actual de los participantes no influencia sus percepciones de facilidad de uso. Estos resultados no nos permiten rechazar la hipótesis nula $H4_0$ y aceptar su hipótesis alternativa, significando que hemos corroborado que la facilidad de uso percibida (PEOU) no está determinada por la Eficiencia.

Efectividad vs Utilidad Percibida

Hemos probado la hipótesis H5 para verificar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Efectividad de los participantes. Similarmente, nosotros construimos un modelo de regresión simple en el cual la Efectividad fue usada como la variable independiente mientras que la PU fue usada como variable dependiente. La ecuación obtenida desde el modelo es la siguiente:

$$PU = 3.697 + (0.277) * Effectiveness \qquad (15)$$

Tabla 7-15. Regresión Simple entre la Efectividad Actual y la Utilidad Percibida.

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	3.697	0.574		6.442	0.000		
Efectividad	0.277	0.661	0.120	0.419	0.682	0.120	0.014

El modelo de regresión no presenta significancia. El R^2 muestra que la efectividad es capaz de explicar el 1.4% de varianza de PU, indicando que la efectividad actual de los participantes no influye en la Utilidad Percibida (PU). Estos resultados no nos permiten rechazar $H5_0$ y aceptar su hipótesis alternativa.

Facilidad de Uso Percibida vs Utilidad Percibida

La hipótesis H6 ha sido probada para verificar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Facilidad de Uso Percibida (PEOU). Similarmente, se construyó un modelo de regresión en el cual la variable PEOU fue usada como variable independiente mientras que PU fue usada como la variable dependiente.

La ecuación obtenida del modelo de regresión es:

$$PU= 1.465 + 0.743 * PEOU \quad (16)$$

Tabla 7-16. Regresión Simple entre la Facilidad de Uso Percibida y la Utilidad Percibida

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	1.465	0.774		1.891	0.083		
PEOU	0.743	0.231	0.680	3.216	0.007	0.680	0.463

Se ha encontrado que el modelo de regresión es altamente significativo con p=0.007. R^2 muestra que la variable Facilidad de Uso Percibida es capaz de explicar el 46.3% de la varianza en PU, indicando que ciertas percepciones con respecto a PU están determinadas por PEOU. Estos resultados nos permiten rechazar $H6_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado empíricamente que PU está determinada por PEOU.

Intención de Uso Vs Utilidad Percibida

La hipótesis H7 ha sido probada para verificar si las percepciones de la Intención de Uso (ITU) están actualmente determinadas por la utilidad percibida (PU) cuando se aplica el método. La ejecución de este análisis fue construido a partir de un modelo simple de regresión en el cual la variable PU fue usada como variable independiente mientras ITU fue usada como variable dependiente, la ecuación obtenida del modelo es como sigue:

$$ITU = 1.124 + (-0.271) * PU \quad (17)$$

Tabla 7-17 Regresión Simple entre Utilidad Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	-0.271	0.978		-0.277	0.786		
PU	1.124	0.247	0.796	4.556	0.001	0.796	0.6336

Mediante el modelo de regresión se encontró una alta significancia, con p=0.001. R^2 muestra que la variable PU es capaz de explicar el 63.36% de la varianza en ITU, lo cual representa un alto valor dado que podrían existir otros

factores que influencien la intención de los participantes de usar un método. Estos resultados permiten rechazar $H7_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado que ITU está determinado por PU.

Intención de Uso vs. Facilidad de Uso Percibida

La hipótesis H8 ha sido probada para verificar si la Intención de Uso (ITU) está determinada por la Facilidad de Uso Percibida (PEOU). De manera similar, se construyó un modelo de regresión simple en el cual la variable PEOU fue usada como una variable independiente e ITU como una variable dependiente. La ecuación obtenida desde el modelo es la siguiente:

$$ITU=1.655+0.751*PEOU \quad (18)$$

Tabla 7-18 Regresión Simple entre Facilidad de Uso Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	1.655	1.303		1.271	0.228		
PEOU	0.751	0.389	0.487	1.930	0.078	0.487	0.237

El modelo de regresión se ha mostrado con una baja significancia. Por otra parte, R^2 muestra que la variable PEOU permite explicar el 23.70% de la varianza en ITU, indicando que las intenciones de los participantes para usar el método en el futuro donde se determinan las percepciones en la facilidad de uso del método. Estos resultados no permiten rechazar $H8_0$ y aceptar la hipótesis alternativa.

7.5.3.3. Cuasi-experimento 3 (UC)

La segunda replicación del cuasi-experimento original fue realizada en la Universidad de Cuenca (UC) en Ecuador, con un grupo de 10 estudiantes de la Facultad de Ingeniería que cursan el último año de la Carrera de Ingeniería de Sistemas. Como en los anteriores experimentos, después de la sesión de entrenamiento, la ejecución del cuasi-experimento fue realizada según estuvo prevista. La ejecución fue controlada y no existieron interacciones significativas entre participantes. El experimento fue conducido en una sesión de 90 minutos de duración. Sin embargo, se permitió a los participantes finalizar el experimento incluso si el tiempo llegaba a su fin, esto para mitigar el efecto techo. La persona encargada de dirigir el experimento estuvo presta a clarificar cualquier pregunta a lo largo de la sesión experimental. Como se ha previsto para todos los experimentos, después de la ejecución de todas las tareas, los participantes llenaron el cuestionario mostrado en la Tabla 7-1.

Para el análisis de los resultados, se usaron pruebas, estadística descriptiva y *box plots* para analizar los datos recogidos. Ya que ambos grupos de participantes tenían perfiles similares, combinamos los datos en un grupo. Los datos fueron analizados de acuerdo a las hipótesis establecidas. Los resultados fueron obtenidos usando SPSS v20 con un $\alpha = 0.05$.

Análisis de las Percepciones de Usuario

La Figura 7-8 muestra los *box plots* para cada tipo de variable PEOU, PU e ITU en las cuales nosotros podemos ver que la media para cada variable es mayor que el valor neutral de la escala de Likert que es 3.

Los *box plots* no muestran ningún punto de datos anómalos. A continuación, se ha aplicado la prueba de Shapiro-Wilk para chequear si los datos estaban normalmente distribuidos para seleccionar que test podría usarse para chequear las hipótesis H_1, H_2 y H_3.

La Tabla 7-19 muestra los resultados de la prueba Shapiro-Wilk para las variables que son estudiadas. Se han aplicado pruebas para verificar las hipótesis comparando si la media de las respuestas a las preguntas relacionadas con una variable dada, fueron significativamente más altas que el valor neutral Likert.

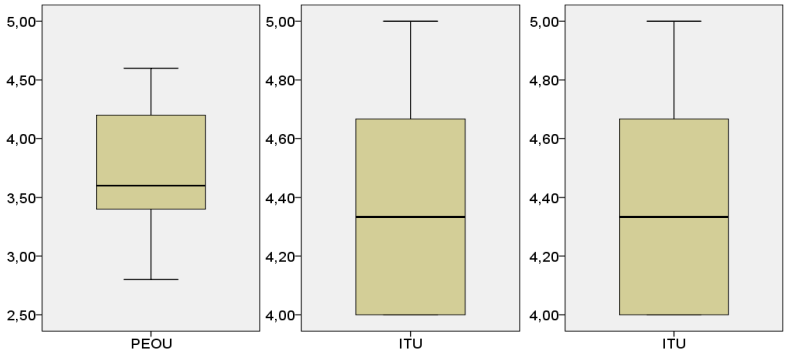


Figura 7-8 *Box-plots* para las variables PEOU, PU e ITU – Cuasi-Experimento 3 (UC).

Las variables PEOU y PU han presentado una distribución normal ($p>0.05$) por lo que se ha probado la hipótesis aplicando el t-test. Mientras que la variable ITU no presenta una distribución normal ($p<0.05$). Esos resultados nos permiten rechazar las hipótesis nulas $H1_0$, $H2_0$ y $H3_0$, lo que significa que los participantes perciben que Cloud MoS@RT es fácil de usar y útil y que ellos muestran su intención de usar este en el futuro cuando se monitoricen servicios cloud.

Tabla 7-19. Prueba de Shapiro Wilk. Cuasi-Experimento 3 (UC)

Var	Min	Max	Mean	Std. Dev.	Std. E.	1-t. p-value	Shapiro-Wilk test p-value
PEOU	2.80	4.60	3.7200	0.59777	0.18903	0.004	0.453
PU	3.00	5.00	4.2667	0.61963	0.19594	0.000	0.376
ITU	4.00	5.00	4.4000	0.40976	0.12958	0.005**	0.046*

*La variable no se aproxima a una distribución normal
**Resultados del test de Wilcoxon unilateral una muestra

Análisis del Rendimiento del Usuario

La Tabla 7-20 presenta los valores de estadística descriptiva para la efectividad y eficiencia de los participantes utilizando Cloud MoS@RT. La efectividad total fue en promedio del 83.34%, indicando que casi todos los participantes fueron capaces de ejecutar la configuración de los requisitos de monitorización correctamente.

Los resultados muestran que la eficiencia de los participantes fue de 17 a 30 minutos. Además se debe tener en cuenta que la eficiencia puede variar considerablemente cuando se configura la monitorización de un servicio. Esto depende de muchos factores tales como la experiencia, la calidad de las especificaciones y el uso de las herramientas. A pesar de esas limitaciones, el propósito de este experimento fue probar si las percepciones de los usuarios estuvieron dirigidas por su rendimiento. Estos resultados además proveen ciertas bases para entender cómo la gente usa el método.

En estos resultados, existe una variación importante de la eficiencia del método con respecto a las otras réplicas, esto se debe al tamaño de la muestra, dado que al ser el grupo más pequeño, las preguntas de los participantes fueron más rápidamente solventadas. Así también, al ser ya varias las ocasiones en las que se llevaron a cabo los experimentos, se pudo poner énfasis en ciertas explicaciones que mejoraron el rendimiento de los participantes.

Tabla 7-20. Estadística Descriptiva para Variables Basadas en la Percepción del Usuario. Cuasi-Experimento 3 (UC)

Variable	Min	Max	Media	Desviación Std
Efectividad	0.56	1.00	0.8334	0.16774
Eficiencia	17.00	30.00	23.300	4.27005

Análisis de las Relaciones Causales

El objetivo de esta sección es validar la parte estructural del MEM en términos de las relaciones causales entre sus constructores, con la excepción del Uso Actual. Para esto, hemos seleccionado el uso de un test de análisis de regresión para evaluar nuestra operacionalización del MEM ya que las hipótesis a ser probadas son relaciones causales entre variables continuas. Para esta evaluación, se ha utilizado los niveles de significancia propuestos por Moody (2001), que se presentan en la Tabla 7-6

Eficiencia vs. Facilidad de Uso Percibida

La hipótesis H4 ha sido probada para verificar si las percepciones de la Facilidad de Uso Percibida (PEOU) son determinadas por la eficiencia cuando se aplica el método. Para ejecutar este análisis se ha construido un modelo de regresión simple en el cual la eficiencia fue usada como la variable independiente (predictor) y PEOU como la variable dependiente (predicho). La ecuación de regresión resultante del análisis es la siguiente:

$$PEOU = 5.447 + (-0.074) * Efficiency \quad (19)$$

Tabla 7-21. Regresión Simple entre la Eficiencia Actual y la Facilidad de Uso Percibida. Cuasi-Experimento 3. Cuenca-Ecuador

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	5.447	0.993		5.484	0.001		
Eficiencia	-0.074	0.042	-0.529	-1.765	0.116	0.529	0.280

El modelo de regresión fue encontrado no significativo, con $p>0.1$. El R^2 muestra que la variable eficiencia permite explicar el 28.0% de la varianza en PEOU, indicando que la eficiencia actual de los participantes no influencia sus percepciones de facilidad de uso. Estos resultados no nos permiten rechazar la hipótesis nula $H4_0$ y aceptar su hipótesis alternativa, significando que hemos corroborado que la facilidad de uso percibida (PEOU) no está determinada por la eficiencia.

Efectividad vs. Utilidad Percibida

La hipótesis H5 ha sido probada para verificar si las percepciones de la Utilidad Percibida (PU) están determinadas por la efectividad de los participantes. Similarmente, nosotros construimos un modelo de regresión simple en el cual la efectividad fue usada como la variable independiente mientras que la PU fue usada como variable dependiente. La ecuación obtenida desde el modelo es la siguiente:

$$PU = 3.166 + (1.321) * \text{Effectiveness} \qquad (20)$$

Tabla 7-22. Regresión Simple entre la Efectividad Actual y la Utilidad Percibida.

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	3.166	1.035		3.059	0.016		
Efectividad	1.321	1.220	0.358	1.083	0.310	0.358	0.128

El modelo de regresión no presenta significancia. El R^2 muestra que la Efectividad es capaz de explicar el 1.28% de varianza de PU, indicando que la eficiencia actual de los participantes no influye en la Utilidad Percibida (PU). Estos resultados no nos permiten rechazar $H5_0$ y aceptar su hipótesis alternativa.

PEOU vs. Utilidad Percibida

La hipótesis H6 ha sido probada para verificar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Facilidad de Uso Percibida (PEOU). Similarmente, se construyó un modelo de regresión en el cual la variable PEOU fue usada como variable independiente mientras que PU fue usada como la variable dependiente.

La ecuación obtenida del modelo de regresión es:

$$PU = 2.401 + (0.502) * PEOU \qquad (21)$$

Tabla 7-23. Regresión Simple entre la Facilidad de Uso Percibida y la Utilidad Percibida

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	2.401	1.207		1.989	0.082		
PEOU	0.502	0.321	0.484	1.564	0.156	0.484	0.234

Se ha encontrado que el modelo de regresión no es significativo con $p=0.1564$. R^2 muestra que la variable Facilidad de Uso Percibida es capaz de explicar el 23.4% de la varianza en PU, indicando que ciertas percepciones con respecto a PU están determinadas por PEOU. Estos resultados no nos permiten rechazar $H6_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado empíricamente que PU no está determinada por PEOU.

Intención de Uso vs. Utilidad Percibida

La hipótesis H7 ha sido probada para verificar si las percepciones de la Intención de Uso (ITU) están actualmente determinadas por la utilidad percibida (PU) cuando se aplica el método. La ejecución de este análisis fue construido a partir de un modelo simple de regresión en el cual la variable PU fue usada como

variable independiente mientras ITU fue usada como variable dependiente, la ecuación obtenida del modelo es como sigue:

$$PU = 0.093 + (0.948) * ITU \quad (22)$$

Tabla 7-24 Regresión Simple entre Utilidad Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	0.093	1.839		0.051	0.961		
ITU	0.948	0.416	0.627	2.278	0.052	0.627	0.393

Mediante el modelo de regresión se encontró una baja significancia, con $p<0.1$. R^2 muestra que la variable PU es capaz de explicar el 39.3% de la varianza en ITU, lo cual representa un valor alto, sin embargo podrían existir otros factores que influencien la intención de los participantes de usar un método. Estos resultados no permiten rechazar $H7_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado que ITU no está determinado por PU.

Intención de Uso vs. Facilidad de Uso Percibida

La hipótesis H8 ha sido probada para verificar si la Intención de Uso (ITU) está determinada por la Facilidad de Uso Percibida (PEOU). De manera similar, se construyó un modelo de regresión simple en el cual la variable PEOU fue usada como una variable independiente e ITU como una variable dependiente. La ecuación obtenida desde el modelo es la siguiente:

$$ITU = 2.796+(0.431)* PEOU \quad (23)$$

Tabla 7-25 Regresión Simple entre Facilidad de Uso Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	2.796	0.709		3.944	0.004		
PEOU	0.431	0.188	0.629	2.289	0.051	0.629	0.396

En el modelo de regresión se muestra una baja significancia con $p<0.1$. R^2 muestra que la variable PEOU permite explicar el 39.60% de la varianza en ITU, indicando que las intenciones de los participantes para usar el método en el futuro donde se determinan las percepciones en la facilidad de uso del método. Estos resultados no permiten rechazar $H8_0$ y aceptar la hipótesis alternativa.

7.5.3.4. Cuasi-experimento 4 (UPV2)

La tercera replicación del cuasi-experimento original fue realizada en la Universidad Politécnica de Valencia en España, en donde, después de la sesión

de entrenamiento, la ejecución del cuasi-experimento fue realizada, según estuvo prevista, con 60 participantes; ésta replicación, permitirá evaluar la tarea de configuración del método, tras realizar un refinamiento y mejora del proceso de evaluación realizado en la réplica anterior.

La ejecución fue controlada, no existieron interacciones significativas entre participantes. El experimento fue conducido en dos sesiones (una sesión con el grupo de la mañana y otra con el de la tarde). Cada sesión tuvo una duración de 90 minutos. Sin embargo, se permitió a los participantes finalizar el experimento incluso si el tiempo llegaba a su fin, esto para mitigar el efecto techo. Las personas encargadas de dirigir el experimento clarificaron cualquier pregunta a lo largo de las sesiones experimentales. Después de las tareas experimentales, los participantes llenaron el cuestionario mostrado en la Tabla 7-1.

Para el análisis de los resultados, se usaron pruebas, estadística descriptiva y *box plots* para analizar los datos recogidos. Ya que ambos grupos de participantes tenían perfiles similares, combinamos los datos en un grupo. Los datos fueron analizados de acuerdo a las hipótesis establecidas. Los resultados fueron obtenidos usando SPSS v20 con un $\alpha = 0.05$.

Análisis de las Percepciones de Usuario

La Figura 7-9 muestra los *box plots* para cada tipo de variable PEOU, PU e ITU en las cuales nosotros podemos ver que la media para cada variable es mayor que el valor neutral de la escala de Likert que es 3.

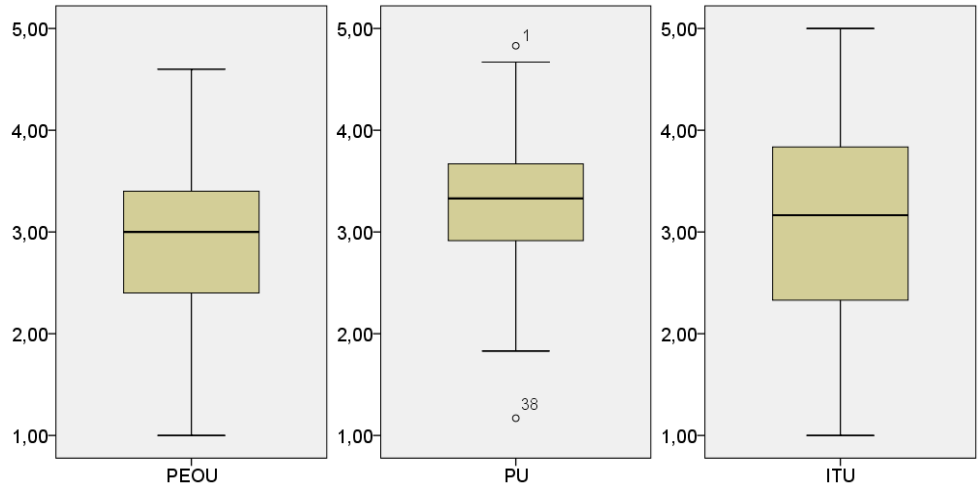


Figura 7-9 *Diagrama de cajas* para las variables PEOU, PU e ITU. Cuasi-Experimento 4- Valencia

Los *box plots* muestran algunos puntos de datos anómalos (participante con id=38), el cual corresponde a un participante que no participó en la sesión de entrenamiento. Se ha eliminado este participante para los análisis sub-siguientes, teniendo un total de 59 participantes. Por otra parte, no se ha creído conveniente eliminar al participante con id=1 (a pesar de que aparece como un dato anómalo en el diagrama de caja) ya que ha participado en la sesión de entrenamiento y todas las sesiones experimentales. Luego, se ha aplicado la prueba de Shapiro-Wilk para chequear si los datos estaban normalmente distribuidos para seleccionar que test podría usarse para verificar las hipótesis H1, H2 y H3.

La Tabla 7-26 muestra los resultados de la prueba Shapiro-Wilk para las variables que son estudiadas. Se han aplicado pruebas para verificar las hipótesis comparando si la media de las respuestas a las preguntas relacionadas con una variable dada, fueron significativamente más altas que el valor neutral Likert.

Para las variables PEOU, PU e ITU, las cuales tienen una distribución normal ($p>0.05$) se ha probado la hipótesis aplicando el t-test on-tailed se ha establecido un valor de prueba igual a tres que corresponde al puntaje neutral de Likert en el cuestionario. Esos resultados nos permiten rechazar la hipótesis nula $H2_0$, lo que significa que los participantes perciben que Cloud MoS@RT es útil, sin embargo ellos lo perciben como difícil de usar y no manifiestan su intención de usar este en el futuro cuando se monitoricen servicios cloud.

Tabla 7-26. Prueba de Shapiro Wilk. Experimento UPV2

Var	Min	Max	Mean	Std. Dev.	Std. E.	1-T. p-value	Shapiro-Wilk test p-value
PEOU	1.00	4.60	3.01	0.7877	0.10255	0.895	0.500
PU	1.83	4.83	3.299	0.6096	0.79362	0.000	0.519
ITU	1.00	5.00	3.169	0.9557	1.24418	0.179	0.219

Análisis del Rendimiento del Usuario

La Tabla 7-27 presenta los valores de estadística descriptiva para las variables basadas en el rendimiento de los participantes. Note que los valores atípicos identificados previamente han sido eliminados de este análisis. La efectividad total fue en promedio del 92.82%, indicando que casi todos los participantes fueron capaces de ejecutar la configuración del conjunto de los RNF correctamente.

Nosotros calculamos la eficiencia como el esfuerzo requerido (en minutos) para aplicar un método (Moody, 2001). Los resultados muestran que la eficiencia de los participantes fue de 19 a 80 minutos. Además se debe tener en cuenta que

la eficiencia puede variar considerablemente cuando se configura la monitorización de un servicio. Esto depende de muchos factores tales como la experiencia, la calidad de las especificaciones y el uso de las herramientas. A pesar de esas limitaciones, el propósito de este experimento fue probar si las percepciones de los usuarios estuvieron dirigidas por su rendimiento. Estos resultados además proveen ciertas bases para entender cómo la gente usa el método.

Tabla 7-27. Estadística Descriptiva para Variables Basadas en la Percepción del Usuario

Variable	Min	Max	Media	Desviación Std
Efectividad	0.46	1.00	0.9282	0.1404
Eficiencia	19.00	80.00	50.42	12.632

Análisis de las Relaciones Causales

El objetivo de esta sección es validar la parte estructural del MEM en términos de las relaciones causales entre sus constructores, con la excepción del uso actual. Para esto, hemos seleccionado el uso de un análisis de regresión para evaluar nuestra operacionalización de MEM ya que las hipótesis a ser probadas son relaciones causales entre variables continuas. Nosotros hemos usado los niveles de significancia presentados por Moody (2001), los mismos que fueron mostrados con anterioridad en la Tabla 7-6

Eficiencia vs Facilidad de Uso Percibida

La hipótesis H4 ha sido probada para verificar si las percepciones de la Facilidad de Uso Percibida (PEOU) son determinadas por la eficiencia cuando se aplica el método. Para ejecutar este análisis se ha construido un modelo de regresión simple en el cual la eficiencia fue usada como la variable independiente (predictor) y PEOU como la variable dependiente (predicho). La ecuación de regresión resultante del análisis es la siguiente:

$$PEOU=3.422+(-0.008)*Efficiency \quad (24)$$

Tabla 7-28. Regresión Simple entre la Eficiencia Actual y la Facilidad de Uso Percibida

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	3.422	0.425		8.043	<0.001		
Eficiencia	-0.008	0.008	-0.130	-0.989	0.327	0.130	0.017

El modelo de regresión fue encontrado no significativo, con $p>0.1$. El R^2 muestra que la variable eficiencia permite explicar solamente el 1.7% de la

varianza en PEOU, indicando que la eficiencia actual de los participantes no influencia sus percepciones de facilidad de uso. Estos resultados no nos permiten rechazar la hipótesis nula $H4_0$ y aceptar su hipótesis alternativa, significando que hemos corroborado que la facilidad de uso percibida (PEOU) no está determinada por la Eficiencia.

Efectividad vs Utilidad Percibida

La hipótesis H5 ha sido probada para verificar si las percepciones de Utilidad Percibida (PU) están determinadas por la Efectividad de los participantes. Similarmente, se ha construido un modelo de regresión simple en el cual la Efectividad fue usada como la variable independiente mientras que la PU fue usada como variable dependiente. La ecuación obtenida para el modelo es la siguiente:

$$PU=3.095+(0.220)*Effectiveness \quad (25)$$

Tabla 7-29. Regresión Simple entre la Efectividad Actual y la Utilidad Percibida.

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	3.095	0.699		4.426	<0.001		
Efectividad	0.220	0.748	0.039	0.295	0.016	0.039	0.002

El modelo de regresión presenta una significancia media con $p<0.05$. El R^2 muestra que la Efectividad es capaz de explicar el 0.2% de varianza de PU, indicando que ciertas percepciones con respecto a PU están determinadas por la efectividad de los participantes cuando aplican el método. Como se esperaba, el coeficiente de regresión para la efectividad fue positivo, lo que significa que el valor más alto para la efectividad más alto el valor de la Utilidad Percibida. Esos resultados nos permiten rechazar $H5_0$ y aceptar su hipótesis alternativa, lo que significa que empíricamente se ha probado que PU está determinado por la efectividad.

Facilidad de Uso Percibida vs Utilidad Percibida

La hipótesis H6 ha sido probada para verificar si las percepciones de Utilidad Percibida (PU) están determinadas por la Facilidad de Uso Percibida (PEOU). Similarmente, se construyó un modelo de regresión en el cual la variable PEOU fue usada como variable independiente mientras que PU fue usada como la variable dependiente.

La ecuación obtenida del modelo de regresión es:

$$PU=1.995+(0.433)*PEOU \quad (26)$$

Tabla 7-30. Regresión Simple entre la Facilidad de Uso Percibida y la Utilidad Percibida

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constante	1.995	0.265		7.541	0.000		
PEOU	0.433	0.085	0.559	5.095	0.000	0.559	0.313

Se ha encontrado que el modelo de regresión es altamente significativo con $p<0.01$. R^2 muestra que la variable Facilidad de Uso Percibida es capaz de explicar el 31.3% de la varianza en PU, indicando que ciertas percepciones con respecto a PU están determinadas por PEOU. Estos resultados nos permiten aceptar $H6_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado empíricamente que PU está determinada por PEOU.

Intención de Uso Vs Utilidad Percibida

La hipótesis H7 ha sido probada para verificar si las percepciones de la Intención de Uso (ITU) están actualmente determinadas por la utilidad percibida (PU) cuando se aplica el método. La ejecución de este análisis fue construido a partir de un modelo simple de regresión en el cual la variable PU fue usada como variable independiente mientras ITU fue usada como variable dependiente, la ecuación obtenida del modelo es como sigue:

$$ITU=-0.494+(1.110)*PU \quad (27)$$

Tabla 7-31 Regresión Simple entre Utilidad Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	-0.494	0.492		-1.004	0.320		
PU	1.110	0.147	0.708	7.572	0.000	0.668	0.446

Mediante el modelo de regresión se encontró una alta significancia, con $p<0.001$. R^2 muestra que la variable PU es capaz de explicar el 44.6% de la varianza en ITU, lo cual representa un alto valor; sin embargo, podrían existir otros factores que influencien la intención de los participantes de usar un método. Estos resultados permiten rechazar $H7_0$ y aceptar su hipótesis alternativa, lo que significa que hemos corroborado que ITU está determinado por PU.

Intención de Uso vs. Facilidad de Uso Percibida

La hipótesis H8 ha sido probada para verificar si la Intención de Uso (ITU) está determinada por la Facilidad de Uso Percibida (PEOU). Similarmente, se construyó un modelo de regresión simple en el cual la variable PEOU fue usada

como una variable independiente e ITU como una variable dependiente. La ecuación obtenida es la siguiente:

$$ITU=1.012+(0.716)*PEOU \qquad (28)$$

Tabla 7-32 Regresión Simple entre Facilidad de Uso Percibida e Intención de Uso

Reg. Element	Coef (b)	Std. E.	Std. Coef	t	Sig (p)	R	R2
Constant	1.012	0.404		2.506	0.015		
PEOU	0.716	0.130	0.590	5.516	0.000	0.590	0.348

A través de este análisis se encontró una alta significancia con $p<0.01$. R^2 muestra que la variable PEOU permite explicar el 34.8% de la varianza en ITU, indicando que la intención de los participantes para usar el método en el futuro está determinada por la percepción en la facilidad de uso del método. Estos resultados permiten rechazar $H8_0$ y aceptar la hipótesis alternativa, lo que significa que se ha corroborado empíricamente ITU está determinado por el PEOU.

7.5.3.5. Documentación y comunicación

Distintos aspectos relacionados con la documentación y a la comunicación entre los experimentadores pueden tener una influencia crucial en el éxito de la replicación de los experimentos (Shull *et al.*, 2004). En esta sección, se analizan posibles problemas relacionados con las deficiencias en la documentación y en los paquetes experimentales y como han sido mitigados en esta familia de cuasi-experimentos. Entre los principales problemas hallados por parte de los participantes en cuanto a la documentación, estuvo el listado de los parámetros dependientes de la plataforma, utilizado para que se formulen las métricas del primer y segundo escenario de recolección de datos; éste listado fue complejo de utilizar, dado que existían ciertos participantes no experimentados en la plataforma. Cabe señalar que no era propósito del experimento evaluar el conocimiento sobre parámetros específicos de la plataforma, de ahí que se propone solventar este problema por medio de una lista clasificada de acuerdo a los RNF a ser evaluados con una documentación específica relacionada a cada parámetro que ayude al usuario a seleccionar el más adecuado. En este contexto, los diseñadores de los experimentos se proponen mejorar los paquetes de laboratorio y el uso de herramientas de intercambio de información, siempre que no afecten el propósito de evaluación del experimento.

En cuanto a los cuestionarios y como se explica en las secciones de validez del constructo, ser realizó la prueba del alpha de Cronbach a fin de poder evaluar la confiabilidad del cuestionario. Estando dentro del umbral de confiabilidad. Como resultado de éste análisis se pudo concluir que las preguntas en el

cuestionario son confiables y las medidas de percepción son válidas en el modelo de evaluación.

Finalmente, para facilitar la replicación de los experimentos, se creó un sitio web que contiene todos los materiales experimentales del cuasi-experimento original. El sitio web describe, bajo una distribución lógica y fácilmente accesible, todos y cada uno de los documentos relacionados tanto con el entrenamiento como las sesiones experimentales.

7.5.4. Análisis de los resultados

En esta sección se discuten los resultados de los experimentos en resumen, con el fin de poder contrastar entre ellos y ver posibles semejanzas y diferencias. De ahí, se presenta cada una de las variables con los valores obtenidos, su media y desviación estándar.

La Tabla 7-33 muestra este resumen a fin de poder realizar las conclusiones pertinentes.

Se han utilizado la media y la desviación estándar como estadísticos descriptivos para las variables subjetivas cualitativas PEOU, PU e ITU. La media aritmética de la escala de Likert de cada una de las respuestas al cuestionario adoptada para la medición de las variables subjetivas, ha sido también considerada como una escala de intervalo (Carifio *et al.*, 2007). Los resultados globales nos han permitido concluir que Cloud MoS@RT ha mejorado el rendimiento de los participantes en prácticamente la totalidad de las estadísticas analizadas.

Tabla 7-33 Tabla Resumen Estadísticos Descriptivos

	UPV1		UNA		Universidad de Cuenca		UPV2	
	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.
Efectividad	0,9126	0,1454	0,8373	0,24016	0,8334	0,16774	0,9282	0,1404
Eficiencia	26,790	8,5700	63,071	4,30112	23,30	4,27005	50,42	12,632
Facilidad de Uso Percibida	3,7100	0,6978	3,3143	0,5067	3,72	0,18903	3,01	0,7877
Utilidad Percibida	3,8330	0,5005	3,9286	0,5536	4,2667	0,19594	3,299	0,6096
Intención de Uso	3,8090	0,6364	4,1429	0,7814	4,40	0,12958	3,169	0,9557

7.5.4.1. Eficiencia

La Figura 7-10 muestra los diagramas de caja para la variable eficiencia por participante y método para cada experimento de la familia. Esos diagramas muestran que bajo las condiciones de estos experimentos, la aplicación de Cloud MoS@RT va desde 20 hasta 60 minutos aproximadamente, para la configuración de 4 requisitos no funcionales a ser monitorizados, entre los cuales uno de ellos corresponde a una modificación, la misma que puede deberse a una renegociación del SLA o el cambio de métrica de un atributo a ser medido.

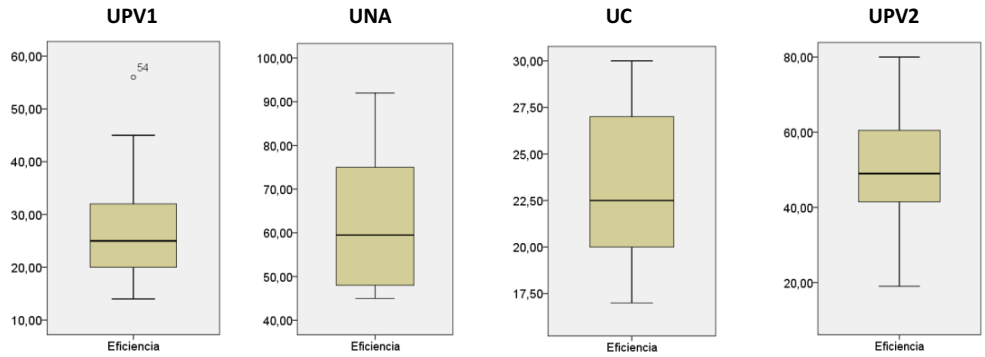


Figura 7-10 *Diagrama de cajas* de comparativas entre la eficiencia de los cuasi-experimentos efectuados.

7.5.4.2. Efectividad

La efectividad en cuanto a la configuración de los RNF planteados en los ejercicios del cuasi-experimento se muestran en la Figura 7-11. Como se puede observar en la misma, los participantes han sido capaces de responder efectivamente a las tareas propuestas, realizándolas en su mayoría de forma correcta. Esto se puede ver claramente en los *diagrama de cajas*.

Anteriormente además, se presentaron los resultados individuales de la efectividad de las tareas, demostrándose que dichas tareas tienen más de un 90% de efectividad, aún en los casos en los cuales la tarea se había descrito como difícil de realizar. Esto demuestra un muy buen resultado, ya que muestra una efectividad en la configuración que conlleva a un éxito en la correcta monitorización de los servicios, ya que ésta depende de una efectiva configuración de los RNF.

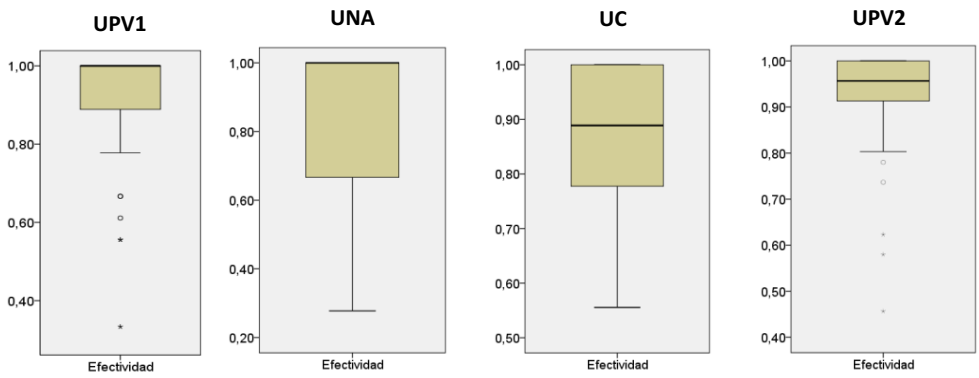


Figura 7-11 Diagrama de cajas de comparativas entre la efectividad de los cuasi-experimentos efectuados.

7.5.4.3. Facilidad de uso percibida (PEOU)

La Tabla 7-34 muestra un resumen de las medias de la facilidad de uso percibida en cada uno de los experimentos. En todos los casos Cloud MoS@RT fue percibido por los participantes como fácil de usar en relación con el valor de prueba v=3. Se puede además concluir, según ciertas experiencias expuestas por los participantes que se podría mejorar sustancialmente la facilidad de uso percibida, al momento de utilizar la herramienta definitiva de configuración de la monitorización, ya que por motivos de configuraciones y acceso se utilizó un prototipo que si bien realiza las acciones necesarias de configuración, no constituyen la herramienta definitiva de configuración de monitorización, presentando ciertas restricciones.

Tabla 7-34 Tabla Resumen Medias de la Facilidad de Uso Percibida

	UPV1	**UNA**	**UC**	**UPV2**
Facilidad de Uso Percibida	3,71	3,3143	3,72	3,01

Otra de las experiencias manifestadas por los participantes es la gran cantidad de parámetros propios de la plataforma que existen al momento de formular las métricas; como solución a este problema, en ambientes de producción definitivos, estos parámetros pueden ser mejor documentados y explicados de acuerdo a la plataforma que se vaya a utilizar.

Sin embargo, se puede decir que durante todos los experimentos realizados, bajo las circunstancias previstas de entrenamiento y posterior desarrollo del ejercicio, los participantes pudieron sin mayor inconveniente que el propio al enfrentarse a una nueva herramienta, llevar a cabo las tareas encomendadas, lo

que se ve reflejado no solamente en la variable de facilidad de uso percibida sino además en la variable de efectividad antes expuesta. Los resultados referentes a las pruebas de normalidad y de hipótesis han sido presentados en cada una de las réplicas de la sección anterior.

7.5.4.4. Utilidad percibida (PU)

La Tabla 7-35 muestra un resumen de las medias de la utilidad percibida en cada uno de los experimentos. En todos los casos Cloud MoS@RT fue percibido por los participantes como útil en relación con el valor de prueba v=3. En todos los experimentos realizados la utilidad percibida de Cloud MoS@RT se ha visto confirmada por las respuestas favorables al cuestionario de percepción.

Tabla 7-35 Tabla Resumen Medias de la Utilidad Percibida

	UPV1	UNA	Universidad de Cuenca	UPV2
Utilidad Percibida	3,833	3,9286	4,2667	3,299

Al igual que con la facilidad de uso percibida, los resultados referentes a las pruebas de normalidad y de hipótesis han sido presentados en cada una de las réplicas de la sección anterior.

7.5.4.5. Intención de uso (ITU)

La Tabla 7-36 muestra un resumen de las medias de la intención de uso en cada una de las réplicas de los experimentos. En todos los casos los usuarios tienen la intención de utilizar Cloud MoS@RT cuando necesiten un método de monitorización de servicios en relación con el valor de prueba v=3. En todos los experimentos realizados la intención de uso de Cloud MoS@RT se ha visto confirmada por las respuestas favorables al cuestionario de percepción.

Tabla 7-36 Tabla Resumen Medias de la Intención de Uso (ITU)

	UPV1	UNA	Universidad de Cuenca	UPV2
Intención de Uso	3,809	4,1429	4,4	3,169

7.5.4.6. Resumen de las hipótesis confirmadas

Una vez llevados a cabo los experimentos, se realizó un análisis global de los resultados para determinar si se había alcanzado el objetivo principal de la familia de experimentos. También se analizaron todos los resultados de los experimentos individuales en busca de diferencias. La Tabla 7-37 muestra un resumen de los resultados obtenidos en cada experimento individual.

Tabla 7-37 Resumen de los resultados de la familia de experimentos

Experimento	Tipo de Participantes	Número de Participantes	Hipótesis Rechazadas
UPV1	Estudiantes de último curso de Ingeniería Informática	58	$H1_0$, $H2_0$, $H3_0$, $H5_0$, $H6_0$, $H7_0$, $H8_0$
UNA	Estudiantes de Máster	14	$H1_0$, $H2_0$, $H3_0$, $H6_0$, $H7_0$
Universidad de Cuenca	Estudiantes de último curso de Ingeniería de Sistemas	10	$H1_0$, $H2_0$, $H3_0$, $H5_0$, $H6_0$, $H7_0$, $H8_0$
UPV2	Estudiantes de último curso de Ingeniería Informática	60	$H2_0$, $H5_0$, $H6_0$, $H7_0$, $H8_0$

Teniendo en cuenta los resultados obtenidos de los cuasi-experimentos, se concluye que para todos los participantes el método se percibe como útil, y aunque en la última réplica no se considera fácil de usar, la efectividad de los resultados demuestran que los participantes no han tenido problemas, ni dificultades al desarrollar los ejercicios.

Con respecto a la efectividad, se tienen valores muy altos que demuestran que el método funcionará adecuadamente, teniendo en cuenta que la configuración adecuada de los RNF y sus métricas es de vital importancia al momento de la generación del modelo en tiempo de ejecución, el cual es la base para la monitorización.

Finalmente, en cuanto a la eficiencia, el tiempo empleado para la configuración de los RNF es bastante aceptable (20 a 60 minutos) y puede ser optimizado a medida que se tenga mayor experticia en esta actividad, así como también cuando se tenga un conocimiento profundo de los contadores de la plataforma.

7.5.4.7. Discusión

Las siguientes conclusiones globales fueron obtenidas de cada pregunta de investigación:

RQ1: "¿Cloud MoS@RT es percibido como fácil de usar y útil? De ser así, ¿las percepciones de los usuarios son el resultado de su rendimiento cuando utilizan el método para configurar los requisitos de calidad a ser monitorizados?"

La mayoría de los participantes encontraron que Cloud MoS@RT es muy útil y fácil de usar al momento de ejecutar las tareas de configuración, las cuales están directamente relacionadas a la interacción del usuario. Esto es soportado por su efectividad cuando se relacionan directamente a la interacción del usuario. Esto

es corroborado por su efectividad cuando se ejecutan las tareas relacionadas a la configuración, efectividad que fue del 91.26%, 83.73%, 83.34% y 92.82%, respectivamente para cada cuasi-experimento realizado. Además se encontró que las hipótesis alternativas H1 relacionadas a las percepciones de los participantes sobre la facilidad de uso son aceptadas en la mayoría de los cuasi-experimentos (con excepción del último), rechazándose la hipótesis nula. Este resultado es alentador para continuar mejorando el método para que sea utilizado con mayor facilidad en contextos industriales para soportar la monitorización de atributos de calidad de alto nivel para SaaS. Por otro lado en todas las réplicas se rechazó la hipótesis $H2_0$, concluyéndose que para la mayoría de los participantes, el método se percibe como útil.

Con respecto a la influencia del rendimiento del usuario en sus percepciones, nosotros encontramos que las percepciones de la facilidad del uso no están determinadas por la eficiencia de los participantes (H4 no fue confirmada en ninguna de las réplicas). Una razón posible de esto podría ser que los participantes percibieron algunos problemas de usabilidad con el configurador de la monitorización. Algunos participantes mencionaron este hecho en sus respuestas a las preguntas abiertas del cuestionario. Sin embargo, en el momento en el que el experimento fue ejecutado, el configurador estaba en un estado temprano y éste ha sido sustancialmente mejorado desde ese momento. En general, los participantes proveyeron sugerencias útiles, las cuales han sido tomadas en cuenta para mejorar la herramienta.

Por otra parte, nuestros resultados indican que las percepciones y utilidad fueron determinadas por la efectividad de los participantes (H5) para la mayoría de los participantes en tres de las cuatro ejecuciones del cuasi-experimento. Una razón posible para esto podría ser el hecho de que el método de monitorización y su configurador guiaron a los participantes en especificar propiamente cómo una cláusula del SLA (expresada como NFR) puede ser mapeada en atributos de calidad específicos, métricas y como esas métricas pueden ser medidas en tiempo de ejecución utilizando contadores de rendimientos provistos por la plataforma cloud. Los participantes indicaron en el cuestionario que ellos encontraron el Modelo de Caridad SaaS muy útil para monitorizar requisitos de calidad específicos de la aplicación ya que este guía la definición de métricas e indicadores combinando distintas métricas de bajo nivel (e.j. *downtime*).

RQ2: "¿Existe una intención de uso de Cloud MoS@RT en el futuro? De ser así, ¿tales intenciones de uso es el resultado de las percepciones de los participantes?"

La mayoría de las respuestas fueron muy positivas sobre el uso de Cloud MoS@RT en el futuro, con una media de 3.8, 4.14, 4.4 y 3.16 para cada ejecución

del cuasi-experimento, con respecto al valor neutral 3, la cual además está soportada por las preguntas abiertas adjuntas en el cuestionario. Las hipótesis H3 fueron confirmadas en tres de las cuatro ejecuciones del cuasi-experimento, lo que significa que los participantes tienen la intención de usar Cloud MoS@RT en el futuro. Existe consciencia de que pueden existir otros factores que pueden afectar la decisión de la gente cuando se usa un método o herramienta de monitorización (p. ej. herramientas integradas a la plataforma, estándares de organización, licencias). Sin embargo, esos son factores que no es posible controlar. Nuestro objetivo aquí fue seleccionar variables que puedan ser controladas, tales como el comportamiento de los participantes utilizando un método de monitorización. Se considera la facilidad de uso percibida y la utilidad percibida porque son los factores más importantes con respecto al uso del sistema (F. Davis et al., 1989), (Moody, 2001).

El objetivo fue proveer bases que puedan ser usadas para trazar el impacto de las variables externas sobre las creencias, actitudes e intenciones internas. Los resultados proveen evidencia futura de la eficacia percibida de Cloud MoS@RT.

Con respecto a la influencia de las percepciones del usuario en sus intenciones, nosotros encontramos soporte para H6, H7 y H8, lo que significa que la percepción de los participantes sobre la facilidad de uso percibida y la utilidad determinan sus intenciones de usar Cloud MoS@RT en el futuro. Ciertas relaciones causales de los cuasi-experimentos UNA y UC, no se pudieron corroborar debido al bajo número de participantes que hace complejo establecer relaciones causales de manera más sólida.

Los resultados globales del análisis de regresión son resumidos en las Figuras 7.12-7.15. Estos hallazgos son consistentes con los resultados obtenidos por Moody (2001). Estos resultados constituyen una primera aproximación empírica de la evaluación del modelo propuesto. Como trabajo futuro, se ve la necesidad de investigar sobre la influencia de otras variables basadas en el rendimiento y en la percepción para predecir la aceptación de Cloud MoS@RT en la práctica. Y corroborar la eficacia de los resultados de monitorización y su intrusividad con los servicios monitorizados a fin de mejorar ciertos factores explotando los modelos en tiempo de ejecución.

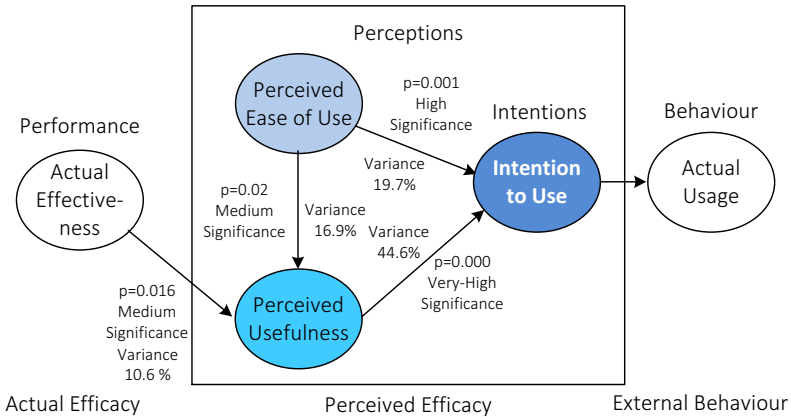


Figura 7-12 Conclusiones de la aplicación de MEM a Cloud MoS@RT – UPV1

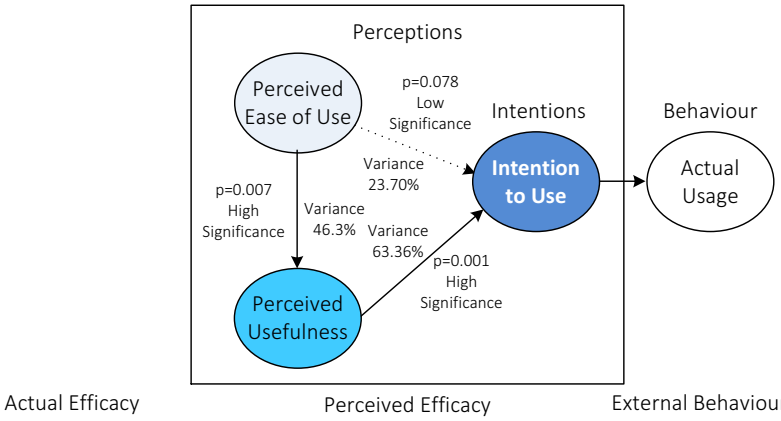


Figura 7-13 Conclusiones de la aplicación de MEM a Cloud MoS@RT – Réplica UNA

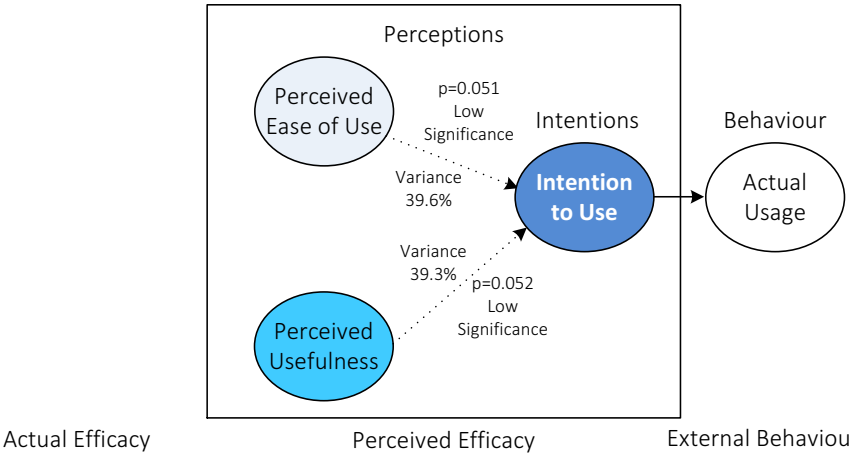


Figura 7-14 Conclusiones de la aplicación de MEM a Cloud MoS@RT – Réplica Universidad de Cuenca

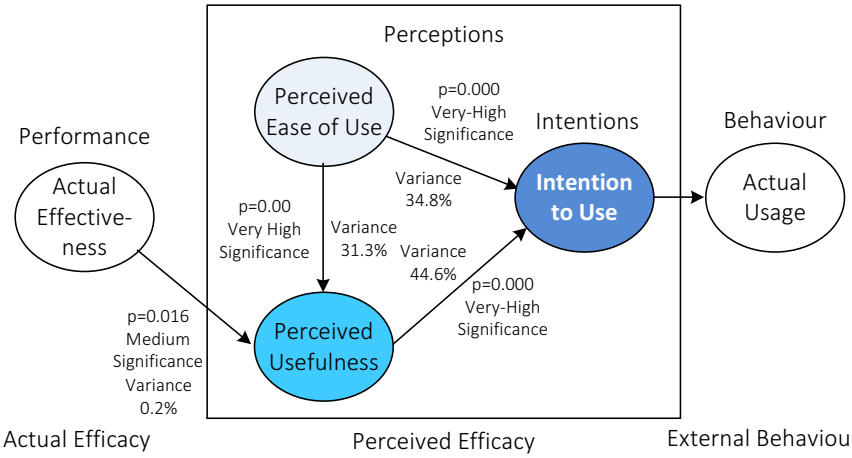


Figura 7-15 Conclusiones de la aplicación de MEM a Cloud MoS@RT – Réplica UPV2

7.6. Amenazas a la validez

En esta sección, se explican los principales problemas que pueden poner en peligro la validez de los cuasi-experimentos, al considerar los cuatro tipos de amenazas que proponen Cook y Campbell, (1979)

7.6.1. Validez interna

Las amenazas a la validez interna son relevantes en los estudios que intentan establecer relaciones causales. Las principales amenazas al a validez interna

fueron: la experiencia de los participantes, los sesgos del autor y los sesgos relacionados al método de monitorización y herramienta que lo soporta.

Para reducir la amenaza relacionada con la experiencia de los participantes, nosotros preparamos un ejemplo de entrenamiento representativo, el cual muestra cada paso del proceso y provee a los usuarios un alto entendimiento tanto de la monitorización de servicios cloud y cómo monitorizar la calidad de los servicios utilizando nuestro método.

Tanto los sesgos del autor como las producidas por la entendibilidad del material fueron reducidas durante la ejecución de un experimento piloto, en el cual un grupo de investigadores expertos en el área evaluaron el material experimental para así reducir posibles errores o malos entendidos relacionados con el experimento.

Los sesgos con respecto a la herramienta de monitorización fueron reducidos a través de su validación en el experimento piloto y pruebas sucesivas para mejorar la usabilidad de la herramienta.

7.6.2. Validez externa

Esta se refiere a la habilidad para generalizar los resultados en diferentes contextos. La principal amenaza a la validez externa es la representatividad de los resultados que puede verse afectada por el diseño de la evaluación, el contexto de participantes seleccionados y el tamaño y complejidad de las tareas experimentales.

El diseño de la evaluación puede tener un impacto en la generalización de los resultados debido a la complejidad de la plataforma cloud, sus características particulares, las herramientas usadas para recuperar los datos y los RNF a ser considerados. Nosotros intentamos reducir este problema seleccionando una plataforma popular y comúnmente utilizada, la cual comparta conceptos con otras plataformas, y considerando un escenario muy común desde el cual recoger datos crudos. Además, nosotros hemos seleccionado los RNF, los cuales son representativos para servicios cloud. Con respecto a la experiencia de los participantes, el cuasi-experimento fue conducido con alumnos de Ciencias de la Computación, Ingeniería de Sistemas y alumnos de máster en Ciencias de la Computación, quienes además en los cuatro casos asistieron al curso de Calidad de Software y tienen un buen conocimiento de modelos de calidad y métricas.

Además, ellos fueron entrenados en el uso de nuestro enfoque y herramienta durante una cantidad razonable de tiempo. Será, sin embargo, necesario ejecutar futuros experimentos con participantes de la industria. El tamaño y complejidad

de las tareas podría además afectar la validez externa. Para ello, hemos propuesto un conjunto de tareas experimentales con un nivel suficiente de complejidad, dado el tiempo que se tenía para las sesiones.

7.6.3. Validez del constructo

La principal amenaza en la validez del constructor es el cómo reflejar la eficacia en la manera en la que los NFR han sido configurados para ser monitorizados y si la monitorización refleja el estado real de los servicios ejecutándose en el cloud, proveyendo la evaluación de los servicios esperada. Las variables subjetivas están basadas en el *Method Evaluation Model* (MEM) (Moody, 2001), el cual como se ha indicado es un método muy bien conocido y validado empíricamente para la evaluación de tecnologías de la información. La principal amenaza es de ahí la confiabilidad del cuestionario. Para analizar la confiabilidad del cuestionario, un análisis de la prueba de confiabilidad del alfa de Cronbach fue realizado para cada conjunto de preguntas relacionadas a cada variable subjetiva. Esas fueron mayores al umbral mínimo aceptado $\alpha = 0.70$, donde el α Cronbach de la facilidad de uso percibida es 0.834, utilidad percibida es 0.776 e intención de uso es 0.820.

7.6.4. Validez de la conclusión

Las amenazas que afectan la validez de la conclusión se refieren a las conclusiones estadísticas. Ejemplos de estas son la elección de los métodos estadísticos, y la elección del tamaño de la muestra, entre otros.

Uno de los principales problemas de validez son los tamaños de la muestra en las diferentes réplicas del cuasi-experimento. Tal es el caso de la réplica realizada en la UC, en donde se toma una muestra de 10 participantes, pudiendo resultar en un problema de validez de la conclusión, dado que puede afectar el tema de causalidad entre las diferentes variables; sin embargo, los resultados fueron alentadores ya que los participantes lograron realizar exitosamente las tareas propuestas.

Para controlar el riesgo de la variación debido a diferencias individuales que se pueden hacer mayores debido al tratamiento, hemos seleccionado un grupo homogéneo de participantes. Y en cuanto a la recogida de datos, se aplicó el mismo procedimiento en cada experimento individual con el fin de extraer los datos, y se aseguró que cada variable dependiente se calculara mediante la misma función de medición.

7.7. Conclusiones

En esta sección hemos presentado un cuasi-experimento y tres replicaciones del mismo, lo que conforman una familia de experimentos, con la cual evaluamos la eficacia percibida de un grupo de estudiantes y profesionales utilizando Cloud MoS@RT para configurar la monitorización de un servicio en la nube y cambiar los requisitos de configuración.

El método usado para validar nuestro enfoque fue el *Method Evaluation Model*, el cual considera dos aspectos complementarios del éxito de un método: rendimiento actual y probabilidad de aceptación en la práctica. El MEM ha sido operacionalizado para evaluar métodos de monitorización de servicios. Básicamente se definieron variables basadas en el rendimiento (eficiencia y efectividad) como factores de influencia para las variables basadas en la percepción (facilidad de uso percibida, utilidad percibida e intención de uso).

El modelo teórico propuesto, que consiste en la extensión del MEM, ha sido utilizado en el diseño de una familia de cuatro cuasi-experimentos. Los resultados obtenidos del análisis de los datos revelan que: (i) la mayoría de los participantes han encontrado el método Cloud MoS@RT fácil de usar y útil; (ii) la mayoría de los participantes fueron muy positivos acerca del uso de Cloud MoS@RT en el futuro; (iii) el rendimiento de los participantes en los cuasi-experimentos determinó sus percepciones positivas; (iv) las percepciones determinan la intención de usar Cloud MoS@RT. En particular, los participantes han indicado en las preguntas abiertas del cuestionario, que les ha parecido útil el método ya que les ha guiado en la selección de métricas para medir cada cláusula del SLA.

Aunque los resultados son prometedores en cuanto a la utilidad del método para apoyar a los *stakeholders* en la configuración y modificación de los parámetros de la monitorización de aplicaciones SaaS, estos son preliminares y deben ser considerados con cautela, ya que se basan en la evaluación de un conjunto reducido de requisitos de monitorización, cláusulas del SLA y atributos de calidad. Además, el método ha sido aplicado en la evaluación de un único servicio en la nube de complejidad baja/media y solo ha sido utilizado por usuarios sin experiencia previa en la monitorización de calidad de servicios. Por lo tanto, es necesario la realización de otros experimentos que consideren un mayor número de atributos de calidad, evalúen otros servicios en la nube de complejidad media/alta e involucren a usuarios expertos, de preferencia a profesionales de la industria, en la monitorización de servicios con el método propuesto.

Asimismo, estos resultados no proporcionan evidencias sobre la utilidad del método de monitorización como un todo (únicamente para la fase de configuración de la monitorización, que es la fase donde interviene el usuario). Aunque creemos que esta fase es la más relevante, ya que es la más propensa a errores y también donde se genera el modelo en tiempo de ejecución, es necesario evaluar las demás fases del método Cloud MoS@RT relacionadas con las fases de monitorización y análisis. Esto se puede hacer comparando la infraestructura de monitorización con otras herramientas de monitorización existentes tales como Amazon CloudWatch.

Capítulo 8

Conclusiones y Trabajos Futuros

Este capítulo revisa los objetivos de la investigación, su nivel de cumplimiento y los principales hallazgos que han sido obtenidos de este trabajo. Finalmente, discute las contribuciones de esta investigación desde el punto de vista teórico, práctico y de las posibles líneas y oportunidades de investigación futura.

8.1. Conclusiones

A continuación se detalla la consecución de cada uno de los objetivos que han sido inicialmente planteados.

8.1.1. Objetivo general

El objetivo general de esta tesis es la *definición y evaluación empírica de un método para la monitorización de servicios desplegados en la nube, bajo el modelo SaaS, haciendo uso de modelos en tiempo de ejecución*.

Este objetivo se ha cumplido en su totalidad, dado que a lo largo de este trabajo, se ha planteado un método para la monitorización de servicios en la nube que utiliza modelos en tiempo de ejecución para solucionar varias limitaciones de las aproximaciones de monitorización existentes (falta de flexibilidad, dificultad para monitorizar atributos de calidad de alto nivel, etc.). Se ha definido un proceso de monitorización que apoya el método y guía a los usuarios en su utilización. Además, se ha diseñado e implementado una infraestructura de monitorización con un middleware independiente de plataforma que ha sido instanciado en la plataforma Microsoft Azure. Se han realizado también varios estudios empíricos para proporcionar evidencia sobre la utilidad del método para apoyar a stakeholders en la configuración de la monitorización de calidad de servicios y la detección de incumplimientos de SLAs.

Esta solución presenta varios beneficios comparada con otras aproximaciones de monitorización existentes:

1. La utilización de un Modelo de Calidad SaaS que sirva de soporte a los usuarios al momento de realizar el establecimiento y configuración de los requisitos no funcionales a ser monitorizados.
2. La posibilidad de monitorizar no solo atributos de calidad de bajo nivel (p. ej., uso de CPU), como es habitual en las aproximaciones existentes, sino que también atributos de más alto nivel como la fiabilidad o la seguridad, gracias al Modelo de Calidad SaaS que permite integrar y combinar métricas directas para obtener métricas indirectas.
3. La versatilidad de la infraestructura en la utilización de varios escenarios de captura de datos de monitorización (p. ej., herramientas de monitorización propias de la plataforma cloud, agentes, envoltorios) que permite recoger datos provenientes de distintas fuentes y operar con ellos.
4. La flexibilidad de la infraestructura de monitorización que no necesita ser re-implementada cada vez que un nuevo requisito no funcional deba monitorizarse, sino que únicamente consuma los nuevos parámetros y se adapte a las nuevas necesidades, todo esto gracias a los modelos en tiempo de ejecución.
5. La formalización de los artefactos que intervienen en la infraestructura de monitorización (p. ej., Modelo de Requisitos de Monitorización, Modelo de Calidad SaaS, Modelo de Calidad en Tiempo de Ejecución), que permite especificar los requisitos de monitorización y sus parámetros basándose en lenguajes de dominio específico propios y validados así como también en estándares que permitan una evaluación sistemática de la calidad de los servicios cloud.
6. La facilidad de especificar las métricas a ser empleadas durante la monitorización, las mismas que no se limitan a métricas pre-establecidas, sino que pueden variar de acuerdo al SLA o a la necesidad de las partes que intervienen en el aprovisionamiento del servicio.
7. La posibilidad de extender la infraestructura a posibles implementaciones futuras que permitan auto-adaptar la herramienta de monitorización al estado de los servicios.

Finalmente, para lograr el objetivo general, se plantearon ciertos objetivos específicos, que son analizados a continuación.

8.1.2. Objetivos específicos

Los objetivos específicos alcanzados se detallan a continuación.

8.1.2.1. Objetivo específico 1

Definir un método de monitorización basado en modelos en tiempo de ejecución que esté soportado por un proceso sistemático y repetible. El proceso describirá las actividades del método, así como también las tareas necesarias para llevar a cabo las actividades.

Este objetivo ha sido alcanzado mediante la definición de un método para la monitorización de servicios en la nube denominado Cloud MoS@RT. Este método explota modelos en tiempo de ejecución con la finalidad de proporcionar flexibilidad a la hora de cambiar o definir nuevos requisitos de monitorización.

El proceso que soporta el método de monitorización ha sido especificado en SPEM, lo que ha permitido modelar todos los artefactos de entrada y de salida del método, sus actividades principales y roles. Además, se describen las tareas específicas en las que se puede dividir cada una de las actividades principales, con el fin de que el método sea claramente entendido y pueda ser instanciado a una plataforma específica y utilizado por los stakeholders a la hora de configurar la monitorización de los servicios.

Se han planificado actividades que permitan hacer frente a la monitorización de servicios dentro de un proceso que utilice como artefactos de entrada los requisitos no funcionales del usuario del servicio, de una manera estandarizada, ordenada y fácilmente entendible. Esta forma de representación de requisitos es de gran utilidad al momento de especificar características propias del servicio, ya que los requisitos pueden estar expresados de varias maneras incluyendo lenguaje natural, para lo cual hace falta una manera estandarizada de representación que sea entendible por el ordenador. Para ello, se ha creado el *Modelo de Requisitos de Monitorización*, el mismo que ha sido diseñado en base a los elementos de un SLA y que ayuda a expresar las necesidades y requisitos de monitorización haciendo uso del lenguaje de dominio específico WSLA.

Por otro lado, el proceso también incluye la utilización del *Modelo de Calidad SaaS* que descompone las características de calidad del servicio en sub-características, atributos y métricas que permiten evaluar la calidad del servicio. Este modelo sirve como soporte a la hora de definir las métricas de monitorización. Para la elaboración de este modelo, se ha utilizado el modelo de calidad de software propuesto en el estándar ISO/IEC 25010, que ha sido extendido con atributos y métricas propias de las plataformas en la nube, con énfasis en SaaS. Esta descomposición se ha realizado solo para algunas

características de calidad como el rendimiento y la fiabilidad, ya que el objetivo no era definir un modelo de calidad completo sino el método de monitorización y la infraestructura que lo soporta. Sin embargo, en un trabajo reciente desarrollado en el contexto del proyecto Value@Cloud, Navas (2016) ha propuesto un Modelo de Calidad para Servicios Cloud que ha sido construido a partir de 178 atributos y 364 métricas para servicios cloud obtenidas como resultado de una revisión sistemática de la literatura. Este trabajo permitirá extender el Modelo de Calidad SaaS propuesto en esta tesis. De hecho, este es uno de los trabajos que se está realizando actualmente.

Finalmente, se ha establecido el diseño del modelo en tiempo de ejecución, el cual además de contener un sub-conjunto de elementos del Modelo de Requisitos de Monitorización y del Modelo de Calidad SaaS, incluye también características propias del servicio que permiten la monitorización del servicio en tiempo de ejecución.

Como se puede apreciar, todos los modelos han sido definidos en base a estándares o lenguajes específicos de dominio ampliamente aceptados, que han permitido establecer de una manera clara y repetible cada una de las necesidades de monitorización.

Una de las limitaciones del método es que no realiza una retroalimentación desde el servicio hacia la infraestructura de monitorización para conseguir un refinamiento automático de los parámetros de monitorización basado en el estado del servicio. Sin embargo, como esta propuesta utiliza modelos en tiempo de ejecución, y la autoadaptabilidad es una de sus características, nuestra solución vislumbra esta autoadaptabilidad como trabajo futuro.

8.1.2.2. Objetivo específico 2

Definir una infraestructura de monitorización cuyo diseño sea independiente de la plataforma cloud, que permita realizar la monitorización de servicios en la nube y la detección de incumplimientos del SLA. La infraestructura deberá hacer uso de modelos en tiempo de ejecución para proporcionar la flexibilidad e interoperabilidad deseables, permitir la especificación de los requisitos no funcionales a partir de acuerdos de nivel de servicios (SLA) y estar alineada con estándares de calidad.

Para cumplir este objetivo, se ha definido una infraestructura de monitorización independiente de la plataforma, diseñada para soportar el método de monitorización propuesto.

Esta solución contiene dos componentes principales, que permiten primero *configurar* los requisitos de monitorización y luego *monitorizar* dichos requisitos. La solución interactúa con los servicios cloud para evaluar su calidad, a través de

la recolección de información desde diferentes fuentes de datos, para lo cual se han establecido escenarios claramente diferenciados y que ayudan a explotar las bondades de las plataformas, operar con datos existentes, adaptar y capturar datos especiales de los servicios a través de envoltorios e incluso permitir la captura de información desde otras soluciones de monitorización.

Adicionalmente, el planteamiento de esta infraestructura, que hace uso de los modelos en tiempo de ejecución, permite que la solución provea el grado de flexibilidad necesario a la hora de definir nuevos RNF a ser monitorizados; ya sea por cuestiones de renegociación de SLA como también por nuevas necesidades de monitorización presentadas por cualquiera de las partes que intervienen en el aprovisionamiento del servicio.

Vale la pena señalar que tras las evaluaciones empíricas, los modelos se han ido refinando con el fin de lograr una mejora en la infraestructura de monitorización, así como también un mejor facilidad de uso por parte del usuario. Sin embargo, queda por comprobar si este diseño es lo suficientemente genérico para ser instanciado también en otras plataformas cloud como Amazon Web Services.

8.1.2.3. Objetivo específico 3

Instanciar e implementar la infraestructura de monitorización en una plataforma cloud específica para realizar monitorizaciones de servicios desplegados en la nube y observar la aplicabilidad y facilidad de adaptación de la infraestructura propuesta.

La instanciación de la solución se ha conseguido y mostrado a lo largo Capítulo 6 de este trabajo. En dicho capítulo se ilustra tanto la instanciación del método de monitorización aplicado a un caso de estudio, así como la instanciación del diseño de la infraestructura de monitorización propuesta en el Capítulo 5, llevada a un escenario real y en una plataforma específica.

Durante la instanciación e implementación de la infraestructura se han presentado una serie de dificultades tecnológicas (p. ej., la utilización del formato XMI en .NET, la navegabilidad de los modelos al momento de la programación de la herramienta), que se han solventado de una manera exitosa, demostrando que el método puede ser implementado con las herramientas adecuadas y que las principales dificultades no responden cuestiones del diseño del método o la infraestructura, sino más bien a temas relacionados con el manejo de la plataforma y las tecnologías. La herramienta desarrollada por Jimenez (2015) ha verificado la vialidad de todo el proceso en Microsoft Azure.

Como se ha observado también, la mayoría de herramientas de monitorización actuales centran sus esfuerzos en obtener los datos de bajo nivel

de las plataformas y servicios desplegados. Siendo éste un esfuerzo muy necesario; de ahí que este trabajo ha demostrado la utilidad de interponer una capa más entre los datos crudos y los intérpretes de los mismos para gestionar el ingente volumen de datos obtenidos, aumentando así la capacidad de interpretar esta información y tomar decisiones que mejoren la calidad del servicio. Así, la infraestructura planteada no considera competidoras al resto de herramientas comerciales de monitorización sino aliadas, dado que éstas permiten conseguir datos de calidad relevantes. Los mecanismos de recolección de información han sido definidos teniendo en cuenta distintos escenarios de captura de información, los mismos que proporcionan al middleware de monitorización la capacidad de utilizar librerías y herramientas de monitorización propias de la plataforma, la posibilidad de definir funciones de medición personalizadas para medir la calidad de los servicios y la posibilidad de extender los servicios para que provean de datos que puedan ser utilizados por los intérpretes.

Si bien en esta tesis se muestra la implementación de la solución en Microsoft Azure, existen también implementaciones adicionales del configurador y del middleware de monitorización para la plataforma Google App Engine, que han sido realizadas por López (2016) y Páez (2016) como parte de sus trabajos de Fin de Grado en el contexto del proyecto Value@Cloud. Estas implementaciones permiten corroborar de una manera fehaciente la factibilidad de instanciar el diseño de la infraestructura de monitorización en otra plataforma. En definitiva, las implementaciones del middleware en dos plataformas distintas nos han permitido evaluar la utilidad de Cloud MoS@RT en soportar la monitorización de servicios desplegados en la nube.

Sin embargo, los prototipos implementados cuentan con algunas limitaciones y margen de mejora en lo que a usabilidad se refiere. En particular, el interfaz de usuario que muestra los datos de monitorización y detección de violaciones del SLA es algo rudimentario y aun no proporciona mecanismos y sugerencias de mejora. Como trabajo futuro, se espera construir un dashboard interactivo que mejore la experiencia de usuario.

Otra limitación es la de establecer una comunicación fluída con otras herramientas de monitorización (p. ej., herramientas de monitorización de propósito general, agentes) a través de conectores genéricos, ya que si bien el sistema plantea la interacción con otras soluciones de monitorización, aún queda por establecer los mecanismos de comunicación específicos para compartir datos de monitorización.

8.1.2.4. Objetivo Específico 4

Evaluar empíricamente el método de monitorización propuesto. Esta evaluación proveerá una retroalimentación para la mejora de la propuesta.

En cuanto a la evaluación empírica del método de monitorización, se ha ejecutado una familia de cuasi-experimentos que analiza la eficacia del método y miden la percepción del usuario (la facilidad de uso percibida, utilidad percibida e intención de uso). Tras la aplicación de este método y con los resultados obtenidos se ha encontrado que los usuarios perciben a Cloud MoS@RT como fácil de usar y útil. Además, la mayoría de los participantes han expresado su intención de uso de este método en un futuro, en caso de necesitar utilizar un método de monitorización de servicios.

Los usuarios que han realizado esta evaluación corresponden a dos perfiles, el primero son estudiantes en formación con conocimientos recientes en cuanto a las nuevas tecnologías y que han cursado de manera reciente la asignatura de Calidad de Software. Por otra parte, están estudiantes de máster quienes constituyen profesionales ya formados, quienes aplican diariamente las tecnologías de la información para realizar su trabajo, por lo que pueden ser capaces de entender la implicación industrial de este tipo de herramientas.

Las limitaciones en cuanto a la evaluación radican principalmente en que muchos de los participantes, si bien tienen la formación necesaria en tecnologías de la información, no trabajan cotidianamente con servicios desplegados en la nube, por lo cual un trabajo futuro, sería replicar los experimentos realizados en entornos industriales. Otra limitación de la evaluación realizada es que ha sido centrada en una fase del método de monitorización, siendo ésta la configuración. Otro tema necesario de evaluar es la infraestructura de monitorización en cada una de sus fases (configuración, monitorización y análisis) con el fin de validar el uso del modelo en tiempo de ejecución, afinar los tiempos de captura de información y otros aspectos como sobrecarga. Estos temas serán abordados en trabajos futuros.

También es necesario replicar los experimentos realizados considerando objetos experimentales con más atributos y métricas de calidad tanto de alto como de bajo nivel, que permitan determinar posibles limitaciones de la solución, empleando SLA complejos y más representativos. Finalmente, es necesario establecer evaluaciones para servicios más complejos, en entornos multi-cloud, que necesiten interacción con otros servicios y que constituyan casos reales en la industria.

8.2. Resumen de las principales contribuciones

Esta tesis ha contribuido con una solución al problema de la monitorización de calidad de los servicios en la nube y detección de incumplimientos del SLA. De entre las contribuciones más importantes se destacan las siguientes:

- **El método Cloud MoS@RT**. Se ha propuesto un método de monitorización de servicios cloud que hace uso de modelos en tiempo de ejecución para proporcionar flexibilidad ante cambios y re-negociaciones del SLA y monitorización de los RNF adicionales. Para ello, se han planteado tres actividades principales: la configuración de la monitorización, la monitorización en sí misma y el análisis de los resultados. Además, se han mostrado, dentro de cada una de las actividades, las tareas involucradas, con el fin de lograr una monitorización efectiva de un servicio cloud.

- **La infraestructura de monitorización.** Se ha planteado una infraestructura de monitorización con sus componentes básicos (configurador y monitor) que permitan soportar el método de monitorización haciendo uso de los modelos en tiempo de ejecución. En particular, se ha propuesto: i) un *Modelo de Requisitos de Monitorización* que soporta la especificación de RNF acorde a lenguajes de dominio específico que contribuye a la automatización de la verificación del cumplimiento de los SLA; ii) un *Modelo de calidad SaaS* que es utilizado por la infraestructura de monitorización con la finalidad de ejecutar una configuración efectiva de los RNF; iii) un *Modelo de Calidad en Tiempo de Ejecución* que relaciona los requisitos no funcionales a ser monitorizados con los datos en crudo que hacen posible la monitorización de estos requisitos de alto nivel.

- **La instanciación del método y de la infraestructura a una plataforma específica.** Se ha diseñado e implementado un prototipo de la infraestructura de monitorización para demostrar la factibilidad de la instanciación de la infraestructura a una plataforma específica. Para esta instanciación se ha seleccionado la plataforma Microsoft Azure y el trabajo ha sido realizado como parte de un trabajo de fin de grado (Jimenez, 2015), contribución que ha ido de la mano con el presente trabajo y en la que se ha utilizado el diseño planteado a lo largo de esta tesis.

- **Modelo teórico para evaluar métodos de monitorización.** Para evaluar el método Cloud MoS@RT e infraestructura de soporte se ha realizado una operacionalización del MEM (Moody, 2001) para evaluar métodos de monitorización de servicios. Esta operacionalización ha consistido en la definición de un modelo teórico (con distintos constructores) para evaluar

métodos de monitorización basados en la percepción de los usuarios y la definición de un instrumento que permite medir estos constructores. Este modelo teórico puede ser utilizado para diseñar i) cuasi-experimentos que permitan evaluar cualquier método de monitorización con el objetivo de predecir su aceptación en la práctica o ii) experimentos controlados que permitan comparar dos o más métodos de monitorización para determinar cual es el más eficiente, efectivo y con más probabilidad de ser aceptado en la práctica. Desde el punto de vista investigador, el modelo teórico ha sido útil para proporcionar una base o teoría sobre la cual basar el diseño de los cuasi-experimentos realizados en esta tesis. Desde un punto de vista práctico, aunque las relaciones causales del MEM (parte estructural del modelo) han sido confirmadas en los cuasi-experimentos realizados, el modelo teórico debe ser aplicado en el diseño de otros cuasi-experimentos más complejos y también en el diseño de experimentos controlados para comparar métodos de monitorización de servicios.

- **Evaluación empírica:** El método e infraestructura propuestos han sido evaluados mediante la realización de cuatro cuasi-experimentos realizados con usuarios en España, Paraguay y Ecuador. Estos estudios permitieron evaluar la utilidad del método de monitorización basado en la percepción de los usuarios. Desde el punto de vista investigador, los cuasi-experimentos nos ha permitido, analizar la aplicabilidad del proceso de configuración de la monitorización, modelar un caso real y analizar el grado de completitud y corrección de las soluciones aportadas por el método e infraestructura. Por último, pero no menos importante, obtener realimentación de los participantes que aplicaron el método, cuyos comentarios nos permitirán mejorar en el futuro tanto el método como la infraestructura de soporte al método. Desde el punto de vista práctico, este es un primer estudio con un número algo reducido de participantes, con usuarios noveles y que solo arroja resultados preliminares sobre la utilidad, facilidad de uso e intención de uso del método Cloud MoS@RT. En el futuro será necesario replicar este cuasi-experimento con usuarios expertos, monitorizando servicios más complejos y con mayor número de participantes para analizar posibles interacciones de factores como la experiencia o la habilidad.

En conclusión, los resultados de esta tesis son útiles desde un punto de vista teórico debido al hecho de que varias tecnologías han sido integradas en una solución concreta de monitorización: estándares ISO, lenguajes de dominio específico para SLA, modelos en tiempo de ejecución, servicios de monitorización de plataformas cloud específicas, etc. Desde un punto de vista práctico, somos conscientes que el método e infraestructura de monitorización

tienen que ser utilizados en entornos reales (cloud y multi-cloud), con participantes con más experiencia en monitorización en la nube y con servicios de mayor complejidad.

8.3. Difusión de resultados

El trabajo descrito en esta tesis han sido publicados en dos conferencias internacionales catalogadas de Nivel A según el ranking ERA-CORE, siendo una de ellas en la conferencia más relevante en el área de computación de servicios (SCC). Además, se ha realizado una publicación en un taller internacional de referencia en el tema de los modelos en tiempo de ejecución, una publicación en la revista ERCIM News, una publicación en las Jornadas Nacionales de Ciencia e Ingeniería de los Servicios (JCIS) así como otras dos publicaciones en conferencias internacionales que han servido para comprender los conceptos de servicios, modelos de calidad y características de calidad. Finalmente, se ha enviado recientemente un artículo a la revista *Information Sciences* con los resultados del diseño y evaluación de la infraestructura de monitorización. A continuación, se describe estas publicaciones con mayor detalle:

- P. Cedillo, J. Jimenez-Gomez, S. Abrahão, and E. Insfran, "Towards a Monitoring Middleware for Cloud Services," *9th IEEE International Conference on Services Computing (SCC 2015)*, June 27-July 2, 2015, New York, USA, pp. 451-458, IEEE Press. ERA CORE Tier A.

- P. Cedillo, J. Gonzalez-Huerta, S. Abrahão, and E. Insfrán, "A Monitoring Infrastructure for the Quality Assessment of Cloud Services," *24th International Conference on Information Systems Development (ISD 2015*), Model-Driven Development and Concepts Track, August 25 - 27, 2015, Harbin, China, pp. 17-32, Springer. ERA CORE Tier A.

- Cedillo, P., Gonzalez-Huerta, J., Insfrán, E., & Abrahão, S. Towards Monitoring Cloud Services Using Models@run time. *9th International Workshop on Models at run.time (MRT 2014)*, co-located with the ACM/IEEE 17th International Conference on Model Driven Engineering Language and Systems (MODELS 2014), Valencia, Spain, pp. 31–40.

- M. A. Zúñiga-Prieto, P. Cedillo, J. González-Huerta, E. Insfrán, and S. Abrahão, "Monitoring Services Quality in the Cloud," *ERCIM News*, no. 99, *Special Theme on Software Quality*, October 2014, pp. 19–20.

- P. Cedillo, J. Jimenez-Gomez, S. Abrahão, and E. Insfrán, "Definición de Mecanismos Personalizados de Monitorización de Servicios Cloud," *XI*

Jornadas de Ciencia e Ingenieria de Servicios (JCIS 2015), Santander, 15-17 Septiembre 2015.

Publicaciones Relacionadas:

- P. Cedillo, A. Fernandez, A. Insfran, and S. Abrahão, "Quality of Web Mashups: A Systematic Mapping Study," *4th International Workshop on Quality in Web Engineering (QWE 2013)*, co-located with the 13th International Conference on Web Engineering (ICWE), Aalborg, Denmark, LNCS 8295, Springer, pp. 66–78, 2013.
- E. Insfran, P. Cedillo, A. Fernandez, S. Abrahão, and M. Matera, "Evaluating the Usability of Mashups Applications," *8th International Conference on the Quality of Information and Communications Technology (QUATIC 2012)*, Lisbon, Portugal, 3-6 September 2012, IEEE Computer Society, pp. 323–326.

Publicaciones Enviadas:

- P. Cedillo, E. Insfrán, , J. González, S. Abrahão, "Design and Evaluation of a Monitoring Infrastructure for Cloud Services", *Information Sciences* Journal, Elsevier, Impact Factor 3.364 (JCR 2015), enviada Septiembre de 2016.

8.4. Estancia de investigación

Se ha realizado una estancia de investigación predoctoral de 3.5 meses en el *National Institute of Informatics (NII)* en Tokio, Japón, del 10 de diciembre de 2013 al 27 de marzo de 2014. El objetivo de la estancia ha sido doble: por un lado se ha estudiado los sistemas auto-adaptativos, lo que ha servido como punto de partida para la definición de la estrategia de monitorización de servicios planteada en esta tesis, y por otro lado, se ha estudiado las características de calidad involucradas con el ahorro de energía de los dispositivos móviles con sistema operativo Android, lo que ha contribuido a uno de los proyectos del NII.

8.5. Becas y galardones

- Beca pre-doctoral (2011-2015) provista por la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación de Ecuador (SENESCYT-Ecuador)
- Estancia pre-doctoral (Diciembre 2013 – Marzo 2014) Programa de Becas NII-Japón. Organización de Investigación y Sistemas.

- Beca para la escuela de verano TwinTide Autumn Trainging School sobre métodos para la interacción computador-hombre (TUTOREM 2013) de la Universidad de Leicester, UK.

8.6. Trabajos futuros

Esta tesis no es el final de los esfuerzos de investigación en este área. Muchas actividades aún se pueden realizar para mejorar y ampliar nuestro método y el trabajo futuro irá en esa dirección. Los principales aspectos que se piensa abordar son analizados a continuación.

Con respecto al método de monitorización:

- La integración de los atributos de calidad y métricas obtenidos en la revisión sistemática realizada por Navas (2016) en el Modelo de Calidad SaaS, con el objetivo de tener un instrumento más robusto que proporcione un amplio conjunto de características, atributos y métricas que apoyen a los stakeholders en la configuración de la monitorización.
- Medición de un mayor número de requisitos no funcionales relacionados con diferentes características de calidad. Que permitan descartar o vislumbrar posibles limitaciones o mejoras de la solución.
- Mejoras constantes de la arquitectura y del proceso de monitorización de acuerdo a nuevos hallazgos y a mejoras tecnológicas presentes en las diferentes plataformas.
- Extender la solución a la monitorización de otras tecnologías relacionadas tales como *fog computing* (Cisco Systems, 2016), teniendo en cuenta sus características intrínsecas, dada su cercanía e interacción con la nube.

Con respecto a la infraestructura de monitorización:

- Se deben incluir los adaptadores necesarios para cumplir con el cuarto escenario de extracción de datos, esto es, definir *wrappers* o APIs para comunicarse con otros servicios y así conseguir datos de monitorización de terceros que poder incorporar a la monitorización. De igual manera sería necesario mejorar el diseño del ciclo de vida del monitor de manera que aprovechara las posibilidades de la computación en paralelo y la gestión de múltiples instancias.
- Instanciación de la solución en otras plataformas (p. ej., Amazon AWS) que nos permitan monitorizar y analizar servicios desplegados en dichas

plataformas, ver futuras potencialidades y descubrir posibles problemas en la instanciación en esas herramientas.

- Creación de una aplicación que de soporte al proceso de análisis. Una vez obtenidos los datos es necesario que una aplicación web sea capaz de utilizar y tratar dichos datos para la obtención de informes sintéticos que permitan al usuario la confirmación del cumplimiento del SLA o la explicación de su violación e interpretación del mismo.

- Métodos de visualización de resultados haciendo uso de librerías que permitan mostrar los datos en diferentes formatos y formas de presentación.

- Autorregulación del tiempo de extracción de datos, para lo cual, sería conveniente el estudio de los datos extraídos para así definir de forma dinámica los tiempos del ciclo de extracción de datos, de manera que el sistema detecte las anomalías del servicio. Por ejemplo, que ésta tenga una actividad reducida y por lo tanto que en el caso de ciertos contadores no sea necesario recuperarlos de forma tan constante dado que los datos aportados son innecesarios. Así mismo sería aconsejable estudiar los tiempos más eficaces para la extracción de datos en relación al valor que pueden aportar con respecto al esfuerzo de extraerlos. Una vez que se pueda adquirir dicha información, poder reflejarla al modelo en tiempo de ejecución a fin de auto-adaptar los parámetros de monitorización, explotando así la potencialidad de los modelos en tiempo de ejecución.

- También se espera como trabajo futuro, explorar más a fondo la interoperabilidad de nuestro middleware de monitorización con otras herramientas de monitorización de calidad de servicios por medio de conectores, agentes, APIs, etc. Esta interoperabilidad potenciaría nuestra propuesta de forma significativa, ya que el cálculo de métricas específicas para ciertos atributos de calidad puede requerir de un gran esfuerzo de desarrollo y sin embargo podría actualmente estar ya disponible en la nube.

Con respecto a la validación y mejora en las evaluaciones de la solución:

- Con los resultados obtenidos tras la validación empírica realizada en el presente trabajo, se prevé la mejora de los instrumentos y métodos de evaluación de esta solución, a fin de eliminar todos los posibles problemas que se han dado durante las sesiones de experimentación.

- Experimentación y pruebas para que con los resultados de la monitorización podamos establecer las tasas de recolección de datos más optimas entre otros parámetros de monitorización; a fin de determinar los patrones y

particularidades que deben ser reflejadas en los modelos en tiempo de ejecución a fin de lograr una solución de monitorización auto-adaptativa. Todo esto explotando los beneficios de los modelos en tiempo de ejecución, desde una causalidad que vaya de los servicios a la infraestructura de monitorización.

- Réplicas de los estudios empíricos teniendo como sujetos a profesionales del área involucrados en la industria y otros sectores que permitan validar la solución desde diferentes puntos de vista de usuarios.
- Finalmente, se buscará evaluar la usabilidad/experiencia de usuario en cuanto al uso de los mecanismos de monitorización propuestos, a través de casos de estudio más complejos.

Capítulo 9
Conclusions and Further Work

This chapter reviews the research objectives established, their level of fulfillment, and the main findings that have been obtained from this work. Finally, we discuss the research results from the methodological and practical point-of-view and the opportunities for further research.

9.1. Conclusions

In the following sub-sections, we examine to what extent the general and specifics objectives of this thesis have been attained.

9.1.1. General objective

The goal of this thesis is the *definition and empirical evaluation of a method with which to monitor cloud services deployed as SaaS by using models at runtime*.

This objective has been attained in the form of a method with wich to monitor cloud services that exploits models at runtime. This solution solves several limitations of the existing approaches (e.g., lack of flexibility, inability to monitor high-level quality attributes). A monitoring process has been defined to support and guide users when applying the method. Moreover, a monitoring infrastructure with a platform-independent middleware has been designed, which has been instantiated and implemented to be deployed on Microsoft Azure. Finally, four quasi-experiments have been performed in order to provide empirical evidence about the usefulness of the method as regards supporting stakeholders during the monitoring configuration process and the detection of SLAs violations.

The proposed solution has several benefits when compared to other monitoring approaches:

1. The use of a SaaS Quality Model which supports users when configuring non-functional requirements to be monitored.
2. The possibility of monitoring not only low-level quality attributes (e.g., CPU use) but also high-level quality attributes (e.g., reliability, security) thanks to the SaaS Quality Model which allows the integration and combination of direct metrics so as to obtain indirect metrics.

3. The versatility of the infrastructure as regards supporting four scenarios in order to capture monitoring data (e.g., platform tools, agents, wrappers), which allows data to be collected from several different sources and users to operate with them.
4. The flexibility of the monitoring infrastructure which does not need to be re-implemented when a new requirement needs to be monitored. It is only necessary to establish new monitoring parameters and adapt them to the new needs. This is possible thanks to the use of the models at runtime.
5. The formalization of the artefacts of the monitoring infrastructure (i.e., Monitoring Requirements Model, SaaS Quality Model and Runtime Quality Model), which allows the specification of the parameters and monitoring requirements. These artifacts are based on domain specific languages and standards that allow a systematic evaluation of the cloud services quality.
6. The ease with which the metrics to be used during the monitoring can be specified. They are not limitated to pre-established metrics, but may vary according to the SLA or the needs of the parties involved in the provision of the service.
7. The possibility of extending the infrastructure to future implementations, which allow the auto-adaptation of the monitoring tool to the service status.

Finally, in order to achieve the main goal, certain specific objectives were formulated.

9.1.2. Specific objectives

The level of fulfillment of the specific objectives is analyzed below.

9.1.2.1. Specific objective 1

To define a monitoring method that uses models at runtime, which is supported by a repeatable and systematic process. The process should describe the method activities and the necessary tasks to carry out the activities.

This objective has been fulfilled by creating a method with which to monitor cloud services named Cloud MoS@RT. This method exploits models at runtime so as to provide flexibility when changing or creating new monitoring requirements.

The process which supports the monitoring method has been specified by using SPEM. It allowed us to model all the input and output artifacts, their main activities and roles. It also allowed us to describe the specific tasks into which each of the main activities can be divided, so that the method can be clearly

understood to be instantiated on a specific platform and be used by the stakeholders when they are configuring the service monitoring.

We planned activities to enable the monitoring of services within a process that uses non-functional requirements of the service user as input, in a systematic, standardized, and understandable way. This means of representing requirements is useful when specifying the service characteristics, as the requirements can be expressed in many different ways including natural language. It is also important to have a systematic means to represent the monitoring requirements in such a way that they can be understood by the computer. This has thus led the creation of a *Monitoring Requirements Model*. This model has been designed on the basis of the elements of an SLA and helps the stakeholders to express the monitoring requirements by using a domain specific language called WLSA.

The process also includes the use of a *SaaS Quality Model*, which decomposes the quality characteristics of services into sub-characteristics, attributes, and metrics in order to evaluate the cloud services quality. This model supports the definition of monitoring metrics. In order to design this model, we have used the software quality model proposed in the ISO/IEC 25010 standard, which has been extended with attributes and metrics for Software as a Service (SaaS). This decomposition has been carried out for the performance and reliability characteristics only, as the objective of this work was not to propose a complete quality model, but rather a monitoring method and its infrastructure. The SaaS Quality Model was sufficient for the purpose of demonstrating the utility of the monitoring method. However, we plan to extend this model with the results of a recent systematic review on quality attributes and metrics for cloud services conducted by Navas (2016) in the context of the Value@Cloud proyect. In this study, a quality model for cloud services has been proposed using 178 quality attributes and 364 metrics for cloud services (SaaS, PaaS and IaaS) collected from a systematic literature review.

Finally, the design of a model at runtime has been performed. This model contains a subset of the Monitoring Requirements Model and the SaaS Quality Model, and includes service characteristics that allow the monitoring of cloud services at runtime.

In summary, the proposed models have been defined in compliance with standards or widely accepted domain specific languages, which allow the establishment of the monitoring requirements and needs, the quality characteristics and attributes, the platform-independent metrics, and the platform-specific metrics and parameters in a systematic and repeatable manner.

The method presents a limitation regards to the feedback from the service to the monitoring infrastructure to attain an automatic tuning of the monitoring parameters based on the state of services. However, as this solution uses models at runtime, and the self-adaptability is one of their characteristics, this solution envisions this self-adaptability as future work.

9.1.2.2. Specific objective 2

To define a monitoring infrastructure whose design is independent of the cloud platform, and allows both the monitoring of cloud services and the detection of SLA violations. The infrastructure should exploit models at runtime in order to provide the desired flexibility and interoperability, and allow the specification of non-functional requirements from Service Level Agreements (SLA) and be in compliance with quality standards.

In order to fulfill this objective, a platform-independent monitoring infrastructure has been designed to support the proposed monitoring method.

The proposed infrastructure contains two main components, which allow the *configuration* of the non-functional requirements and the *monitoring* of the cloud service at runtime. The solution interacts with cloud services in order to evaluate their quality by gathering the information needed from various data sources. To support this task, we have established data extraction scenarios, which help stakeholders to explore the advantages of the platforms, operate with existing data, adapt and gather special data from the services, create wrappers and even allow the use of third-party applications.

The infrastructure additionally provides a high level of flexibility owing to the use of models at runtime which allows the definition of new non-functional requirements to be monitored that may be caused by either the renegotiations of SLAs or new monitoring needs from any stakeholder involved in the service provision.

It is worth noting that, after the empirical evaluations, the models have been refined in order to achieve a fine-tuning of the monitoring infrastructure, and an improvement of the usability. However, it will be necessary to check whether this design is generic enough to be instantiated on another cloud platforms such as Amazon Web Services.

9.1.2.3. Specific objective 3

To instantiate and implement the monitoring infrastructure on a specific cloud platform in order to perform the monitoring of services deployed in the cloud and corroborate the applicability and flexibility of the proposed infrastructure.

The instantiation of the solution has been attained and shown in Chapter 6. The instantiation of the method has been applied by means of a case study and the instantiation of the monitoring infrastructure proposed in Chapter 5, has been implemented to be deployed on a specific platform with an actual scenario usage.

We experienced several technological difficulties during the instantiation and implementation of the monitoring infrastructure in Microsoft Azure (e.g., the use of XMI format in .NET languages, the navigability of the models when programming the infrastructure). Nevertheless, these difficulties were successfully solved, showing that the method is easily applicable and that the problems confronted during the instantiation do not correspond with issues related to the design of the method or its supporting infrastructure but rather with issues related to the cloud platform and technologies. The development of these tools has enabled us to verify both the feasibility of using models at runtime and the proposed data gathering mechanisms.

Current monitoring tools are focused on obtaining low-level data from the platforms and deployed services. Although this is necessary, this work goes one step further by creating a layer between the raw data and their interpreters. This layer is useful as regard managing the huge amount of data obtained and interpreting them as high-quality information, which helps to make more informed decisions that improve the quality of services. The infrastructure designed does not, therefore, consider other tools as competitors but rather as allies, since they make it possible to obtain raw data that can be combined into more meaningful quality data. The mechanisms which gather external data have been taking into account by means of several data gathering scenarios. These scenarios provide the middleware with the ability to use libraries from the cloud platform, to access other third-party monitoring tools, to define customized measurement functions, and to extend the services (by means of wrappers) to allow them providing with data to be used by the interpreters.

Although this thesis shows the implementation of the monitoring infrastructure in Microsoft Azure, there are also additional implementations of the configurator and the middleware for the monitoring of cloud services deployed on the Google App Engine platform, which has been recently performed by López (2016) and Páez (2016) as part of their final degree project in the context of the Value@Cloud project. These implementations show the feasibility of instantiating the monitoring infrastructure design on another cloud platform. These implementations allowed us to corroborate the usefulness of Cloud MoS@RT as regards supporting the monitoring of cloud services.

However, the implemented prototypes have some limitation and need improvements with respect to usability. In particular, the user interface that shows the monitoring data and the SLA violations needs to have more options for visualizing the current state of the services and well as to follow up with the SLA violations and improvement recommendations. As future work, it is expected to build an interactive dashboard that improves the user experience.

Finally, another issue is the establishment of a fluid communication with other monitoring tools (e.g., general purpose monitoring tools, agents) by using generic connectors. Although our solution was defined to allow these interactions, it is necessary to study and define the specific communication channels that will be used to share the monitoring information.

9.1.2.4. Specific objective 4

To empirically evaluate the proposed monitoring method. This evaluation will provide us relevant information for the improvement of the solution.

We have performed a family of four quasi-experiments to analyze the actual efficacy of participants using the method, along with their perceptions (perceived ease of use, perceived usefulness, and intention to use). The results show that MoS@RT has been perceived as easy to use and useful. Moreover, the participants have expressed their intention to use this method in the future if they are required to monitor cloud services.

The users who performed the evaluation were of two types: Computer Science students who were enrolled on a course on Software Quality and Master's degree students, who are professionals that apply the information technologies in their work, and may understand the industrial implications of this kind of solutions.

The limitations of the evaluation are related to the participants' lack of experience with the services deployed in cloud environments. As future work, we therefore plan to replicate the quasi experiment in industrial environments. Another limitation is that the evaluation was focused only on the monitoring configuration phase of MoS@RT. It will thus be necessary to evaluate all the activities in the method (configuration, monitoring, and analysis of results) with the purpose of validating the use of the model at runtime, adjust the data gathering frequency, and other aspects such as overload. These issues will be addressed as future work.

Finally, it is also necessary to replicate the experiments, taking into account other high and low-level quality attributes and metrics to determine possible limitations of the solution, and other more complex SLAs, as well. We also

consider to evaluate more complex services deployed on multi-cloud environments.

9.2. Summary of the main contributions

This thesis has contributed to a solution to the problem of monitoring the quality of the cloud services and detecting non-compliances in the SLA. Among the most important contributions are the following:

- **The Cloud MoS@RT method**. We presented a cloud services monitoring method that uses models at runtime in order to provide flexibility and easy modifiability when there are changes in the monitoring requirements. These changes can be produced by SLA re-negotiations or by new monitoring needs. In order to accomplish the monitoring of cloud services, three main activities have been established: i) the monitoring configuration, ii) the cloud service monitoring, and iii) the monitoring information analysis. In addition, we have provided a detailed description of each one of the tasks involved in these activities in order to accomplish an effective cloud service monitoring.

- **The monitoring infrastructure.** We have proposed a monitoring infrastructure, which supports the monitoring method by using models at runtime. We have specifically proposed: i) a *Requirements Quality Model* which supports the specification of NFRs according to domain specific languages which allows the verification of SLA's compliance to be automated. ii) a *SaaS Quality Model* which is used by the monitoring infrastructure in order to obtain an effective configuration of NFRs; iii) a *Runtime Quality Model* which relates NFRs with raw data in order to monitor high-level requirements.

- **The instantiation of the method and the implementation of the infrastructure on a specific platform.** A prototype of the monitoring infrastructure has been designed and implemented in order to demonstrate the feasibility of the instantiation of the infrastructure in a specific platform. The Microsoft Azure platform has been selected and this task has been developed by Jimenez (2015) as part of his final degree work. This work has been conducted by following the infraestructura design presented in this thesis.

- **Theoretical model with which to evaluate monitoring methods:** In order to evaluate the Cloud MoS@RT method and its infrastructure, an operationalization of the MEM (Moody, 2001) has been applied, which can be used to evaluate service monitoring methods. This operationalization consists of the definition of a theoretical model (with different constructors)

in order to evaluate monitoring methods that are based on the user's perception and the definition of an instrument which allows the measurements of these constructors. This theoretical model can be used to design: i) quasi-experiments which allow the evaluation of any monitoring method with the objective of predicting its acceptation in practice, or ii) controlled experiments which allow the comparison of two or more monitoring methods so as to determine which one is the most efficient, effective, and with a higher possibility of being adopted in practice. From the research point of view, the theoretical model has been useful to provide a basis or theory with which to design the quasi-experiments that have been carried out in this thesis. From a practical point of view, although the causal relations of MEM (structural part of the model) have been confirmed in the quasi-experimented, the theoretical model should be applied in the design of other experiments and also in the design of controlled experiments to compare methods for services monitoring.

- **Empirical Evaluation:** The method and infrastructure proposed have been evaluated by mean of four quasi-experiments. In these experiments, participants from Spain, Paraguay, and Ecuador were selected. These studies allowed the evaluation of the usefulness of the monitoring method based on the participants' perception. From the research point of view, the quasi-experiments allowed the analysis of the applicability of the configuration process, the modeling of an actual case and the analysis of the degree of completeness and correction of the solutions. Finally, the participants' feedback helped us to improve the method and the infrastructure. From the practical point of view, this is the first study with non-advanced users and provides preliminary results about the usefulness, ease of use, and intention to use of the Cloud MoS@RT method. In future work, it will be necessary to replicate this quasi-experiment with expert users, and monitor more complex services and with a higher number of participants in order to analyze interactions and factors such as the participants' experience or skills.

Finally, the results of this thesis are useful from a theoretical point of view due to the integration of several specific technologies integrated into a specific monitoring solution: ISO standards, domain specific languages for SLAs, models at runtime, cloud platforms, etc. From a practical point of view, the method and the infrastructure still need to be used in actual environments (cloud and multi-cloud) with participants with more experience on cloud services, monitoring of services, and also with more complex services.

9.3. Related Publications

The solutions presented in this thesis were published in two international conferences, which are ranked as A in the ERA CORE ranking. One of them was published in the most relevant conference in the service computing field (SCC). Moreover, we have presented a contribution in an international workshop, which is specific for models at runtime, a publication in the ERCIM News Journal, a publication in the *Jornadas Nacionales de Ciencia e Ingeniería de Servicios* (JCIS), and also other publications in international conferences which were helpful to understand the service principles, quality models, and quality characteristics. Finally, we have sent a manuscript to the *Information Sciences* Journal, which includes the results of the design and evaluation of the monitoring solution. We show following these publications with more details:

- P. Cedillo, J. Jimenez-Gomez, S. Abrahão, and E. Insfran, "Towards a Monitoring Middleware for Cloud Services," *9th IEEE International Conference on Services Computing (SCC 2015)*, June 27-July 2, 2015, New York, USA, pp. 451-458, IEEE Press. ERA CORE Tier A.

- P. Cedillo, J. Gonzalez-Huerta, S. Abrahão, and E. Insfrán, "A Monitoring Infrastructure for the Quality Assessment of Cloud Services," *24th International Conference on Information Systems Development (ISD 2015*), Model-Driven Development and Concepts Track, August 25 - 27, 2015, Harbin, China, pp. 17-32, Springer. ERA CORE Tier A.

- Cedillo, P., Gonzalez-Huerta, J., Insfrán, E., & Abrahão, S. Towards Monitoring Cloud Services Using Models@run time. *9th International Workshop on Models at run.time (MRT 2014)*, co-located with the ACM/IEEE 17th International Conference on Model Driven Engineering Language and Systems (MODELS 2014), Valencia, Spain, pp. 31–40.

- M. A. Zúñiga-Prieto, P. Cedillo, J. González-Huerta, E. Insfrán, and S. Abrahão, "Monitoring Services Quality in the Cloud," *ERCIM News*, no. 99, *Special Theme on Software Quality*, October 2014, pp. 19–20.

- P. Cedillo, J. Jimenez-Gomez, S. Abrahão, and E. Insfrán, "Definición de Mecanismos Personalizados de Monitorización de Servicios Cloud," *XI Jornadas de Ciencia e Ingenieria de Servicios (JCIS 2015)*, Santander, 15-17 Septiembre 2015.

- P. Cedillo, A. Fernandez, A. Insfran, and S. Abrahão, "Quality of Web Mashups: A Systematic Mapping Study," *4th International Workshop on Quality in Web Engineering (QWE 2013)*, co-located with the 13th International

Conference on Web Engineering (ICWE), Aalborg, Denmark, LNCS 8295, Springer, pp. 66–78, 2013.

- E. Insfran, P. Cedillo, A. Fernandez, S. Abrahão, and M. Matera, "Evaluating the Usability of Mashups Applications," *8th International Conference on the Quality of Information and Communications Technology (QUATIC 2012)*, Lisbon, Portugal, 3-6 September 2012, IEEE Computer Society, pp. 323–326.

Submitted Publications:

- P. Cedillo, E. Insfrán , J. González, S. Abrahão, "Design and Evaluation of a Monitoring Infrastructure for Cloud Services", *Information Sciences* Journal, Elsevier, Impact Factor 3.364 (JCR 2015), which was sent in September, 2016.

9.4. Research Stay

Pre-doctoral research stay for 3.5 months at the National Institute of Informatics (NII) in Tokyo, Japan, from December 10th, 2013 to March 27th, 2014. The purpose of this stay was twofold: on the one hand, the author studied the subject of self-adaptive systems, which was useful as a starting point for the definition of the strategy proposed in this thesis, and on the other, she studied the characteristics of quality involved with the energy preservation of mobile devices with the Android OS, which has contributed to one of the NII projects.

9.5. Grants and awards

- Doctoral fellowship (2011-2015) provided by the Senescyt-Ecuador (Secretaría de Educación Superior; Ciencia, Tecnología e Innovación).
- NII Internship Program. Research Organization of Information and Systems.
- Scholarship for TwinTide Auntumn Training School on Research Methods for Human-Computer Interaction (TUTOREM 2013) at Leicester University, UK.

9.6. Next steps

This PhD thesis does not represent the end of this research work. Although we have achieved all objectives defined, there is still enough space for improvement. Some of these improvements are now in progress; however, many

others will be addressed in the near future. Some of these are discussed following.

With regard to the monitoring method:

- The integration into the SaaS Quality Model of the quality attributes and metrics obtained in the systematic review performed by Navas (2016). The objective will be to obtain a robust instrument, which provides a wide set of characteristics, attributes and metrics that will support stakeholders when configuring the monitoring.
- We plan to measure a larger number of NFRs related to different quality attributes. These measurements will allow us to detect the limitations that should be considered for future improvements of the solution.
- Improvement and refining the method and the infrastructure according to new findings and technological developments.
- Extend the solution to other related technologies such as *fog computing* (CISCO, 2015), taking into account its intrinsic characteristics since they are very those of the cloud computing.

With regard to the monitoring infrastructure:

- The building of adaptors to be used in the fourth scenario of data gathering (wrappers), by defining the corresponding packages or APIs in order to communicate with other services or solutions to gather data from third-party sources and to use these data in our infrastructure.
- The instantiation of the solution on other platforms (e.g., Amazon AWS), which will allow us to determine new future potentialities and discover possible problems on other platforms.
- The creation of an application in order to support the analysis process. Once the data have been obtained and interpreted, it will be necessary a means to show the information and to create reports, which includes the explanation of SLA violations.
- The creation of methods with which to visualize the monitoring results by using graphics or infographics and specialized libraries to show the information in different formats and styles.
- The implementation of a causal connection between the model at runtime and the service in order to auto-adapt the monitoring with the status of the service (e.g., data extraction rate). This implementation will be useful to

avoid service overload, improve the precision, and optimize the storage of monitoring data.

- Create specific plugins with third-party tools in order to improve interoperability (e.g., agents, specialized monitoring tools, other tools). This interoperability may improve our solution in a significant way since several specialized tools may offer functionalities that can be aggregated to our solution (e.g., analysis, visualization).

With regard to the validation and improvements of the solution:

- With the results obtained during the empirical validation presented in this thesis, we plan to improve the instruments and evaluation methods that will be used in future replications of the experiments.
- Experimentation and testing to obtain monitoring results, which will allow us to establish the most appropriate rates for data collecting, and other monitoring parameters. The objective is to determine patterns and particularities that should be reflected in the models at runtime, in order to achieve an auto-adaptive monitoring solution. These activities will help us to exploit the benefits of the models at runtime, with a causality from the services towards the infrastructure.
- Replicate the empirical validation with participants specialized in cloud platforms and industrial environments in order to prove the solution from different viewpoints.
- Finally, we plan to evaluate the usability/user experience as regards the use of the monitoring mechanisms through advanced case studies.

Bibliografía

Abdelmaboud, A., Jawawi, D. N. a., Ghani, I., Elsafi, A., Kitchenham, B. Quality of service approaches in cloud computing: A systematic mapping study. *Journal of Systems and Software*, *101*, 159–179 (2015).

Abrahão, S., Insfran, E., Carsí, J. A., Genero, M. Evaluating requirements modeling methods based on user perceptions: A family of experiments. *Information Sciences*, *181*(16), 3356–3378 (2011).

Abramowicz, W., Hofman, R. SQuaRE based web services quality model. Presentado en: *International Association of Engineers (IAENG)* (Vol. I, pp. 19–21). Hong Kong, China (2008).

Aceto, G., Botta, A., de Donato, W., Pescapè, A. Cloud monitoring: A survey. *Computer Networks*, *57*(9), 2093–2115 (2013).

Adams, D. A., Nelson, R. R., Todd, P. A. Perceived Usefulness, Ease of Use, and Usage of Information Technology: A Replication. *MIS Quarterly*, *16*(2), 227–247 (1992).

Alhamad, M., Dillon, T., Chang, E. A survey on SLA and performance measurement in cloud computing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7045 LNCS*(PART 2), 469–477 (2011).

Alhamazani, K., Ranjan, R., Mitra, K., Rabhi, F., Jayaraman, P. P., Khan, S. U., Guabtni, A., Bhatnagar, V. An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing*, *97*(4), 357–377 (2015).

Aljohani, A. M., Holton, D. R. W., Awan, I., Alanazi, J. S. Performance evaluation of local and cloud deployment of web clusters. Presentado en: *International Conference on Network-Based Information Systems, NBiS* (pp. 274–278). Tirana, Albania (2011).

Almeida, A., Bencomo, N., Batista, T., Cavalcante, E., Dantas, F. Dynamic decision-making based on NFR for managing software variability and configuration selection. Presentado en: *30th Annual ACM Symposium on Applied Computing - SAC '15* (pp. 1376–1382). New York, NY, USA (2015).

Almeida, V., Menasce, D. Capacity planning an essential tool for managing Web services. *It Professional*, *4*(4), 33–38 (2002).

Amazon. Amazon Web Services (2016). https://aws.amazon.com/es/products/?nc2=h_ql_ny_livestream_blu

Amazon Cloud Services. Amazon Cloud Watch (2016). https://aws.amazon.com/es/cloudwatch/

Amazon Web Services. *Amazon Cloud Watch. User Guide* (2016). Retrieved from http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/acw-

ug.pdf

Ameller, D., Franch, X. Service level agreement monitor (SALMon). Presentado en: *7th International Conference on Composition-Based Software Systems, ICCBSS 2008* (pp. 224–227). Madrid, Spain (2008).

Anand, M. Cloud Monitor: Monitoring Applications in Cloud. Presentado en: *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)* (pp. 1–4). Bangalore, KA, India (2012).

Andrieux, A., Czajkowski, K., Keahey, K., Dan, A., Keahey, K., Ludwig, H. Web Services Agreement Specification (WS-Agreement). *Global Grid Forum GRAAP-WG*, *192*, 1–80 (2004).

Ardagna, D., Ghezzi, C., Mirandola, R. Rethinking the Use of Models in Software Architecture. Presentado en: *4th International Conference on Quality of Software Architectures. Models and Architectures (QoSA)* (Vol. 5281, pp. 1–27). Karlsruhe, Germany (2008).

Ardagna, D., Panicucci, B., Passacantando, M. Generalized nash equilibria for the service provisioning problem in cloud systems. *IEEE Transactions on Services Computing*, *6*(4), 429–442 (2013).

Ardagna, D., Zhang, L. *Run-time Models for Self-managing Systems and Applications*. *Annals of Physics* (Vol. 54). Milan, Italy: Springer (2010).

Armbrust, M., Fox, A., Griffith, R., Joseph, A., RH. Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has (Finally) Come. *University of California, Berkeley, Tech. Rep. UCB*, *53*(UCB/EECS-2009-28), 07–013 (2009).

Armellin, G., Chiasera, A., Frankova, G., Pasquale, L., Torelli, F., Zacco, G. The eGovernment Use Case Scenario SLA Management Automation of Public Services Giampaolo. Presentado en: *SLAs for Cloud Computing* (pp. 343 – 357). NY (2011).

Bai, X., Liu, Y., Wang, L., Tsai, W.-T., Zhong, P. Model-Based Monitoring and Policy Enforcement of Services. Presentado en: *1st. World Conference on Services* (pp. 789–796). Los Angeles, CA, USA (2009).

Baresi, L., Ghezzi, C. The Disappearing Boundary Between Development-time and Run-time. Presentado en: *Workshop on Future of Software Engineering Research FSE/SDP* (pp. 17–22). Santa Fe, New Mexico, USA: ACM (2010).

Barret, R. Windows Azure FC (2013). https://snarfed.org/windows_azure_details#Configuration_and_APIs

Baset, S. A. Cloud SLAs. *ACM SIGOPS Operating Systems Review*, *46*(2), 57 (2012).

Basili, V. R. The Experimental Paradigm in Software Engineering. Presentado en: *Workshop on Experimental Software Engineering Issues: Critical Assessment and Future*

Directions (pp. 1–12). London, UK: LNCS 706, Springer (1992).

Basili, V. R., Caldiera, G., Rombach, H. D. The goal question metric approach. Presentado en: *Encyclopedia of Software Engineering* (Vol. 2, pp. 1–10). Wiley (1994).

Basili, V. R., Selby, R. W., Hutchens, D. H. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, *SE-12*(7), 733–743 (1986).

Basili, V., Shull, F., Lanubile, F. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, *25*(4), 456–473 (1999).

Bauer, E., Adams, R. *Service Quality of Cloud-Based Applications* (Vol. 18). Hoboken, New Jersey, USA: John Wiley & Sons, Inc. (2013).

Becker, M., Lehrig, S., Becker, S. Systematically deriving quality metrics for cloud computing systems. *6th ACM/SPEC International Conference on Performance Engineering. ICPE*, (January 2016), 169–174 (2015).

Ben Hamida, A., Bertolino, A., Calabrò, A., De Angelis, G., Lago, N., Lesbegueries, J. Monitoring Service Choreographies from Multiple Sources. Presentado en: *4th International Workshop on Software Engineering for Resilient Systems* (Vol. 7527, pp. 134–149). Pisa, Italy: Springer Berlin Heidelberg (2012).

Bencomo, N., Blair, G., Götz, S., Morin, B., Rumpe, B. Report on the 7th International Workshop on Models@Run.Time. *SIGSOFT Softw. Eng. Notes*, *38*(1), 27–30 (2013).

Bertolino, A., Calabrò, A., Lonetti, F., Di Marco, A., Sabetta, A. Towards a Model-Driven Infrastructure for Runtime Monitoring. Presentado en: *3rd International Workshop on Software Engineering for Resilient Systems* (Vol. 6968, pp. 130–144). Geneva, Switzerland: Springer Berlin Heidelberg (2011).

Bianco, P., Lewis, G., Merson, P. *Service Level Agreements in Service-Oriented Architecture Environments*. Pittsburgh, USA (2008).

Blair, G., Bencomo, N., France, R. B. Models@run.time. *Computer*, *42*(10), 22–27 (2009).

Bodenstaff, L., Wombacher, A., Reichert, M. Empirical Validation of MoDe4SLA; Approach for Managing Service Compositions. Presentado en: *14th International Conference on Business Inofrmation Systems* (pp. 98–110). Poznan, Poland (2011).

Bolch, G., Greiner, S., de Meer, H., Trivedi, K. S. *Queueing Networks and Markov Chains. Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications: Second Edition* (2nd ed.). Wiley Publishing, Inc (2006).

Brambilla, M., Cabot, J., Wimmer, M. *Model-Driven Software Engineering in Practice. Synthesis Lectures on Software Engineering* (Vol. 1). Morgan y Claypool (2012).

Breton, E., Bézivin, J. Towards an Understanding of Model Executability. Presentado en: *International Conference on Formal Ontology in Information Systems - Volume 2001* (pp. 70–80). New York, NY, USA: ACM (2001).

Bruijn, J. de, Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., … Stollberg,

M. Web Service Modeling Ontology (WSMO) (2007). http://www.wsmo.org/TR/d2/v1.4/

Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., Konig-Ries, B. Web Service Modeling Ontology (WSMO) (2005). http://www.w3.org/Submission/WSMO/

Bugtracking system. collectd – The system statistics collection daemon (2015). https://collectd.org/

Cabrera, O., Franch, X. A quality model for analysing web service monitoring tools. Presentado en: *6th International Conference on Research Challenges in Information Science, RCIS*. Valencia, Spain (2012).

Cancian, H. M. *A proposal of reference guide for providers of software as a service. Dissertation in Automation and Systems Engineering* (2009).

Cancian, H. M., Rabelo, R. J., Wangenheim, C. G. V. A proposal for drawing up of the contract for Level of Service to Software as a Service (SaaS). Presentado en: *8th International Information and Telecommunication Technologies Symposium* (p. 410). Florianópolis, Brasil (2009).

Cappiello, C., Daniel, F., Koschmider, A., Matera, M., Picozzi, M. A Quality Model for Mashups. Presentado en: *9th International Conference, (ICWE)* (Vol. 6757, pp. 137–151). San Sebastián, Spain: Springer Berlin Heidelberg (2011).

Carifio, J., Perla, R. J. Ten Common Misunderstandings, Misconceptions, Persistent Myths and Urban Legends about Likert Scales and Likert Response Formats and their Antidotes. *Journal of Social Sciences*, *3*(3), 106–116 (2007).

Caron, E., Rodero-Merino, L., Desprez, F., Muresan, A. Auto-scaling, load balancing and monitoring in commercial and open-source clouds. *Inria Informatics Matematics*, 0–24 (2012).

Cedillo, P., Gonzalez-Huerta, J., Abrahao, S., Insfrán, E. A Monitoring Infrastructure for the Quality of Cloud Services. Presentado en: *24th International Conference on Information Systems Development , ISD*. Harbin, China (2015).

Cedillo, P., Gonzalez-Huerta, J., Insfrán, E., Abrahao, S. Towards Monitoring Cloud Services Using Models@run time. Presentado en: *Wshop. on Models@run.time, MODELS* (pp. 31–40). Valencia, Spain (2014).

Cedillo, P., Jimenez-Gomez, J., Abrahao, S., Insfrán, E. Definición de Mecanismos Personalizados de Monitorización de Servicios Cloud. Presentado en: *XI Jornadas de Ciencia e Ingenieria de Servicios*. Salamanca, Spain (2015).

Chauhan, M. A., Babar, M. A. Cloud Infrastructure for Providing Tools As a Service: Quality Attributes and Potential Solutions. Presentado en: *Working IEEE/IFIP Conference on Software Architecture, WICSA* (pp. 5–13). New York, NY, USA: ACM (2012).

Chen, Y., Paxson, V., Katz, R. H. What's New About Cloud Computing Security? *University of California, Berkeley Report No. UCB/EECS-2010-5 January*, *20*, 1–8 (2010).

Cheng, S. W., Garlan, D., Schmerl, B., Steenkiste, P., Hu, N. Software architecture-based adaptation for Grid computing. Presentado en: *IEEE International Symposium on High Performance Distributed Computing* (Vol. January, pp. 389–398). Edinburgh, Scotland (2002).

Chung, W.-C., Chang, R.-S. A new mechanism for resource monitoring in Grid computing. *Future Generation Computer Systems*, *25*(1), 1–7 (2009).

Cianciaruso, L., Di Forenza, F., Di Nitto, E., Miglierina, M., Ferry, N., Solberg, A. Using models at runtime to support adaptable monitoring of multi-clouds applications. Presentado en: *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014* (pp. 401–408). Timisoara, Romania (2015).

Cisco Systems. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are (2016). http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

Cloud WatchHUB. MODAClouds (2015). http://www.modaclouds.eu/

Cloud WatchHUB. Cloud4SOA (2016). http://www.cloudwatchhub.eu/cloud4soa-%E2%80%93-bringing-interoperability-portability-paas

CloudHarmony (2010). https://cloudharmony.com/

Comuzzi, M., Jacobs, G., Grefen, P. *Clearing the Sky - Understanding SLA Elements in Cloud Computing*. Eindhoven, Nederland (2013).

Comuzzi, M., Kotsokalis, C., Spanoudakis, G., Yahyapour, R. Establishing and monitoring slas in complex service based systems. Presentado en: *IEEE International Conference on Web Services, ICWS 2009* (pp. 783–790). Los Angeles, CA, USA (2009).

Comuzzi, M., Pernici, B. A framework for QoS-based Web service contracting. *ACM Transactions on the Web*, *3*(3), 1–52 (2009).

Consortium, C. Cloud Service Measurement Index Consortium (CSMIC), SMI framework. (2015). https://slate.adobe.com/a/PN39b/

Cook, T. D., Campell, D. T. *Quasi-experimenation design and analysis issues for field settings*. Boston, MA, USA: Houghton Mifflin Company (1979).

Cordova, S. *Model-Driven Engineering* (1st Editio). White Word Publications (2012).

Correia, A., e Abreu, F. Model-Driven Service Level Management. Presentado en: B. Stiller & F. De Turck (Eds.), *Mechanisms for Autonomous Management of Networks and Services* (Vol. 6155, pp. 85–88). Springer Berlin Heidelberg (2010).

Correia, A., e Abreu, F. B., Amaral, V. SLALOM: a Language for SLA specification and

monitoring. Presentado en: *Inforum* (Vol. abs/1109.6, pp. 556–567). Coimbra, Portugal (2011).

Costa, F. M., Provensi, L. L., Vaz, F. F. Using runtime models to unify and structure the handling of meta-information in reflective middleware. Presentado en: *International conference on Models in software engineering* (pp. 232–241). Genova, Italy (2006).

Csmic. Service Measurement Index Introducing the Service Measurement Index (SMI), (July), 1–8 (2014).

Davis, C. R., Neville, S., Fernandez, J. M., Robert, J. M., McHugh, J. Structured peer-to-peer overlay networks: Ideal botnets command and control infrastructures? Presentado en: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5283 LNCS, pp. 461–480) (2008).

Davis, F., Bagozzi, R. P., Warshaw, P. R. *Technology Acceptance Model for Empirically Testing New End-user Information Systems: Theory and Results. PHD Thesis.* Massachusetts Institute of Technology, Sloan School of Management (1989). https://doi.org/10.1287/mnsc.35.8.982

De Chaves, S. A., Uriarte, R. B., Westphall, C. B. Toward an architecture for monitoring private clouds. *IEEE Communications Magazine*, *49*(12), 130–137 (2011).

Di Penta, M., Canfora, G., Esposito, G., Mazza, V., Bruno, M. Search-based testing of service level agreements. Presentado en: *9th annual conference on Genetic and evolutionary computation - GECCO '07* (p. 1090). London, UK (2007).

Dubey, A., Wagle, D. Delivering software as a service. *The McKinsey Quarterly*, *6*(May), 1–12 (2007).

EGI. EGI Federated cloud (2015). https://www.egi.eu/infrastructure/cloud/

Elitzur, M. On the Unification of Megamodels. *Electronics Communications of the EASST*, *42*, 12 (2011).

Emeakaroha, V. C., Brandic, I., Maurer, M., Dustdar, S. Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments. Presentado en: *Int. Conf. on High Performance Computing and Simulation (HPCS)* (pp. 48–54). Caen, France (2010).

Emeakaroha, V. C., Ferreto, T. C., Netto, M. A. S., Brandic, I., De Rose, C. A. F. CASViD: Application Level Monitoring for SLA Violation Detection in Clouds. Presentado en: *36th Computer Software and Applications Conference (COMPSAC)* (pp. 499–508). Izmir, Turkey (2012).

E-Speak Web (2015). http://www.e-speak.net/

Estoy en la Nube. Qué es Windows Azure (2016). http://www.estoyenlanube.com/recursos/windows-azure/que-es-windows-

azure/

European, C., (FP7). Framework Programme Seven. CloudScale (2012). http://www.cloudscale-project.eu/

Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P., Lynn, T. A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, *74*, 2918–2933 (2014).

Fenech, T. Using perceived ease of use and perceived usefulness to predict acceptance of the World Wide Web. *Computer Networks and ISDN Systems*, *30*(1-7), 629–630 (1998).

Fernández, A. *A Usability Inspection Method for Model-driven Web Development Processes. PhD Thesis*. Universitat Politècnica de València (2012). Retrieved from riunet.upv.es/bitstream/handle/10251/17845/tesisUPV3981.pdf?

Ferry, N., Song, H., Rossini, A., Chauvel, F., Solberg, A. Cloud MF: Applying MDE to tame the complexity of managing multi-cloud applications. Presentado en: *7th International Conference on Utility and Cloud Computing, UCC* (pp. 269–277). London, UK (2014).

Fishbein, M., Ajzen, I. Belief, Attitude, Intention and Behaviour: An Introduction to Theory and Research. *Reading MA AddisonWesley* (1975).

Freitas, N., Filho, D., Isaias, C., Bermejo, H. D. S., Zambalde, A. L., Barros, U. S. De. SaaS Quality - A Method for Quality Evaluation of Software as a Service. *International Journal of Computer Science & Information Technology (IJCSIT)*, *5*(3), 101–117 (2013).

Frutos, H. M., Kotsiopoulos, I., Gonzalez, L. M. V., Merino, L. R. Enhancing Service Selection by Semantic QoS. Presentado en: *6th European Semantic Web Conference on The Semantic Web: Research and Applications, ESWC* (pp. 565–577). Crete, Greece (2009).

Ganglia Monitoring System (2012). http://ganglia.info/

García, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruíz, F., Piattini, M., Genero, M. Towards a consistent terminology for software measurement. *Information and Software Technology*, *48*, 631–644 (2006).

Garlan, D., Cheng, S. W., Huang, A. C., Schmerl, B., Steenkiste, P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, *37*(10), 46–54 (2004).

Garlan, D., Schmerl, B. Model-based adaptation for self-healing systems. Presentado en: *1st workshop on Selfhealing systems WOSS 02* (p. 27). South Carolina, USA (2002).

Garlan, D., Schmerl, B., Chang, J. Using gauges for architecture-based monitoring and adaptation. Presentado en: *Working Conference on Complex and Dynamic Systems Architecture* (pp. 27–32). Brisbane, Australia (2001).

Gehlert, a, Metzger, A. Quality reference model for SBA. *Deliverable# CD-JRA-1.3* (2008).

Gens, F. Defining "Cloud Services" and "Cloud Computing" (2008). http://blogs.idc.com/ie/?p=190

GEYSER - Green nEtworked data centers as energY proSumers in smaRt city environments (2013). http://www.geyser-project.eu/

Giannakouris, K., Smihily, M. Cloud computing - statistics on the use by enterprises (2014). http://ec.europa.eu/eurostat/statistics-explained/index.php/Cloud_computing_-_statistics_on_the_use_by_enterprises

Giese, H., Lambers, L., Becker, B., Hildebrandt, S., Neumann, S., Vogel, T., Wätzoldt, S. Graph transformations for MDE, adaptation, and models at runtime. Presentado en: *12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems* (Vol. 7320 LNCS, pp. 137–191). Bertinoro, Italy (2012).

Giese, H., Wagner, R. Incremental Model Synchronization with Triple Graph Grammars. Presentado en: *9th International Conference, MoDELS* (pp. 543–557). Genova, Italy (2006).

Giese, H., Wagner, R. From model transformation to incremental bidirectional model synchronization. *Software and Systems Modeling*, *8*(1), 21–43 (2009).

Gonzalez, A. J., Helvik, B. E. System Management to Comply with SLA Availability Guarantees in Cloud Computing. Presentado en: *4th Int. Conf. on Cloud Computing Technology and Science* (pp. 325–332). Taipei, China (2012).

Gonzalez-Huerta, J. *Derivación , Evaluación y Mejora de la Calidad de Arquitecturas Software en el Desarrollo de Líneas de Producto Software Dirigido por Modelos. PHD Thesis*. Universitat Politècnica de València (2014).

Google. Google Cloud Platform (2016). https://cloud.google.com/

Gorschek, T., Wohlin, C., Garre, P., Larsson, S. A Model for Technology Transfer in Practice. *IEEE Software*, *23*(6), 88–95 (2006).

Grati, R., Boukadi, D., Ben-Abdallah, H. A QoS Monitoring Framework for Composite Web Services in the Cloud. Presentado en: IARIA (Ed.) (pp. 65–70). Barcelona, Spain (2012).

Grobauer, B., Walloschek, T., Stucker, E. Understanding cloud computing vulnerabilities. *IEEE Security and Privacy*, *9*(2), 50–57 (2011).

Hasselmeyer, P., D'Heureuse, N. Towards holistic multi-tenant monitoring for virtual data centers. Presentado en: *IEEE/IFIP Network Operations and Management Symposium Workshops, NOMS* (pp. 350–356). Osaka, Japan (2010).

Havelka, D., Sutton, S. G., Arnold, V. A methodology for developing measurement criteria for assurance services: An application in information systems assurance.

Auditing, *17*(SUPPL. 1), 58–92 (1998).

He, Q., Han, J., Yang, Y., Grundy, J., Jin, H. QoS-driven service selection for multi-tenant SaaS. Presentado en: *5th International Conference on Cloud Computing, CLOUD 2012* (pp. 566–573). Honolulu, Hawaii, USA (2012).

Hendrickson, A. R., Massey, P. D., Cronan, T. P. On the Test-Retest Reliability of Perceived Usefulness and Perceived Ease of Use Scales. *MIS Quarterly*, *17*(2), 227–230 (1993).

Herold, S., Klus, H., Welsch, Y., Deiters, C., Rausch, A., Reussner, R., … Pfaller, C. CoCoME - The Common Component Modeling Example. Presentado en: *The Common Component Modeling Example* (pp. 16–53). Germany (2008).

Hugos, M., Hulitzky, D. *Business in the Cloud: What every business needs to know about cloud computing*. USA: John Wiley & Sons, Inc. (2010).

IDERA (2016). http://www.idera.com/infrastructure-monitoring-as-a-service

Insfran, E., Cedillo, P., Fernandez, A., Abrahao, S., Matera, M. Evaluating the Usability of Mashups Applications. Presentado en: *8th International Conference on the Quality of Information and Communications Technology (QUATIC)* (pp. 323–326). Lisbon, Portugal (2012).

IRMOS (n.d.). http://www.irmosproject.eu/Default.aspx

ISO. ISO/IEC 25010 Systems and software engineering System and software product Quality Requirements and Evaluation (SQuaRE) (2011).

ISO. Standards and projects under the direct responsability of ISO/IEC JTC 1/SC 38 Secretariat (2016). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=601355

ISO/IEC. *ISO/IEC 9126. Software engineering -- Product quality*. ISO/IEC (2001).

ISO/IEC. ISO/IEC 25010 Systems and Software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. (2011).

ISO/IEC 17788:2014. Information technology -- Cloud computing -- Overview and vocabulary (2014). http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=60544

Jimenez, J. *Middleware para la monitorización de la calidad de servicios cloud*. Universitat Politècnica de València (2015).

Jin, L., Machiraju, V., Sahai, A. Analysis on Service Level Agreement of Web Services. *HP June* (2002).

Johnson, R. A., Hardgrave, B. C. Object-oriented methods: current practices and attitudes. *Journal of Systems and Software*, *48*(1), 5–12 (1999).

Joseph, M. *Software metrics: Establishing a company-wide program*. (Finantial Times / Prentice

Hall, Ed.), *Information and Software Technology* (02 ed., Vol. 30) (1988).

Jureta, I. J., Herssens, C., Faulkner, S. A comprehensive quality model for service-oriented systems. *Software Quality Journal*, *17*(1), 65–98 (2009).

Kan, S. *Metrics and Models in Software Quality Engineering* (2nd Editio). Indiana, USA: Addison Wesley (2003).

Katsaros, G., Kousiouris, G., Gogouvitis, S. V., Kyriazis, D., Menychtas, A., Varvarigou, T. A Self-adaptive hierarchical monitoring mechanism for Clouds. *Journal of Systems and Software*, *85*(5), 1029–1041 (2012).

Katsaros, G., Kübert, R., Gallizo, G. Building a service-oriented monitoring framework with REST and nagios. Presentado en: *IEEE International Conference on Services Computing, SCC* (pp. 426–431). Washington DC, USA (2011).

Keller, a, Ludwig, H. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, *11*(1), 57–81 (2003).

Keller, A., Ludwig, H. Defining and monitoring service level agreements for dynamic e-business. Presentado en: *16th System Administration Conference, LISA* (pp. 189–204). Philadelphia, PA, USA (2002).

Keller, A., Ludwig, H. The WSLA Framework: Specifying and Monitoring SLAs for Web Services. *Journal of Network and Systems Management*, *11*(1), 57–81 (2003).

Kim, E., Lee, Y. Quality model for web services. *Working Draft, OASIS*, (September), 1–45 (2005).

Kim, E., Lee, Y., Park, H., Kim, J., Moon, B., Yun, J., Kang, G. Web Services Quality Factors Version 1 . 0 Candidate OASIS Standard 01, (October), 1–29 (2012).

Kiwi. Kiwi Application Monitor (2013). http://www.kiwimonitor.com/dl.php

Kreger, H. Web Services Conceptual Architecture (WSCA) (2001). http://users.cs.uoi.gr/~pitoura/courses/ds04_gr/webt.pdf

Kritikos, K., Plexousakis, D. Semantic QoS-based Web Service Discovery Algorithms. Presentado en: *5th IEEE European Conference on Web Services*. Halle, Germany: IEEE Computer Society (2007).

Kumar, V., Cooper, B. F., Eisenhauer, G., Schwan, K., Mansour, M., Seshasayee, B., Widener, P. Implementing Diverse Messaging Models with Self-Managing Properties using IFLOW. Presentado en: *3rd IEEE International Conference on Autonomic Computing, ICAC* (pp. 243–252). Dublin, Ireland (2006).

Kutare, M., Eisenhauer, G., Wang, C., Schwan, K., Talwar, V., Wolf, M. Monalytics: online monitoring and analytics for managing large scale data centers. Presentado en: *7th International conference on Autonomic computing* (pp. 141–150). Washington, DC, USA (2010).

Kyriazis, D. *Cloud Computing Service Level Agreements - Exploitation of Research Results*.

European Commission (2013). Retrieved from http://ec.europa.eu/digital-agenda/en/news/cloud-computing-service-level-agreements-exploitation-research-results

Lakshmanan, G. T., Keyser, P., Slominski, A., Curbera, F., Khalaf, R. A business centric end-to-end monitoring approach for service composites. Presentado en: *7th International Conference on Services Computing, SCC* (pp. 409–416). Miami, Florida, USA (2010).

Le Duc, B., Collet, P., Malenfant, J., Rivierre, N. A QoI-aware Framework for Adaptive Monitoring. Presentado en: *2nd International Conference on Adaptive and Self-Adaptive Systems and Applications, ADAPTIVE* (pp. 133–141). Lisbon, Portugal (2010).

Lee, K., Jeon, J., Lee, W., Jeong, S.-H., Park, S.-W. QoS for Web Services: Requirements and Possible Approaches. Ontario, Canada (2003).

Lee, Y., Lee, J., Cheun, D., Kim, S. A Quality Model for Evaluating Software-as-a-Service in Cloud Computing. Presentado en: *7th International Conference on Software Engineering Research, Management and Applications, SERA* (pp. 261–266). Washington, DC, USA: IEEE Computer Society (2009).

Legrand, I., Newman, H., Voicu, R., Cirstoiu, C., Grigoras, C., Dobre, C., … Stratan, C. MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems. *Computer Physics Communications*, *180*(12), 2472–2498 (2009).

Lehmann, G., Blumendorf, M., Trollmann, F., Albayrak, S. Meta-modeling Runtime Models. Presentado en: *International Conference on Models in Software Engineering, MODELS* (pp. 209–223). Berlin, Germany: Springer-Verlag (2010).

Lehrig, S., Eikerling, H., Becker, S. Scalability, Elasticity, and Efficiency in Cloud Computing. Presentado en: *11th International ACM SIGSOFT Conference on Quality of Software Architectures - QoSA* (pp. 83–92). New York, New York, USA: ACM Press (2015).

Lew, P., Zhang, L., Wang, S., Jiang, W. Guidelines to Determine Quality for Web-Based Software Applications. Presentado en: *International Conference on Computational Intelligence for Modelling Control & Automation* (pp. 709–714). Vienna, Austria (2008).

Li, A., Yang, X., Kandula, S., Zhang, M. CloudCmp: Comparing Public Cloud Providers. Presentado en: *10th annual conference on Internet measurement - IMC* (p. 1). Melbourne, Australia (2010).

Li, Z., O'Brien, L., Zhang, H., Cai, R. On a catalogue of metrics for evaluating commercial cloud services. Presentado en: *13th International Conference on Grid Computing* (pp. 164–173). Beijing, China: IEEE (2012).

Liu, T., Lu, T., Wang, W., Wang, Q., Liu, Z., Gu, N., Ding, X. SDMS-O: A service deployment management system for optimization in clouds while guaranteeing users' QoS requirements. *Future Generation Computer Systems*, *28*(7), 1100–1109

(2012).

LogicMonitor (2008). http://ls.logicmonitor.com/why-logicmonitor/

López, A. *Configurador de la monitorización de la calidad de servicios en Google App Engine*. Universitat Politècnica de València (2016).

Ludwig, H., Keller, A. Web Service Level Agreement (WSLA) Language Specification, 1–110 (2003).

Lynch, M. *The Cloud Wars: $100 + billion at stake* (2008).

Mahbub, K., Spanoudakis, G. Monitoring WS-agreements: An event calculus-based approach. Presentado en: P. di M. Dipto. Elettronica Informazione (Ed.), *Test and Analysis of Web Services* (pp. 265–306). Milan, Italy: Springer Berlin Heidelberg (2007).

Mani, A., Nagarajan, A. Understanding quality of service for Web services (2002, January 1). http://www.ibm.com/developerworks/library/ws-quality/

Massonet, P., Naqvi, S., Ponsard, C., Latanicki, J., Rochwerger, B., Villari, M. A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures. Presentado en: *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum* (pp. 1510–1517). Anchorage, Alaska, USA (2011).

Maximilien, E. M., Singh, M. P. A framework and ontology for dynamic web, services selection. *IEEE Internet Computing*, *8*(5), 84–93 (2004).

McCall, J. a., Richards, P. K., Walters, G. F. Factors in Software Quality. *Nat'l Tech. Information Servicel*, *1, 2 and 3*(ADA049055) (1977).

McCarthy, J. *SLAs for Cloud Computing*. (P. Wieder, J. M. Butler, W. Theilmann, & R. Yahyapour, Eds.). NY: Springer (2011).

Mell, P., Grance, T. The NIST Definition of Cloud Computing. *Nist Special Publication*, *145*, 7 (2011).

Menascé, D. a., Casalicchio, E. QoS in grid computing. *IEEE Internet Computing*, *8*(4), 85–87 (2004).

Meng, S., Liu, L. Enhanced monitoring-as-a-service for effective cloud management. *IEEE Transactions on Computers*, *62*(9), 1705–1720 (2013).

Menychtas, A., Gogouvitis, S., Konstanteli, K., Gallizo, G., Kuebert, R., Movilla, J. M., Cucinotta, T. *Interactive Realtime Multimedia Applications on Service Oriented Infrastructures ICT FP7-214777 Guaranteeing QoS with Dynamic and Automated SLAs in real-time aware SOIs*. Atenas, Grecia (2009).

Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S. Comprehensive qos monitoring of web services and event-based sla violation detection. Presentado en: *4th international workshop on middleware for service oriented computing* (pp. 1–6). Urbana Champaign, Illinois, USA (2009).

Microsoft. Microsoft .NET Home (2016). http://www.microsoft.com/net

Microsoft Azure. What Is Azure? (2014). http://msdn.microsoft.com/en-us/library/azure/dd163896.aspx

Miguel, P. A., Salomi, G. E. A review of the models for measurement of the quality of services. *Magazine Production*, 12–30 (2004).

Mircea, T., Weiss, R., Huljenic, D., Graf, A., Crnkovic, I., Mirandola, R., Masetti, M. Q-ImPrESS (2010). http://www.q-impress.eu/wordpress/

Monitis (2014). http://www.monitis.com/contact

Montes, J., Sánchez, A., Memishi, B., Pérez, M. S., Antoniu, G. GMonE: A complete approach to cloud monitoring. *Future Generation Computer Systems*, *29*(8), 2026–2040 (2013).

Moody, D. L. *A Practical Method for Representing Large Entity Relationship Models, P.h.D. Thesis*. University of Melbourne, Australia (2001).

Moraga, M. A., Calero, C., Piattini, M., Walker, D. Towards a quality model for grid portals. Presentado en: *1 st International Conference on Software Technologies* (pp. 195–203). Setúbal, Portugal (2006).

Morfeo 4CaaSt (2013). http://www.4caast.eu/

Moses, J., Iyer, R., Illikkal, R., Srinivasan, S., Aisopos, K. Shared resource monitoring and throughput optimization in cloud-computing datacenters. Presentado en: *25th IEEE International Parallel and Distributed Processing Symposium, IPDPS* (pp. 1024–1033). Anchorage, Alaska, USA (2011).

Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Cortes, A., Rodriguez, M. Comprehensive Explanation of SLA Violations at Runtime. *IEEE Transactions on Service Computing*, *7*(2), 168–183 (2014).

Muntés-Mulero, V., Matthews, P., Omerovic, A., Gunka, A. Eliciting Risk, Quality and Cost Aspects in Multi-cloud Environments. Presentado en: *4th International Conference on Cloud Computing, GRIDs, and Virtualization*. Valencia, Spain (2013).

Myerson, J. Best practices to develop SLAs for cloud computing (2013). http://ibm.com/developerWorks/

Nagios - The Industry Standard In IT Infrastructure Monitoring (2007).

Nagios - The Industry Standard In IT Infrastructure Monitoring (2012). https://www.nagios.org/

Nallur, V., Bahsoon, R. A decentralized self-adaptation mechanism for service-based applications in the cloud. *IEEE Transactions on Software Engineering*, *39*(5), 591–612 (2013).

Nathuji, R., Kansal, A. Q-Clouds : Managing Performance Interference Effects for QoS-Aware Clouds. Presentado en: *5th European conference on Computer systems* (pp.

237–250). Paris, France (2010).

Navas Rosales, R. *Modelo de Calidad para Servicios Cloud*. Universitat Politècnica de València (2016).

Network & IT Systems Monitoring – Monitis (2005). http://www.monitis.com/

Ngan, L. D., S, T. F., Keong, C. C., Kanagasabai, R. Towards a common benchmark framework for cloud brokers. Presentado en: *18th International Conference on Parallel and Distributed Systems - ICPADS* (pp. 750–754). Singapore (2012).

Niehörster, O., Brinkmann, A., Keller, A., Kleineweber, C., Krüger, J., Simon, J. Cost-Aware and SLO-Fulfilling Software as a Service. *Journal of Grid Computing*, *10*(3), 553–577 (2012).

Nimsoft (2011). http://www.ca.com/us/products/ca-unified-infrastructure-management.html

O'Brien, L., Merson, P., Bass, L. Quality Attributes for Service-Oriented Architectures. Presentado en: *International Workshop on Systems Development in SOA Environments, SDSOA* (p. 3). Minneapolis, USA (2007).

Ocean. ETICS (2015). http://www.ocean-project.eu/bin/view/Services/Testing

Olsina, Luis; Lew, Philip; Dieser, A. et al. Updating Quality Models for Evaluating New Generation Web Applications. *Journal of Web Engineering*, *11*(3), 209–246 (2012).

Omg. Software & Systems Process Engineering Meta-Model Specification V2.0, (April), 236 (2008).

OpenNebula (2010). http://opennebula.org/

Optimis - Optimized Infrastructure Services (2013). http://www.optimis-project.eu/

Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. S., Wolf, A. L. An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems*, *3*(3), 54–62 (1999).

Oriol, M., Franch, X., Marco, J. SALMon : A SOA System for Monitoring Service Level Agreements, 1–12 (2010).

Oriol, M., Marco, J., Franch, X. Quality models for web services: A systematic mapping. *Information and Software Technology*, *56*(10), 1167–1182 (2014).

Oriol, M., Marco, J., Franch, X., Ameller, D. Monitoring Adaptable SOA-Systems using SALMon. Presentado en: *Workshop of Service Monitoring, Adaptation and Beyond, ServiceWave Conference, MONA*. Barcelona, Spain (2008).

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D. An Early Performance Analysis of Cloud Computing Services for Scientific Computing. *Computing*, *22*(December), 115–131 (2010).

Páez Rosales, A. *Monitor de Calidad de Servicios en Google App Engine*. Universitat Politècnica de València (2016).

Palacios, M., Garci, x, A-Fanjul, J., Tuya, J., de la Riva, C. A Proactive Approach to Test Service Level Agreements. Presentado en: *5th Conference on Software Engineering Advances (ICSEA)* (pp. 453–458). Nice, France (2010).

Panzieri, Fabio and Babaoglu, Ozalp and Ferretti, Stefano and Ghini, Vittorio and Marzolla, M. *Dependable and Historic Computing*. United Kingdom: Springer Berlin Heidelberg (2011).

Povedano-Molina, J., Lopez-Vega, J. M., Lopez-Soler, J. M., Corradi, A., Foschini, L. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, *29*(8), 2041–2056 (2013).

Quality, T., Challenge, M. Agile Product Quality. *Agile Product Quality Management*, 1–3 (2009).

R&D Project SERSCIS. SERSCIS - Semantically Enhanced Resilient and Secure Critical Infrastructure Services (2008). http://www.serscis.eu/

Radulovic, F., García-Castro, R. Extending Software Quality Models - A Sample In The Semantic Technologies Domain (2011).

Ran, S. A model for web services discovery with QoS. *ACM Sigecom Exchanges*, *4*(1), 1–10 (2003).

Rescher, N. *The Primacy of Practice*. Basil Blackwell (1973).

Reyes, S., Muñoz-Caro, C., Niño, A., Sirvent, R., Badia, R. M. Monitoring and steering Grid applications with GRID superscalar. *Future Generation Computer Systems*, *26*(4), 645–653 (2010).

Riemenschneider, C. K., Hardgrave, B. C., Davis, F. D. Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE Transactions on Software Engineering*, *28*(12), 1135–1145 (2002).

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I., … Galan, F. The Reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, *53*(4), 4:1–4:11 (2009).

Rosenberg, F., Leitner, P., Michlmayr, A., Celikovic, P., Dustdar, S. Towards Composition as a Service - A Quality of Service Driven Approach. Presentado en: *25th International Conference on Data Engineering* (pp. 1733–1740) (2009).

Samimi, P., Patel, A. Review of pricing models for grid & cloud computing. Presentado en: *IEEE Symposium on Computers and Informatics, ICSI* (pp. 634–639). Chongquing, China (2011).

Schad, J., Dittrich, J., Quiané-Ruiz, J.-A. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *The VLDB Endowment*, *3*(1-2), 460–471 (2010).

Schroeter, J., Cech, S., Götz, S., Wilke, C., A\ssmann, U. Towards Modeling a Variable Architecture for Multi-tenant SaaS-applications. Presentado en: *Proceedings of the*

Sixth International Workshop on Variability Modeling of Software-Intensive Systems (pp. 111–120). New York, NY, USA: ACM (2012).

Schubert, P., Dettling, W. Extended Web Assessment Method (EWAM) - evaluation of e-commerce applications from the customer's viewpoint. Presentado en: *35th Annual Hawaii International Conference on System Sciences, HICSS*. (p. 10 pp.). Hawaii, USA (2002).

Segars, A. H., Grover, V. Re-Examining Perceived Ease of Use and Usefulness: A Confirmatory Factor Analysis. *MIS Quarterly* (1993).

Shao, J., Wei, H., Wang, Q., Mei, H. A Runtime Model Based Monitoring Approach for Cloud. Presentado en: *Int.Conf. on Cloud Computing (CLOUD)* (pp. 313–320) (2010).

Shirey, R. Internet Security Glossary, Version 2. Presentado en: *Request for Comments (RFC)* (Vol. 4949, pp. 1–365) (2007).

Shull, F., Carver, J., Vegas, S., Juristo, N. The role of replications in empirical software engineering. *Empirical Software Engineering* (2008).

Shull, F., Mendoncça, M. G., Basili, V., Carver, J., Maldonado, J. C., Fabbri, S., Travassos, G. H., Ferreira, M. C. Knowledge-Sharing Issues in Experimental Software Engineering. *Empirical Software Engineering*, *9*(1/2), 111–137 (2004).

SLA@SOI (2008). http://sla-at-soi.eu/

Song, H., Huang, G., Chauvel, F., Sun, Y., Mei, H. SM@RT: Representing Run-time System Data As MOF-compliant Models. Presentado en: *32nd ACM/IEEE International Conference on Software Engineering* (pp. 303–304). Cape Town, South Africa (2010).

Song, H., Huang, G., Chauvel, F., Zhang, W., Sun, Y., Shao, W., Mei, H. Instant and incremental QVT transformation for runtime models. Presentado en: *14th International Conference, MODELS* (Vol. 6981 LNCS, pp. 273–288). Wellington, New Zealand (2011).

Song, H., Xiong, Y., Chauvel, F., Huang, G., Hu, Z. Generating Synchronization Engines between Running Systems and Their Model-Based Views. Presentado en: *International conference on Models in Software Engineering* (Vol. 509, pp. 140–154). Denver, CO, USA (2010).

SPAE (2002). http://www.shalb.com/en/spae/spae_features/

Spanoudakis, G., Mahbub, K. Non-Intrusive Monitoring of Service-Based Systems. *International Journal of Cooperative Information Systems*, *15*(03), 325–358 (2006).

Spring, J. Monitoring cloud computing by layer, Part 2. *IEEE Security and Privacy*, *9*(3), 52–55 (2011).

Stachowiak, H. *Allgemeine Modelltheorie*. Springer-Verlag (1973).

Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S. *Model-Driven Software Development: Technology, Engineering, Management*. Wiley (2006).

Szajna, B. Software Evaluation and Choice: Predictive Validation of the Technology Acceptance Instrument. *MIS Quarterly*, *18*(3), 319–324 (1994).

Szvetits, M., Zdun, U. Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*, 1–39 (2013).

The Cacti Group, I. Cacti (2012). www.cacti.net

Tolosana-Calasanz, R., Bañares, J. Á., Pham, C., Rana, O. F. Enforcing QoS in scientific workflow systems enacted over Cloud infrastructures. *Journal of Computer and System Sciences*, *78*(5), 1300–1315 (2012).

Torres, E., Segrelles, D., Blanquer, I., Hernández, V. Service monitoring and differentiation techniques for resource allocation in the grid, on the basis of the level of service. *Future Generation Computer Systems*, *27*(8), 1142–1152 (2011).

Truong, H. L., Samborski, R., Fahringer, T. Towards a framework for monitoring and analyzing QoS metrics of grid services. Presentado en: *2nd IEEE International Conference on e-Science and Grid Computing*. Amsterdam, Netherlands (2006).

Van der Meulen, R., Christy, P. Gartner Says Cloud Computing Will Be As Influential As E-business (2008). http://www.gartner.com/newsroom/id/707508

Varela, M., Technical, V. T. T. Challenges of QoE Management for Cloud Applications. *IEEE Communications Magazine*, *50*(April), 28–36 (2012).

Vasar, M., Srirama, S. N., Dumas, M. Framework for monitoring and testing web application scalability on the cloud. Presentado en: *International Conference on Software Architecture* (p. 53). Gothenburg, Sweden (2012).

Villegas, D., Sadjadi, S. M. Mapping Non-Functional Requirements to Cloud Applications. *SEKE Knowledge Systems Institute Graduate School*, 527–532 (2011).

Vogel, T., Giese, H. Adaptation and Abstract Runtime Models. Presentado en: *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS* (pp. 39–48). Cape Town, South Africa (2010).

Vogel, T., Holger, G. Language and Framework Requirements for Adaptation Models. Presentado en: *6th International Workshops on Models@run.time*. Wellington, New Zealand (2011).

Vogel, T., Neumann, S., Hildebrandt, S., Giese, H., Becker, B. Model-driven architectural monitoring and adaptation for autonomic systems. Presentado en: *International Conference on Autonomic Computing, ICAC* (pp. 67–68). Barcelona, Spain (2009).

Wagner, S. *Software Product Quality Control* (2013).

Wang, C., Schwan, K., Talwar, V., Eisenhauer, G., Hu, L., Wolf, M. A Flexible Architecture Integrating Monitoring and Analytics for Managing Large-scale Data Centers. Presentado en: *8th ACM International Conference on Autonomic Computing* (pp. 141–150). New York, NY, USA: ACM (2011).

Wang, X., Vitvar, T., Kerrigan, M., Toma, L. *A QoS-Aware Selection Model for Semantic Web Services*. (A. Dan & W. Lamersdorf, Eds.) (Vol. 4294). Berlin, Heidelberg: Springer Berlin Heidelberg (2006).

Watch, A. Elasticity as a Service Monitoring for Windows Azure (2016). https://www.paraleap.com/

Weins, K. Cloud computing Trends: 2016 State of the Cloud Survey (2016). http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey

Wen, P. X., Dong, L. Quality Model for Evaluating SaaS Service. Presentado en: *4th International Conference on Emerging Intelligent Data and Web Technologies, EIDWT* (pp. 83–87). Xi'an, China (2013).

Wind, S., Schrödl, H. Requirements Engineering for Cloud Computing: A Comparison Framework. Presentado en: *Workshop in Web Information Systems Engineering, WISE* (Vol. 6724, pp. 404–415). Doha, Qatar (2011).

Windows Azure Diagnostic Monitoring and Autoscaling (2016). https://www.paraleap.com/

Wohlin, C. Introduction to Aggregation of Case Studies Why aggregation. Sweden (2007).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A. *Exeperimentation in Software Engineering*. Springer Heidelberg New York Dordrecht London (2012).

Xiao, L., Yun, Y., Dong, Y., Gaofeng, Z., Wenhao, L., Dahai, C. A Generic QoS Framework for Cloud Workflow Systems. Presentado en: *9th International Conference on Dependable, Autonomic and Secure Computing, DASC* (pp. 713–720). Sydney, Australia (2011).

Yin, B., Yang, H., Fu, P., Chen, X. A Semantic Web Services Discovery Algorithm Based on QoS Ontology. *Active Media Technology*, *6335*, 166–173 (2010).

Zeng, W., Chang, G., Wang, X., Wang, S., Han, G., Zhou, X. A QoS Model for Grid Computing Based on DiffServ Protocol. Presentado en: *2nd International Workshop, GCC 2003,* (Vol. 3033, pp. 549–556). Shanghai, China (2004).

Zheng, X., Martin, P., Brohman, K., Xu, L. Da. Cloudqual: A quality model for cloud services. *IEEE Transactions on Industrial Informatics*, *10*(2), 1527–1536. article (2014).

Zhou, P., Wang, Z., Li, W., Jiang, N. Quality Model of Cloud Service. Presentado en: *7th International Symposium on Cyberspace Safety and Security (CSS)* (pp. 1418–1423). inproceedings, New York, USA (2015).

Anexo I Definición del proceso con SPEM 2

En este anexo se detalla cómo se utiliza la notación SPEM 2 (Software & System Process Engineering Meta-model Specification V2.0), propuesta por el grupo Object Management Group, que provee un marco formal para la definición de procesos de desarrollo de sistemas y de software, así como también para definir y describir sus elementos. El objetivo de su uso en este trabajo, es el proveer una definición detallada del proceso de monitorización de una manera simple y fácil de entender.

SPEM es parte de la Ingeniería de Procesos de Software (SEP), dedicándose a la definición, implementación, medición y mejora de los procesos de Ingeniería del Software. Está basada en MOF (Meta Object Facility), que es un estándar de la OMG, siendo SPEM a los procesos software lo mismo que UML a los sistemas software.

SPEM constituye un meta-modelo que permite definir modelos de procesos de Ingeniería del Software y de Ingeniería de sistemas. Este detalla los elementos mínimos necesarios que permitan definir dichos procesos sin añadir aspectos del dominio particular. La idea central de SPEM 2 está basada en tres elementos básicos: rol, producto de trabajo y tarea.

Tarea	Esfuerzo a realizar.
Rol	Quien realiza el esfuerzo.
Productos de trabajo:	Entradas que se utilizan y salidas que estos producen.

Tabla A I.1

Por medio de estos elementos se puede especificar "Quien (rol) realiza qué (tarea) para desde unas entradas conseguir unas salidas (productos de trabajo)".

SPEM 2 distingue dos etapas cuando se define un proceso:

1. Definición de los elementos de contenido que son los elementos primarios o constructores básicos.
    - Roles
    - Tareas.
    - Productos de trabajo.
    - Categorías.

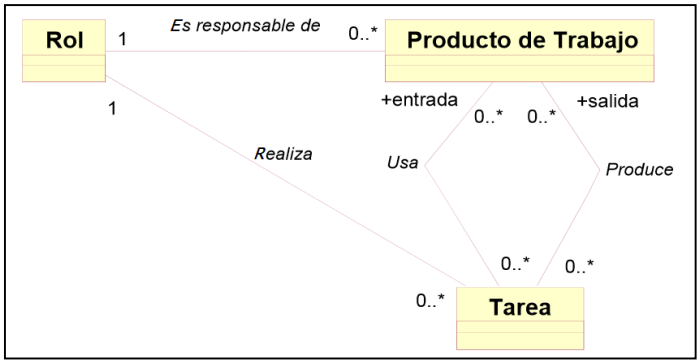


Figura A I.1 Metamodelo SPEM. (Fernández, 2012)

2. Luego se combinan y reutilizan los elementos esenciales para obtener los Procesos.

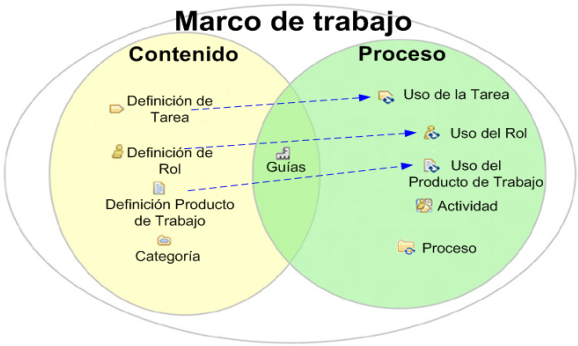


Figura A I.2 Mapeo contenido-proceso SPEM 2 (Fernández, 2012)

Ventajas de utilizar SPEM:

- Se puede disponer de modelos de Procesos de software en formato procesable por computador.
- Facilita la comprensión y comunicación entre las personas puesto que propicia un conocimiento homogéneo.
- Da soporte a la mejora de procesos.
- Da soporte a la gestión de procesos.
- Guía la automatización de procesos y da soporte para la ejecución automática.

A continuación se describe el subconjunto de primitivas de modelado que son las más comúnmente utilizadas a la hora de definir un proceso:

Tabla A I.2 Nomenclatura SPEM 2

Icono	Nombre	Descripción
	Definición de Rol	Conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo.
	Definición de Tarea	Unidad de trabajo asignable y gestionable, identificando el trabajo que se ejecuta por los roles. Puede dividirse en varios pasos.
	Definición de Producto de trabajo	Producto usado o producido por las *Tareas*. Existen dos tipos de productos: *Artefacto* de naturaleza tangible (modelo, documento, código, archivos, etc.) y *Entregable* para empaquetar productos con fines de entrega a un cliente interno o externo. Se pueden asociar entre ellos mediante relaciones de agregación, composición e impacto.
	Categoría	Clasificación de elementos como *Tareas, Roles* y *Productos* en base a los criterios que desee el ingeniero de procesos. Existen diversos tipos de categorías: *Conjunto de Roles* (para *Roles*), *Disciplina* (para *Tareas*), *Dominio* (para *Productos*),
	Guías	Información adicional relacionada con otros elementos. Los sub-tipos de guías pueden ser (entre otros): *Activo Reutilizable, Directriz, Documentación, plantillas.*
	Uso de Rol	Representación del r*ol* que lleva a cabo una *Tarea* o *Actividad* dentro de un proceso determinado. Hace referencia a una Definición de Rol (elemento de Contenido).
	Uso de Tarea	Representación de una *tarea* atómica dentro de un proceso determinado. Hace referencia a una Definición de Tarea (elemento de Contenido).
	Uso de Producto de Trabajo	Representación de un *Producto de Trabajo* de entrada o salida, relacionado con una *Actividad* o *Tarea*. Hace referencia a una Definición de un Producto de Trabajo (elemento de Contenido)
	Actividad	Representación de un conjunto de *Tareas* que se ejecutan dentro del proceso, junto con sus *Roles* y *Productos* asociados. Si únicamente se quiere representar una agrupación de tareas, se puede usar los elemento Actividad o Fase (incluido por retro-compatibilidad y más empleado en tareas de desarrollo), o bien si es un conjunto de tareas que se repite un determinado número de veces, se puede usar el elemento Iteración.
	Fase	
	Iteración	
	Paquete de Proceso	Representación de un paquete agrupando todos los elementos del proceso

Anexo II Modelo en Tiempo de Ejecución en XML

```
<?xml version="1.0"?>
<RuntimeModel xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <CloudService>
  <Name>CoCoMe</Name>
<connectionString>Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1S
ZFPTOtr/KBHBeksoGMGw==</connectionString>
  <deploymentID>fdc45005307044e0a00073e118ab71b0</deploymentID>
 </CloudService>
 <Characteristics>
  <Characteristic>
   <name>Reliability</name>
   <attributes>
    <Attribute>
     <name>Fault Tolerance</name>
     <metrics>
      <Metric>
       <name>Defective Operations Per Million (DPM)</name>
       <operationalization xsi:type="IndirectMetric">
        <name>DPM</name>
        <function>
         <formula>(([36]-[35])/[36])*1000000</formula>
         <operands>
          <DirectMetric>
                                    <name>\ASP.NET Applications(*)\Requests Total</name>
           <identifier>[36]</identifier>
           <extractionRate>4</extractionRate>
           <extractionType>0</extractionType>
          </DirectMetric>
          <DirectMetric>
 <name>\ASP.NET Applications(*)\Requests Succeeded </name>
           <identifier>[35]</identifier>
           <extractionRate>4</extractionRate>
           <extractionType>0</extractionType>
          </DirectMetric>
          <DirectMetric>
  <name>\ASP.NET Applications(*)\Requests Total</name>
           <identifier>[36]</identifier>
           <extractionRate>4</extractionRate>
           <extractionType>0</extractionType>
          </DirectMetric>
         </operands>
        </function>
       </operationalization>
      </Metric>
     </metrics>
     <nfr>
```

<name>**GuaranteeOfReliability**</name>
<nFRReference>**0**</nFRReference>
<attributes />
</nfr>
</Attribute>
</attributes>
<subCharacteristics />
</Characteristic>
<Characteristic>
<name>**Performance Efficiency**</name>
<attributes />
<subCharacteristics>
<Characteristic>
<name>**Time Behaviour**</name>
<attributes>
<Attribute>
<name>**Response Time**</name>
<metrics>
<Metric>
<name>**Latency**</name>
<operationalization xsi:type="**DirectMetric**">
<name>**\ASP.NET Applications(*)\Request Execution Time**</name>
<identifier>**[68]**</identifier>
<extractionRate>**3**</extractionRate>
<extractionType>**0**</extractionType>
</operationalization>
</Metric>
</metrics>
<nfr>
<name>**Latency**</name>
<nFRReference>**2**</nFRReference>
<attributes />
</nfr>
</Attribute>
</attributes>
<subCharacteristics />
</Characteristic>
</subCharacteristics>
</Characteristic>
</Characteristics>
<NFRs>
<NFR>
<name>**GuaranteeOfReliability**</name>
<nFRReference>**0**</nFRReference>
<attributes />
</NFR>
<NFR>
<name>**Latency**</name>
<nFRReference>**2**</nFRReference>

```
 <attributes />
</NFR>
</NFRs>
    </RuntimeModel>
```

Anexo III Instrumentación y material experimental

Guía de Aplicación de Cloud-Cloud MoS@RT

El método Cloud MoS@RT (Cloud Monitoring Services at Runtime) es un método de monitorización de la calidad de servicios en la nube a nivel de Software as a Service (SaaS). Los pasos a seguir en este método se presentan a continuación:

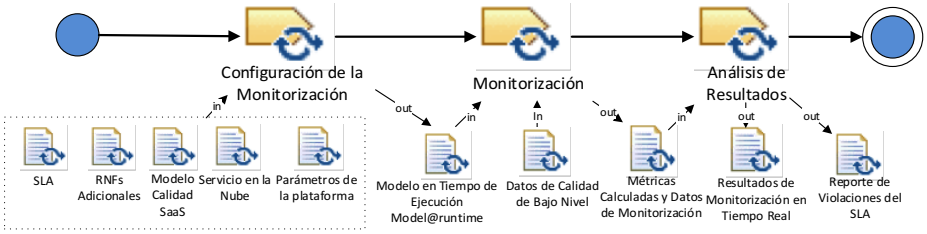


Figura A III.1 Proceso de Monitorización

Configuración de la Monitorización

Entradas

- Requisitos no funcionales (RNFs) a ser monitorizados (SLA + RNFs adicionales)
- Modelo de Calidad SaaS
- Servicio a monitorizar
- Parámetros de la plataforma

Tarea de configuración de la monitorización

Salida

- Modelo de Calidad en Tiempo de Ejecución

Sub-tareas de la Configuración de la Monitorización

Establecimiento de Requisitos de Monitorización de Calidad

Actor: Planificador de la Monitorización

Entradas

- Requisitos No Funcionales (RNFs) a ser monitorizados (SLA + RNFs adicionales)
- Servicio a monitorizar

Acciones

- Establecer los requisitos de monitorización (SLA + NFR adicionales)
- Especificar el servicio en la nube a ser monitorizado

Salida

- Modelo de Requisitos de Monitorización

Selección de Atributos de Calidad

Actor: Configurador de la Monitorización

Entradas

- Modelo de Requisitos de Monitorización
- Modelo de Calidad SaaS

Acciones

- Usar la lista de atributos de calidad (Modelo de Calidad SaaS) para identificar los RNFs a ser monitorizados y seleccionar los atributos correspondientes.

Salida

- Atributos de calidad seleccionados

Selección de Métricas de Calidad

Actor: Configurador de la Monitorización

Entradas

- Modelo de Requisitos de Monitorización
- Modelo de Calidad SaaS

Acciones

- Para cada atributo de calidad seleccionado en la tarea anterior, seleccionar una métrica para ser utilizada en la monitorización. La métrica debe ser ***equivalente*** a la métrica especificada en el Modelo de Requisitos de Monitorización.

Salida

- Métricas para la monitorización seleccionados

Mapeo de Métricas de Calidad

Actor: Configurador de la Monitorización

Entradas

- Atributos de calidad seleccionados
- Métricas seleccionadas
- Modelo de calidad SaaS
- Parámetros de bajo nivel

Acciones

Utilizando el configurador realizar el mapeo de cada uno de los atributos de calidad con sus respectivas métricas a sus equivalentes parámetros de bajo nivel. En este caso existirán diferentes escenarios:

- La métrica a mapear es una métrica directa, para lo cual existe un contador propio o personalizado con el cual directamente se realiza la monitorización. En ese caso seleccionar el parámetro de bajo nivel correspondiente dependiente de la plataforma.
- La métrica a mapear es una métrica indirecta (métrica que puede ser calculada a partir de métricas directas o a partir de otras métricas ya existentes), si la métrica indirecta ya ha sido definida, basta con seleccionarla, en el caso de no existir una métrica ya definida, se puede definir la función de medición. Para ello, se debe crear una nueva operacionalización y definir la operación matemática utilizando la herramienta de creación de fórmulas o funciones de medición.
- En caso de no existir la posibilidad de crear una operacionalización a partir de la información ya existente debido a que se trata de una métrica muy específica para el servicio dado, se pueden definir modos de extracción de información de monitorización personalizados en el servicio. Esto se consigue a través de programación, este tipo de información se puede agregar al configurador y seleccionarlo de entre la lista de operacionalizaciones de tipo envoltorio.
- Si se desea utilizar información proveniente de otras fuentes, se puede elegir de entre la lista de APIs, plugins, agentes o herramientas de terceros; las fuentes de datos con las que se extraerán los datos de monitorización.

Salida

- Métricas mapeadas con los parámetros propios de la plataforma, que servirán para recoger los datos de monitorización.

Generación del Modelo de Calidad en Tiempo de Ejecución

Actor: Configurador de la Monitorización

Entradas

- Métricas mapeadas

Acciones

- Utilizar el configurador (botón de generación de Modelo de Calidad en Tiempo de Ejecución) para generar el modelo.

Salida

- Modelo de calidad en tiempo de ejecución para la monitorización.

Proceso de Medición

Actor: Middleware de Monitorización & Análisis

Entradas

- Modelo de Calidad en Tiempo de Ejecución

Acciones

- Recoger los datos de monitorización desde los servicios desplegados en la nube.
- Ejecutar las mediciones de acuerdo a las métricas y sus funciones de medición especificadas en el modelo de calidad en tiempo de ejecución.
- Realizar las funciones de agregación necesarias para el cálculo de las métricas (promedios, percentiles, etc).
- Gestionar las bases de datos con los resultados obtenidos tras las mediciones.

Salida

- Métricas calculadas y datos de monitorización.

Análisis de Resultados

Actor: Middleware de Monitorización y Análisis

Entradas:

- Métricas calculadas y datos de monitorización

Acciones

- Mostrar los resultados de la monitorización en la pantalla en forma de gráficos estadísticos (ej. Tartas, Barras, Radares, etc).
- Comparar con los umbrales establecidos y generar reportes de cumplimiento del SLA.
- Generar reportes de monitorización generales.
- Realizar consultas a la base de datos en búsqueda de datos históricos.

Salida

- Resultados de la monitorización.

Fragmento del modelo de Calidad SaaS utilizado

Tabla A III.1. Fragmento de un Modelo de Calidad SaaS

Fragmento de un Modelo de Calidad SaaS				
Caract.	**Sub-Caract.**	**Atributos**	**Métricas**	**Operacionalizaciones**
Reliability		Maturity		
		Availability	Robustness of a Service (ROS)	Available Time for Invoking SaaS/Total Time for Operating SaaS
			Metric of Availability	(Agreed Service Time - Outage Downtime)/Agreed Service Time
				Uptime/Agreed Service Time
		Fault Tolerance	Defective Operations Per Million (DPM)	((Operations Attempted - Operations Successful)/Operations Attempted)*10^6
				(Operations Failed / Operations Attempted)*10^6
				(Oprations Failed / (Operations Successful+Operations Failed))*10^6
		Service Stability	Coverage of Fault Tolerance (CFT)	Number of Faults Without Become Failures / Total Faults Occured
			Coverage of Failure Recovery (CFR)	Number of Failures Remedied / Total Number of Failures
		Service Accuracy	Service Accuracy (SA)	Number of Correct Responses / Total Number of Requests
Perfor-mance Efficiency	Time Behavior	Response Time	Latency	Request Execution Time + Request Response Time
			Request Execution Time	Execution Time
			Time Behavior (TB)	Execution Time / Total Service Invocation Time
		Data Exchange Workload		
	Resource Utilization	Elasticity in Resources	Coverage of Scalability (COS)	Sum(Amount of Allocated Resources of ist Request / Total Amount of Requested Resources of ist Request)/Total Requests for Extending Resources Used
		Optimization in the use of resources		
	Capacity	Concurrent Users		
		Throughput of Services	Throughput	Number of Successful Transactions per Hour

Material para el entrenamiento

Lista de parámetros dependientes de la plataforma para el entrenamiento

En este Anexo hemos incluido los parámetros dependientes de la plataforma que más se aproximan a la función de medición. Hemos realizado este filtro debido a la gran cantidad de parámetros que una plataforma puede ofrecer. Por tanto a continuación encontrará para cada requisito no funcional a monitorizar algunos parámetros que pueden ser de utilidad para que usted pueda construir la función de medición apropiada.

NFR 1: $Execution\ Time = Request\ Execution\ Time$

CONTADORES APROXIMADOS
\ASP.NET\Request Execution Time
\Web Service(*)\Service Uptime
\ASP.NET\Requests Current
\ASP.NET Applications(*)\Sessions Active
\CUSTOM\Total Time Running Application

Tabla A III-2. Parámetros provenientes de la plataforma NFR1– Entrenamiento

NFR 2: $$SA = \frac{Number\ of\ Correct\ Responses}{Number\ of\ Total\ Responses}$$

CONTADORES APROXIMADOS
\ASP.NET Applications(*)\Errors Total
\ASP.NET Applications(*)\Requests Total
\ASP.NET Applications(*)\Requests Succeeded
\ASP.NET Applications(*)\Sessions Active
\ASP.NET Applications(*)\Requests Failed

Tabla A III-3. Parámetros provenientes de la plataforma NFR2– Entrenamiento

NFR MODIFICADO:

$$Latency = Request\ Execution\ Time + Request\ Response\ Time$$

CONTADORES APROXIMADOS
\ASP.NET\Request Execution Time
\Web Service(*)\Service Uptime
\ASP.NET\Request Wait Time
\ASP.NET Applications(*)\Sessions Active
\CUSTOM\Total Time Running Application

Tabla A III-4. Parámetros provenientes de la plataforma Modificación– Entrenamiento

Boletín de Entrenamiento

Método Cloud-Cloud MoS@RT

SaaS para Supermercados

Boletín de entrenamiento y caso de estudio planteado

Presentación del Método de Monitorización Cloud MoS@RT

En la documentación adjunta (Anexo 1), se encuentra la guía de aplicación de Cloud MoS@RT en la que se pueden consultar los detalles durante la realización del presente ejercicio.

Presentación de los Objetivos del Negocio

El objetivo del negocio que motiva el uso del método de monitorización corresponde a un conjunto de servicios referentes a las ventas y procesos de stock de una cadena de supermercados, cubriendo al departamento principal de la empresa (manejo central), tiendas (manejo local) y escritorios de pago (cajas). El uso de esta aplicación bajo el modelo SaaS provee algunos beneficios, por ejemplo, costos operacionales reducidos para el supermercado, reducción de los tiempos de espera en las cajas, elasticidad en tiempos críticos de ventas, etc. Las dos características a ser consideradas en este ejemplo son la eficiencia y la **precisión del mismo**.

En la Figura 1, se presenta una vista de la estructura del escenario concreto, donde se asume que el proveedor del servicios hace uso de un servicio bancario con un servicio externo, utiliza como infraestructura también servicios de IaaS provistos por terceros y además se muestran los servicios de software propios de la solución.

Los servicios que el proveedor SaaS ofrecerá para la configuración de estos sitios están descritos a continuación:

- Servicio de Inventario (Inventory Service): Este servicio está a cargo de presentar los productos que se comercializan en el supermercado.
- Servicio de Información de Tienda (Store Information Service): Provee una operación para recuperar información sobre la tienda, la cual incluye la información acerca de la sucursal y la empresa a la que pertenece.
- Servicio para órdenes de compra (Order Service): Este servicio es usado para controlar las órdenes de productos que están fuera de stock.
- Servicio de validación de tarjeta (Card Validation Service) es uno de los dos servicios que deben ser provistos por una entidad bancaria externa, su funcionalidad es la de validar una tarjeta de crédito.
- Servicio de débito de pago: Es el segundo servicio que es ofrecido por una entidad externa y controla si la tarjeta tiene fondos y realiza el débito.

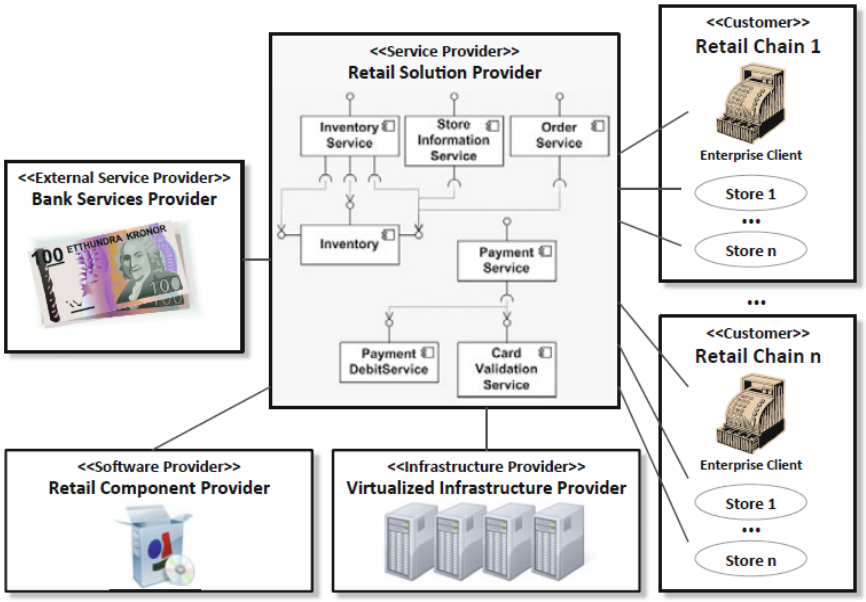


Figura A III.2 Vista de la provisión de servicios para supermercados

Los requisitos de monitorización a ser evaluados pueden pertenecer ya sea a los descritos en el SLA o a requisitos de monitorización adicionales que se necesiten evaluar dependiendo de las necesidades de las partes que intervienen en el contrato. A continuación se presenta los términos con los requisitos no funcionales a ser monitorizados en este ejercicio:

1. La eficiencia será medida a través del tiempo de ejecución, para ello se usará la métrica del tiempo de ejecución de la petición y se ha establecido que será de máximo 500 milisegundos y será calculada con la siguiente función de medición:

$$Execution\ Time = Request\ Execution\ Time$$

2. La confiabilidad será medida a través de la precisión del servicio (Service Accuracy – SA) será superior al 99.50% y será medida de la siguiente manera:

$$SA = \frac{Number\ of\ Correct\ Responses}{Number\ of\ Total\ Responses}$$

Pasos para la aplicación del método

Paso 1. Obtener el Modelo de Requisitos de Monitorización

Cargar el modelo de requisitos de monitorización al configurador. En este se encuentra la lista de requisitos no funcionales (NFRs) a ser monitorizados, métricas, umbrales, etc. Además, en el Anexo II de encuentra un resumen de los mismos, con sus métricas y explicación.

Tarea 1: Cargar los atributos de calidad a monitorizar y especificar la plataforma y el servicio en la nube.

1. Descargar el simulador del configurador. Este puede ser accedido desde la página web del ejercicio:

   (http://users.dsic.upv.es/~icedillo/Entrenamiento/)

2. Ejecutar el archivo de Excel del simulador y **permitir el uso de macros** (Habilitar contenido).

Figura A III.3. Página web sitio para materiales – Entrenamiento

3. Especificar la plataforma del servicio a ser monitorizado.
4. En el configurador seleccionar el servicio a ser monitorizado.
5. En la pantalla siguiente se encuentra ya cargado el modelo de requisitos de monitorización proporcionado para este ejercicio (Ejercicio > Anexo2. Métricas a Evaluar (Monitoring Requirements Model). El modelo de requisitos de monitorización contiene la lista de los NFRs a ser monitorizados, sus métricas y sus umbrales.

Paso 2 y 3. Seleccionar los Atributos de Calidad, sus Métricas y Operacionalizaciones

En el paso 2 y 3 se debe categorizar cada uno de los NFRs del Modelo de Requisitos de Monitorización con el correspondiente atributo de calidad contenido en el Modelo de Calidad SaaS así como escoger una métrica y su operacionalización equivalente a la establecida en el Modelo de Requisitos de Monitorización de entre las existentes en el Modelo de Calidad SaaS (Anexo III).

Tarea 2: Categorizar los atributos de calidad y seleccionar métricas y operacionalizaciones.

Utilizar el configurador para realizar la categorización de los atributos de calidad, sus métricas y operacionalizaciones.

Anote la hora de inicio (hh:mm): _____

Anote la hora de finalización (hh:mm): _____

Paso 4. Seleccionar o Construir la Métrica Específica de la Plataforma

En el paso 4, se deberá realizar un mapeo de la operacionalización escogida en el paso anterior con la función de medición dependiente de la plataforma que permita la captura de la información de bajo nivel desde los servicios desplegados en la nube.

En el Anexo IV encontrará una lista de parámetros dependientes de la plataforma que le permitirán construir la función de medición para la operacionalización, en caso de no existir. En el configurador, por facilidad de uso hemos realizado un "creador de fórmulas (funciones de medición)" y además hemos agrupado los posibles parámetros específicos de la plataforma (Performance Counters) de acuerdo a cada ejercicio. Están ubicados en la parte derecha de la pantalla como se muestra a continuación:

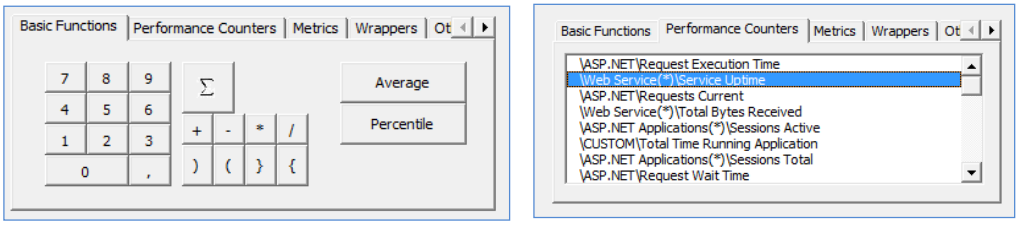


Figura A III.4. Modelo de negocio de sitio de subastas on-line

Tarea 3: Selección de métricas, operacionalizaciones y construcción de funciones de medición

Con cada requisito no funcional (NFR) de la tarea 2 realice las siguientes actividades:

Copiar el número del requisito no funcional del paso anterior.

Para cada métrica del paso anterior cree la función de medición correspondiente, utilizando los contadores del Anexo IV.

Anote la hora de inicio (hh:mm): _____

Anote la hora de finalización (hh:mm): _____

Actualización de Requisitos no funcionales

En caso de requerirse una modificación en los requisitos no funcionales a ser monitorizados, se podrá cambiar los mapeos establecidos previamente. Realizar el cambio propuesto en la tarea 4.

Tarea 4: Modificación de la Monitorización

De los NFRs de este ejercicio escoja el apropiado y realice las acciones correspondientes en caso de presentarse un cambio en los RNFs que se exprese de la siguiente manera:

En vista de que el tiempo de respuesta no se ha cumplido durante los últimos meses por fallos en el sistema de comunicaciones del proveedor de servicios, se ha visto necesaria una renegociación de un término del SLA que afecta al modelo de requisitos de monitorización, el nuevo término especifica que la eficiencia será medida por la latencia de la petición, que viene dada por la función de medición:

$$Latency = Request\ Execution\ Time + Request\ Response\ Time$$

Para realizar la modificación del requisito no funcional en cuestión, se asume el cambio en el Modelo de Requisitos de Monitorización y se debe hacer un mapeo con la nueva métrica. Por tanto se pide lo siguiente:

Anote la hora de inicio (hh:mm):

De entre los NFRs monitorizados, indique a cuál afectaría este cambio.

Anote el número del NFR: ____

Actualice las tablas de acuerdo a la nueva métrica y operacionalización seleccionada:

Métrica del Modelo de Calidad SaaS

Nombre de la Métrica: _____

Operacionalización / Función de Medición: _____

Operacionalización / Función de Medición dependiente de la plataforma: _____

Anote la hora de finalización (hh:mm): _____

Material para el cuasi-experimento

Lista de parámetros dependientes de la plataforma para el cuasi-experimento

En este Anexo hemos incluido los parámetros dependientes de la plataforma que más se aproximan a la función de medición. Hemos realizado este filtro debido a la gran cantidad de parámetros que una plataforma puede ofrecer. Por tanto a continuación encontrará para cada requisito no funcional a monitorizar algunos parámetros que pueden ser de utilidad para que usted pueda construir la función de medición apropiada.

NFR 1: $$\mathrm{ROS} = \frac{\mathrm{AvailableTimeforInvokingSaaS}}{\mathrm{TotalTimeforOperatingSaaS}}$$

CONTADORES APROXIMADOS
\ASP.NET\Request Execution Time
\Web Service(*)\Service Uptime
\ASP.NET\Requests Current
\Web Service(*)\Total Bytes Received
\ASP.NET Applications(*)\Sessions Active
\CUSTOM\Total Time Running Application
\ASP.NET Applications(*)\Sessions Total
\ASP.NET\Request Wait Time

Tabla A III-5. Parámetros provenientes de la plataforma NFR1– Cuasi-experimento

NFR 2: $$DPM = \frac{Operations\ Attempted\ -\ OperationsSuccessful}{Operations\ Attempted} * 10^6$$

CONTADORES APROXIMADOS
\ASP.NET Applications(*)\Errors Total
\ASP.NET Applications(*)\Requests Total
\ASP.NET Applications(*)\Requests Succeeded
\ASP.NET Applications(*)\Requests Timed Out
\ASP.NET Applications(*)\Sessions Active
\ASP.NET Applications(*)\Requests Failed
\ASP.NET\State Server Sessions Active
\ASP.NET\State Server Sessions Abandoned

Tabla A III-6. Parámetros provenientes de la plataforma NFR2– Cuasi-experimento

NFR 3: $$Latency = Request\ Execution\ Time + Response\ Time$$

NFR 3. CONTADORES APROXIMADOS
\ASP.NET\Application Restarts
\ASP.NET\Applications Running
\ASP.NET\Requests Disconnected
\ASP.NET\Request Execution Time
\ASP.NET\Requests Rejected
\ASP.NET\Requests Queued
\ASP.NET\Worker Processes Running
\ASP.NET\Request Wait Time

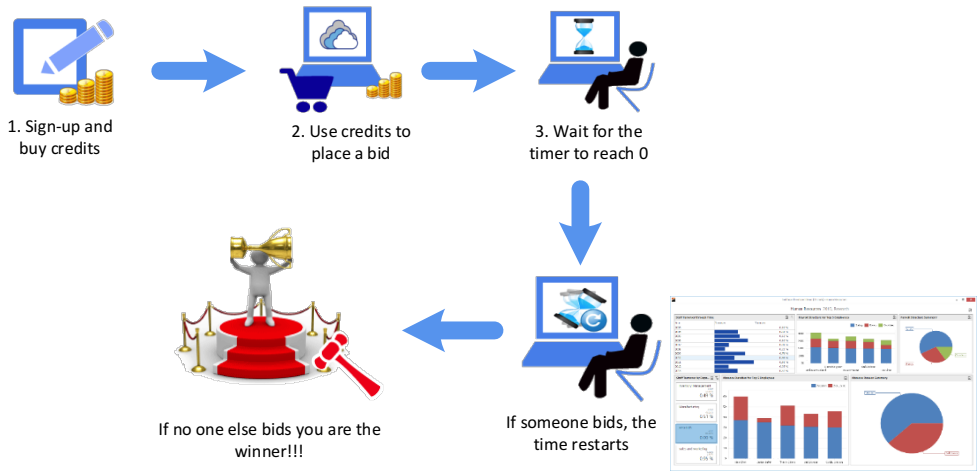
Tabla A III-7. Parámetros provenientes de la plataforma NFR3– Cuasi-experimento

NFR MODIFICADO:

$$Availability = \frac{Agreed\ Service\ Time - Outage\ Downtime}{Agreed\ Service\ Time}$$

NFR MODIFICADO. CONTADORES APROXIMADOS
\ASP.NET\Request Execution Time
\CUSTOM\Outage Downtime
\ASP.NET\Requests Current
\Web Service(*)\Total Bytes Received
\ASP.NET Applications(*)\Sessions Active
\CUSTOM\Total Service Time
\ASP.NET Applications(*)\Sessions Total
\ASP.NET\Request Wait Time

Tabla A III-8. Parámetros provenientes de la plataforma modificación– Cuasi-experimento



Boletín
Servicios de Subastas en Línea

Boletín para la ejecución del cuasi-experimento

Presentación del Método de Monitorización Cloud MoS@RT

En la documentación adjunta, se encuentra la guía de aplicación de Cloud MoS@RT en la que se pueden consultar los detalles durante la realización del presente ejercicio.

Presentación de los Objetivos del Negocio

El objetivo de negocio que motiva el uso del método de monitorización, es el control de la calidad de los servicios Cloud de un sitio de subastas online. Las características de calidad a ser monitorizadas están contenidas en el Anexo II. Para ello es necesario especificar de qué manera se configurarán los servicios para que estos atributos de calidad puedan ser monitorizados.

Un sitio de subastas en línea permite comprar y vender mercancías o servicios a través de pujas a modo de subasta, asignando un artículo o servicio al mejor postor como se indica en la Figura A III.5. ***Estos sitios necesitan contar con una serie de características de calidad***, entre las cuales se contemplan altos niveles de disponibilidad, elasticidad, precisión, etc. Los servicios de subasta pueden tener diferentes formatos, los más populares son las subastas directas (ascendentes) e inversas (descendentes). Un claro ejemplo de subasta descendente es de aquellas plataformas de compras públicas en entidades gubernamentales de muchos países para adquirir bienes y servicios más convenientes para el estado. Por otro lado, las subastas ascendentes son aquellas en las cuales el producto es adjudicado al mejor postor, estas han proliferado en Internet siendo el caso de Madbid, Ebay, etc.

Por tanto, en este ejercicio, asumiremos que el proveedor ofrecerá un conjunto de servicios que permitan configurar un sitio de subastas. Dependiendo de la necesidad del consumidor de los servicios, se podrán contratar uno u otro. En este caso configuraremos un sitio de subastas al mejor postor. Para que los usuarios puedan realizar una puja deben contar con créditos, los mismos que pueden ser adquiridos a través de un proceso de compra on-line. Se dará un tiempo durante el cual otro usuario podrá mejorar la puja, en el caso de que no sea mejorada y el tiempo haya transcurrido, el usuario que realizó la puja, adquirirá el producto al precio en el cual realizó la subasta. Una vez que el producto haya sido adjudicado, mediante un servicio de pago ofrecido por una entidad financiera, permitirá realizar el pago del mismo. Cada vez que se realice una puja, los créditos del usuario serán descontados, y una vez que estos se terminen se podrá usar la misma plataforma de pago para adquirir más crédito y poder participar en más pujas.

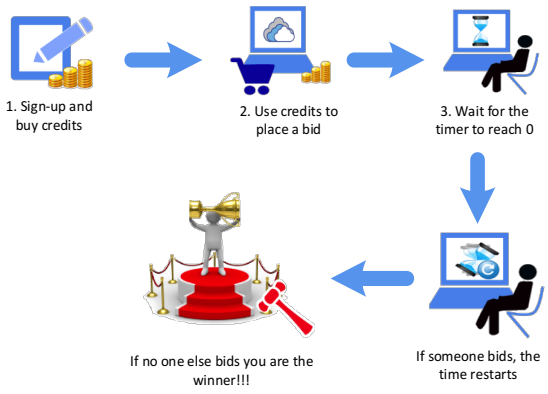


Figura A III.5. Modelo de negocio de sitio de subastas on-line

Los servicios que el proveedor SaaS ofrecerá para la configuración de estos sitios están descritos a continuación:

Servicio de Inventario: Este servicio está a cargo de presentar los productos que pueden ser vendidos durante el proceso de subasta.

Servicio de Subastas: Este servicio maneja las pujas, tiempo, precio actual, tiempo durante el cual la puja esté abierta, auto-pujas, etc. Este servicio es el central y debe presentar altas características de calidad.

Servicio de Crédito de Pujas: Este servicio maneja el crédito de los clientes que quieren participar en las subastas. En este caso el crédito está expresado por una moneda electrónica. El nombre más utilizado para esta moneda es el "*bid*". Los bids son usados al momento de la subasta para que el usuario pueda incrementar el valor de una puja.

Servicio de Usuarios: Este servicio maneja las cuentas de usuarios y el acceso al sistema.

Servicio de Manejo de Incidencias: Este servicio permite a los usuarios presentar posibles defectos, quejas o incidencias en general que se hayan presentado en el sitio de subastas.

Servicio de Pago: El servicio de pago es el encargado de verificar el balance de los créditos del usuario, así como también manejar el pago interno de los productos subastados. Este se conecta a un servicio externo, el cual maneja las divisas reales. El proveedor de pagos externo (p. ej. entidad bancaria, tarjeta de crédito, PayPal, etc) se encargará de soportar la transacción final con el usuario.

Los servicios que intervienen en este sitio, se muestran en la Figura A III.6 en el que se ilustra el diagrama del sitio de subastas y sus servicios principales:

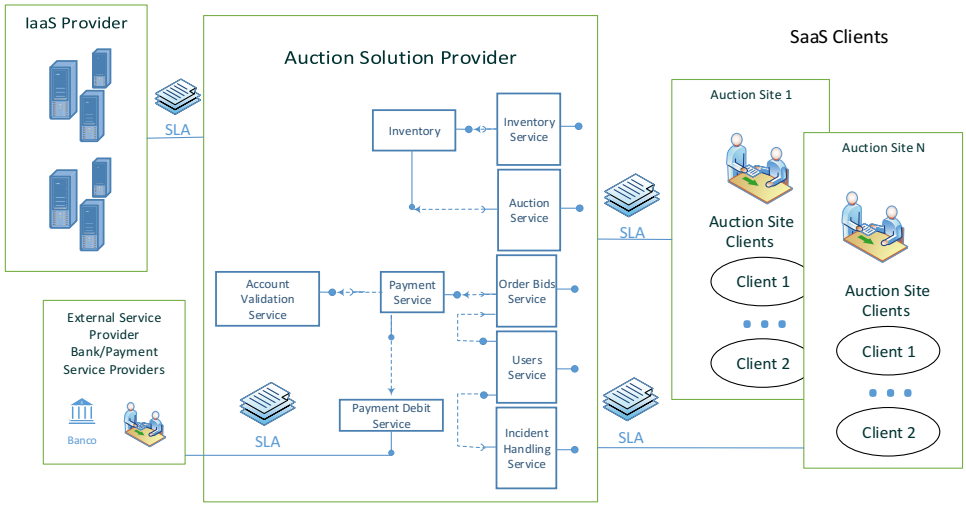


Figura A III.6. Servicios del sitio de subastas on-line

Los requisitos de monitorización a ser evaluados pertenecerán a los descritos en el SLA y a requisitos de monitorización adicionales que se necesiten evaluar dependiendo de las necesidades de las partes que intervienen en el contrato.

El servicio a ser evaluado será el servicio de subastas (Auction Service - Figura 2). Y la plataforma para la cual configuraremos la monitorización es Microsoft Azure.

A continuación se presentan los términos con los requisitos no funcionales a ser monitorizados en este ejercicio:

1. La disponibilidad (Availability) del servicio será superior al 99.995%. Y será calculada en base a la métrica robustez del servicio (Robustness of a Service - ROS) calculada con la siguiente función de medición:

$$ROS = \frac{Available\ Time\ for\ Invoking\ SaaS}{Total\ Time\ for\ Operating\ SaaS}$$

2. La confiabilidad (Reliability) será medida por el número de operaciones defectuosas por millón. En este caso, el servicio tendrá un máximo de 10 operaciones defectuosas por millón (99.999% de confiabilidad de servicio). Este requisito deberá ser calculado utilizando la métrica correspondiente (Defective Operations per Million- DPM)

$$DPM = \frac{Operations\ Attempted\ - OperationsSuccessful}{Operations\ Attempted} * 10^6$$

3. La latencia (Latency) máxima del servicio será de 500 milisegundos y será calculada con la siguiente función de medición:

$$Latency = Request\ Execution\ Time + Response\ Time$$

Pasos para la aplicación del método

Paso 1. Obtener el Modelo de Requisitos de Monitorización

Cargar el modelo de requisitos de monitorización al configurador. En este se encuentra la lista de requisitos no funcionales (NFRs) a ser monitorizados, métricas, umbrales, etc. Además, en el Anexo II se encuentra un resumen de los mismos, con sus métricas y explicación.

Tarea 1: Cargar los atributos de calidad a monitorizar y especificar la plataforma y el servicio en la nube.

1. Descargar el simulador de la configuración. Este puede ser accedido desde la página web del ejercicio:

   (http://users.dsic.upv.es/~icedillo/EjercicioCalidad/)

2. Ejecutar el archivo de Excel del simulador y permitir el uso de macros (Habilitar contenido).
3. Especificar la plataforma del servicio a ser monitorizado (Azure).
4. En el configurador seleccionar el servicio a ser monitorizado (Servicio de Subastas).

Figura A III.7. Página web para recolectar los materiales para el cuasi-experimento

5. En la pantalla siguiente se encuentra ya cargado el modelo de requisitos de monitorización proporcionado para este ejercicio (Un archivo pdf con los requisitos no funcionales puede también ser accedido a través de la página web del ejercicio en: Ejercicio Final > Anexo II. Métricas a Evaluar (Monitoring Requirements Model). El modelo de requisitos de monitorización contiene la lista de los NFRs a ser monitorizados, sus métricas y sus umbrales

Paso 2 y 3. Seleccionar los Atributos de Calidad, sus Métricas y Operacionalizaciones

En el paso 2 y 3 se debe categorizar cada uno de los NFRs del Modelo de Requisitos de Monitorización con el correspondiente atributo de calidad contenido en el Modelo de Calidad SaaS así como escoger una métrica y su operacionalización equivalente a la establecida en el Modelo de Requisitos de Monitorización de entre las existentes en el Modelo de Calidad SaaS (Anexo III).

Tarea 2: Categorizar los atributos de calidad y seleccionar métricas y operacionalizaciones.

Utilizar el configurador para realizar la categorización.

Anote la hora de inicio (hh:mm): _____

Anote la hora de finalización (hh:mm): _____

Paso 4. Seleccionar o Construir la Métrica Específica de la Plataforma

En el paso 4, se deberá realizar un mapeo de la operacionalización escogida en el paso anterior con la función de medición dependiente de la plataforma que permita la captura de la información de bajo nivel desde los servicios desplegados en la nube.

Adjunto a esta documentación, encontrará una lista de parámetros dependientes de la plataforma que le permitirán construir la función de medición para la operacionalización, en caso de no existir. En el configurador, por facilidad de uso hemos realizado un "creador de funciones de medición" y además hemos agrupado los posibles parámetros específicos de la plataforma (Performance Counters) de acuerdo a cada ejercicio.

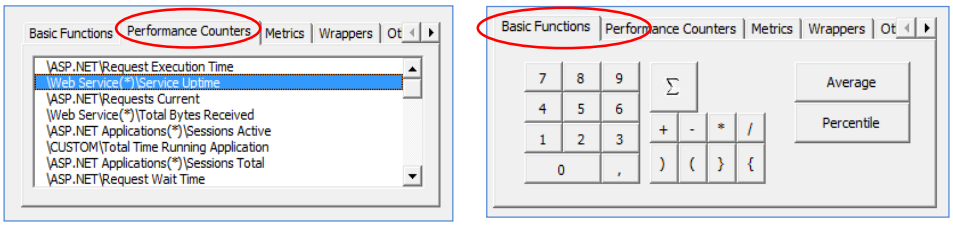


Figura A III.8. Vista para selección e contadores propios de la plataforma

Tarea 3: Selección de métricas, operacionalizaciones y construcción de funciones de medición

Con cada requisito no funcional de la tarea 2 realice las siguientes actividades:

1. Seleccione cada uno de los requisitos no funcionales del paso anterior.
2. Para cada métrica del paso anterior cree la función de medición correspondiente, utilizando los contadores del Anexo IV.

Anote la hora de inicio (hh:mm): _____

Anote la hora de finalización (hh:mm): _____

Actualización de Requisitos no funcionales

En caso de requerirse una modificación en los requisitos no funcionales a ser monitorizados, se podrá cambiar los mapeos establecidos previamente. Realizar el cambio propuesto en la tarea 4.

Tarea 4: Modificación de la Monitorización

Se ha producido una renegociación en el NFR de disponibilidad, de los NFRs de este ejercicio escoja el apropiado y realice las acciones apropiadas para que el NFR cambiado se exprese de la siguiente manera:

La disponibilidad (Availability) del servicio será superior al 99.995%. Y será calculada en base a la siguiente función de medición:

$$Availability = \frac{Agreed\ Service\ Time - Outage\ Downtime}{Agreed\ Service\ Time}$$

Anote la hora de inicio (hh:mm): _____

Para realizar la modificación del requisito no funcional en cuestión, se asume el cambio en el Modelo de Requisitos de Monitorización y se debe hacer un mapeo con la nueva métrica. Por tanto se pide lo siguiente:

De entre los NFRs monitorizados, indique a cuál afectaría este cambio.

Anote el número del NFR: ____

Actualice las tablas de acuerdo a la nueva métrica y operacionalización seleccionada:

Métrica del Modelo de Calidad SaaS

Nombre de la Métrica: _____

Operacionalización / Función de medición: _____

Operacionalización / Función de medición dependiente de la plataforma:_____

Anote la hora de finalización (hh:mm): _____

Tarea 5. Completar el cuestionario sobre el método y entrega de material

En este paso se deberá:

1. Completar un cuestionario sobre el método de monitorización de la calidad de servicios en la nube Cloud MoS@RT. El cuestionario se encuentra en el siguiente enlace: **http://goo.gl/forms/lBrWP6gEWL**
2. Entregar el boletín de la práctica.
3. Adjuntar en la tarea habilitada en PoliformaT los siguientes materiales:
    - Fichero Excel en el que se ha realizado la configuración.
    - Fichero TXT resultante de la configuración.

Anexo IV. Resultados de los cuasi-experimentos

Resultados cuasi-experimento 1 (UPV1)

ID	PEOU						PU							ITU				EFECTIVIDAD				EFICIENCIA / MINS.			
	P1	P3	P4	P6	P11	$\bar{x}$	P2	P5	P7	P9	P10	P12	$\bar{x}$	P8	P13	P14	$\bar{x}$	CAT.	MAP.	MOD.	$\bar{x}$	CAT.	MAP.	MOD.	Σ
1	2	4	3	4	5	3,60	4	4	5	4	3	4	4,00	4	4	5	4,33	1,00	1,00	1,00	1,00	11,00	7,00	5,00	23
2	4	4	4	3	3	3,60	5	5	4	5	4	4	4,50	5	4	5	4,67	1,00	1,00	1,00	1,00	3,00	4,00	7,00	14
3	2	2	2	2	2	2,00	4	2	3	3	3	3	3,00	2	3	4	3,00	1,00	1,00	0,33	0,78	13,00	8,00	11,00	32
4	4	4	4	5	5	4,40	3	3	3	4	4	4	3,50	4	4	4	4,00	1,00	0,67	1,00	0,89	5,00	6,00	7,00	18
5	4	3	4	4	3	3,60	4	4	4	3	4	3	3,67	3	4	4	3,67	1,00	1,00	0,67	0,89	3,00	12,00	6,00	21
6	4	4	4	3	4	3,80	4	4	4	2	3	4	3,50	3	3	3	3,00	1,00	0,33	1,00	0,78	8,00	10,00	11,00	29
7	3	4	1	3	4	3,00	4	2	4	3	3	4	3,33	5	2	4	3,67	1,00	0,67	0,17	0,61	7,00	15,00	7,00	29
8	4	4	4	4	4	4,00	5	5	4	5	5	4	4,67	4	4	4	4,00	1,00	0,67	1,00	0,89	22,00	15,00	8,00	45
9	2	5	3	3	3	3,20	4	5	5	5	5	5	4,83	4	4	5	4,33	1,00	0,00	1,00	0,67	11,00	3,00	9,00	23
10	3	3	4	5	4	3,80	4	4	4	2	3	4	3,50	3	3	4	3,33	1,00	1,00	1,00	1,00	10,00	14,00	6,00	30
11	5	4	4	4	5	4,40	5	5	5	4	4	5	4,67	4	4	5	4,33	1,00	1,00	1,00	1,00	5,00	16,00	11,00	32
12	3	4	4	5	2	3,60	4	3	5	5	3	4	4,00	3	4	3	3,33	1,00	1,00	0,67	0,89	6,00	8,00	6,00	20
13	3	4	4	4	5	4,00	4	4	4	3	4	4	3,83	4	4	5	4,33	1,00	1,00	1,00	1,00	9,00	8,00	5,00	22
14	4	4	4	4	5	4,20	3	5	3	5	3	4	3,83	5	5	5	5,00	1,00	0,67	1,00	0,89	11,00	15,00	14,00	40
15	4	5	4	4	5	4,40	4	4	4	4	3	4	3,83	3	4	4	3,67	1,00	1,00	1,00	1,00	3,00	7,00	8,00	18
16	3	3	4	3	4	3,40	3	3	4	3	3	3	3,17	2	3	3	2,67	1,00	1,00	1,00	1,00	11,00	13,00	11,00	35
17	2	3	2	3	3	2,60	2	4	4	3	4	3	3,33	4	3	3	3,33	1,00	0,00	0,00	0,33	20,00	14,00	10,00	44
18	4	4	5	4	5	4,40	2	4	4	4	5	5	4,00	4	4	2	3,33	1,00	1,00	1,00	1,00	10,00	14,00	6,00	30
19	4	3	4	4	2	3,40	3	4	4	3	4	4	3,67	3	3	4	3,33	1,00	1,00	1,00	1,00	4,00	9,00	10,00	23
20	2	1	3	2	4	2,40	5	5	5	4	4	4	4,50	5	4	5	4,67	1,00	1,00	1,00	1,00	20,00	14,00	10,00	44
21	4	4	4	4	4	4,00	5	5	5	4	5	4	4,67	4	4	4	4,00	1,00	1,00	1,00	1,00	5,00	17,00	18,00	40
22	4	4	5	4	4	4,20	4	5	4	3	4	5	4,17	4	4	5	4,33	1,00	1,00	1,00	1,00	5,00	5,00	14,00	24

23	4	4	5	4	4	4,20	4	5	4	3	5	4	4,17	4	4	5	4,33	1,00	1,00	1,00	1,00	3,00	7,00	9,00	19
24	4	4	5	5	5	4,60	4	4	3	3	3	5	3,67	4	4	4	4,00	1,00	1,00	1,00	1,00	5,00	8,00	8,00	21
25	5	4	1	4	5	3,80	5	5	5	1	5	1	3,67	2	4	2	2,67	1,00	1,00	0,83	0,94	4,00	7,00	4,00	15
26	4	5	4	4	5	4,40	3	5	4	1	2	3	3,00	3	4	4	3,67	1,00	1,00	1,00	1,00	6,00	16,00	9,00	31
27	4	2	2	2	2	2,40	1	2	2	2	2	1	1,67	2	2	1	1,67	1,00	1,00	1,00	1,00	2,00	9,00	9,00	20
28	2	2	3	2	3	2,40	3	3	3	2	3	4	3,00	2	3	3	2,67	1,00	1,00	0,67	0,89	3,00	9,00	5,00	17
29	4	5	4	4	5	4,40	4	4	4	3	3	4	3,67	3	3	4	3,33	1,00	1,00	1,00	1,00	4,00	12,00	7,00	23
30	4	4	4	5	5	4,40	5	5	4	4	4	5	4,50	4	4	4	4,00	1,00	1,00	1,00	1,00	7,00	9,00	7,00	23
31	4	2	3	4	4	3,40	4	4	4	4	4	4	4,00	4	4	4	4,00	1,00	1,00	0,83	0,94	15,00	5,00	5,00	25
32	3	4	4	4	4	3,80	3	3	4	4	4	4	3,67	3	3	4	3,33	1,00	1,00	1,00	1,00	5,00	5,00	7,00	17
33	2	3	2	3	3	2,60	2	4	4	3	4	5	3,67	2	2	3	2,33	1,00	1,00	1,00	1,00	6,00	6,00	7,00	19
34	4	4	4	4	4	4,00	4	3	4	4	3	4	3,67	3	5	5	4,33	1,00	1,00	1,00	1,00	6,00	9,00	6,00	21
35	5	5	5	5	3	4,60	4	4	4	3	4	4	3,83	4	5	4	4,33	1,00	1,00	1,00	1,00	4,00	6,00	7,00	17
36	4	5	5	5	5	4,80	4	4	4	4	4	4	4,00	4	4	4	4,00	1,00	1,00	1,00	1,00	5,00	10,00	8,00	23
37	5	4	5	4	5	4,60	4	5	4	4	4	4	4,17	4	4	4	4,00	1,00	1,00	1,00	1,00	5,00	8,00	12,00	25
38	4	3	4	4	4	3,80	4	4	4	3	3	4	3,67	4	4	4	4,00	1,00	1,00	1,00	1,00	3,00	10,00	6,00	19
39	4	5	2	4	2	3,40	5	3	5	1	2	3	3,17	2	2	4	2,67	1,00	1,00	0,83	0,94	10,00	11,00	11,00	32
40	3	3	3	5	5	3,80	4	4	4	4	4	4	4,00	4	4	4	4,00	1,00	1,00	0,83	0,94	6,00	5,00	10,00	21
41	4	4	4	5	5	4,40	5	5	4	4	3	5	4,33	4	3	4	3,67	1,00	1,00	1,00	1,00	6,00	18,00	7,00	31
42	2	3	2	1	1	1,80	2	3	3	3	3	3	2,83	2	1	2	1,67	1,00	1,00	1,00	1,00	4,00	20,00	5,00	29
43	5	5	5	5	4	4,80	5	5	4	4	4	4	4,33	5	5	5	5,00	1,00	1,00	1,00	1,00	3,00	10,00	5,00	18
44	4	3	3	3	2	3,00	3	4	2	3	3	2	2,83	4	3	3	3,33	1,00	1,00	0,67	0,89	9,00	15,00	10,00	34
45	4	3	3	3	3	3,20	2	4	5	5	4	5	4,17	4	4	4	4,00	1,00	1,00	1,00	1,00	5,00	8,00	6,00	19
46	4	4	4	4	4	4,00	4	5	4	4	5	4	4,33	5	5	4	4,67	1,00	1,00	1,00	1,00	9,00	18,00	4,00	31
47	4	3	4	2	5	3,60	4	4	4	4	5	4	4,17	5	4	5	4,67	1,00	0,67	1,00	0,89	3,00	21,00	8,00	32
48	4	3	5	5	5	4,40	4	4	4	5	5	4	4,33	4	5	5	4,67	1,00	1,00	1,00	1,00	5,00	17,00	10,00	32
49	4	4	4	4	5	4,20	5	5	5	4	4	4	4,50	4	5	5	4,67	1,00	1,00	1,00	1,00	6,00	17,00	8,00	31
50	3	4	4	4	2	3,40	4	4	3	2	3	4	3,33	4	3	4	3,67	1,00	1,00	0,00	0,67	18,00	10,00	6,00	34
51	4	3	4	3	4	3,60	3	4	4	4	3	3	3,50	3	4	4	3,67	1,00	0,67	0,00	0,56	13,00	15,00	4,00	32

52	4	4	4	4	4	4,00	5	4	5	4	4	1	3,83	4	4	4	4,00	1,00	0,33	1,00	0,78	5,00	10,00	10,00	25
53	2	3	3	2	2	2,40	2	3	4	1	1	1	2,00	1	1	1	1,00	1,00	1,00	0,67	0,89	5,00	4,00	10,00	19
54	2	2	2	4	2	2,40	4	4	4	3	3	3	3,50	4	3	3	3,33	1,00	0,33	1,00	0,78	8,00	10,00	9,00	27
55	2	2	3	2	3	2,40	3	2	2	4	3	5	3,17	5	4	2	3,67	1,00	0,00	0,67	0,56	6,00	15,00	7,00	28
56	1	2	2	2	1	1,60	3	2	2	2	2	2	2,17	2	1	1	1,33	1,00	1,00	1,00	1,00	15,00	5,00	8,00	28
57	4	3	2	4	4	3,40	3	3	3	3	3	3	3,00	3	3	3	3,00	1,00	1,00	1,00	1,00	4,00	6,00	5,00	15
58	2	3	2	4	3	2,80	4	5	4	4	5	2	4,00	2	5	4	3,67	1,00	1,00	0,67	0,89	20,00	26,00	10,00	56

Resultados cuasi-experimento 2 (UNA)

ID	PEOU						PU							ITU				EFECTIVIDAD				EFICIENCIA / MIN.			
	P1	P3	P4	P6	P11	$\bar{x}$	P2	P5	P7	P9	P10	P12	$\bar{x}$	P8	P13	P14	$\bar{x}$	CAT.	MAP.	MOD.	$\bar{x}$	CAT.	MAP.	MOD.	Σ
1	2	4	3	4	5	3,60	4	4	5	4	3	4	4,00	4	4	5	4,33	1,00	1,00	1,00	1,00	11,00	7,00	5,00	23
2	4	4	4	3	3	3,60	5	5	4	5	4	4	4,50	5	4	5	4,67	1,00	1,00	1,00	1,00	3,00	4,00	7,00	14
3	2	2	2	2	2	2,00	4	2	3	3	3	3	3,00	2	3	4	3,00	1,00	1,00	0,33	0,78	13,00	8,00	11,00	32
4	4	4	4	5	5	4,40	3	3	3	4	4	4	3,50	4	4	4	4,00	1,00	0,67	1,00	0,89	5,00	6,00	7,00	18
5	4	3	4	4	3	3,60	4	4	4	3	4	3	3,67	3	4	4	3,67	1,00	1,00	0,67	0,89	3,00	12,00	6,00	21
6	4	4	4	3	4	3,80	4	4	4	2	3	4	3,50	3	3	3	3,00	1,00	0,33	1,00	0,78	8,00	10,00	11,00	29
7	3	4	1	3	4	3,00	4	2	4	3	3	4	3,33	5	2	4	3,67	1,00	0,67	0,17	0,61	7,00	15,00	7,00	29
8	4	4	4	4	4	4,00	5	5	4	5	5	4	4,67	4	4	4	4,00	1,00	0,67	1,00	0,89	22,00	15,00	8,00	45
9	2	5	3	3	3	3,20	4	5	5	5	5	5	4,83	4	4	5	4,33	1,00	0,00	1,00	0,67	11,00	3,00	9,00	23
10	3	3	4	5	4	3,80	4	4	4	2	3	4	3,50	3	3	4	3,33	1,00	1,00	1,00	1,00	10,00	14,00	6,00	30
11	5	4	4	4	5	4,40	5	5	5	4	4	5	4,67	4	4	5	4,33	1,00	1,00	1,00	1,00	5,00	16,00	11,00	32
12	3	4	4	5	2	3,60	4	3	5	5	3	4	4,00	3	4	3	3,33	1,00	1,00	0,67	0,89	6,00	8,00	6,00	20
13	3	4	4	4	5	4,00	4	4	4	3	4	4	3,83	4	4	5	4,33	1,00	1,00	1,00	1,00	9,00	8,00	5,00	22
14	4	4	4	4	5	4,20	3	5	3	5	3	4	3,83	5	5	5	5,00	1,00	0,67	1,00	0,89	11,00	15,00	14,00	40

Resultados cuasi-experimento 3 (UC)

ID	PEOU	PU	ITU	EFECTIVIDAD	EFICIENCIA / MINS.

	P1	P3	P4	P6	P11	$\bar{X}$	P2	P5	P7	P9	P10	P12	$\bar{X}$	P8	P13	P14	$\bar{X}$	CAT.	MAP.	MOD.	$\bar{X}$	CAT.	MAP.	MOD.	Σ
1	5	5	4	4	5	4,60	5	5	5	5	5	5	5,00	5	5	5	5,00	1,00	1,00	1,00	1,00	5,00	9,00	6,00	20
2	2	3	5	3	4	3,40	5	4	5	4	4	4	4,33	4	4	4	4,00	1,00	0,67	1,00	0,89	3,00	9,00	5,00	17
3	4	4	5	5	5	4,60	5	5	4	4	4	4	4,33	4	5	5	4,67	1,00	0,67	0,00	0,56	4,00	15,00	0,00	19
4	3	5	4	5	4	4,20	4	5	4	4	4	4	4,17	4	4	4	4,00	1,00	1,00	1,00	1,00	9,00	7,00	6,00	22
5	2	4	4	3	3	3,20	1	2	2	4	5	4	3,00	3	4	5	4,00	1,00	0,67	0,67	0,78	1,00	17,00	9,00	27
6	4	4	4	2	5	3,80	5	5	5	5	4	5	4,83	5	5	5	5,00	1,00	1,00	1,00	1,00	4,00	10,00	8,00	22
7	3	1	5	3	2	2,80	4	4	4	2	3	5	3,67	4	4	4	4,00	1,00	1,00	0,67	0,89	1,00	3,00	25,00	29
8	3	4	3	3	4	3,40	5	5	5	5	5	5	5,00	4	4	5	4,33	1,00	1,00	0,67	0,89	6,00	9,00	9,00	24
9	4	4	3	3	5	3,80	4	5	5	4	4	2	4,00	5	4	5	4,67	1,00	0,67	0,00	0,56	11,00	12,00	7,00	30
10	5	3	1	4	4	3,40	1	5	5	5	5	5	4,33	4	4	5	4,33	1,00	0,67	0,67	0,78	4,00	10,00	9,00	23

Resultados cuasi-experimento 4 (UPV2)

ID	PEOU						PU							ITU				EFECTIVIDAD				EFICIENCIA / MINS.			
	P1	P3	P4	P6	P11	$\bar{X}$	P2	P5	P7	P9	P10	P12	$\bar{X}$	P8	P13	P14	$\bar{X}$	CAT.	MAP.	MOD.	$\bar{X}$	CAT.	MAP.	MOD.	Σ
1	4	5	5	4	5	4,60	4	5	5	5	5	5	4,83	4	5	5	4,67	1,00	1,00	1,00	1,00	8,00	11,00	9,00	28
2	2	2	2	2	2	2,00	4	3	4	2	3	4	3,33	1	5	1	2,33	1,00	0,87	0,67	0,84	28,00	24,00	7,00	59
3	4	4	4	4	5	4,20	3	4	4	5	3	5	4,00	3	4	5	4,00	1,00	1,00	1,00	1,00	32,00	16,00	16,00	64
4	2	2	5	3	1	2,60	2	3	3	1	3	2	2,33	1	2	2	1,67	1,00	1,00	1,00	1,00	26,00	18,00	7,00	51
5	2	2	5	3	5	3,40	4	5	4	2	4	3	3,67	4	3	1	2,67	1,00	1,00	1,00	1,00	16,00	20,00	10,00	46
6	3	3	2	2	2	2,40	3	2	3	1	3	2	2,33	2	2	2	2,00	1,00	0,87	1,00	0,96	14,00	22,00	7,00	43
7	3	3	2	2	2	2,40	4	3	3	4	3	3	3,33	3	3	4	3,33	1,00	0,87	0,67	0,85	18,00	12,00	15,00	45
8	3	3	3	3	5	3,40	4	4	4	2	4	2	3,33	3	3	3	3,00	1,00	0,87	0,67	0,85	20,00	9,00	8,00	37
9	3	3	5	4	4	3,80	4	5	4	4	4	4	4,17	4	4	4	4,00	0,87	0,87	0,67	0,80	19,00	13,00	8,00	40
10	1	3	2	3	3	2,40	2	3	3	4	2	3	2,83	2	2	3	2,33	1,00	0,67	0,67	0,78	32,00	8,00	7,00	47
11	2	2	2	2	4	2,40	2	3	3	3	2	3	2,67	1	3	3	2,33	1,00	1,00	1,00	1,00	3,00	4,00	12,00	19
12	4	2	4	2	4	3,20	4	4	4	2	4	2	3,33	4	4	2	3,33	1,00	1,00	1,00	1,00	23,00	17,00	6,00	46

13	4	4	3	3	2	3,20	3	3	2	3	2	3	2,67	2	2	3	2,33	1,00	0,87	1,00	0,96	21,00	19,00	12,00	52
14	2	2	1	1	1	1,40	3	4	3	3	3	3	3,17	3	3	3	3,00	1,00	0,87	1,00	0,96	15,00	22,00	10,00	47
15	3	3	3	3	4	3,20	4	4	3	3	4	4	3,67	3	4	4	3,67	0,53	0,17	0,67	0,46	22,00	18,00	6,00	46
16	3	3	4	3	3	3,20	3	4	3	3	4	3	3,33	4	3	3	3,33	1,00	1,00	1,00	1,00	16,00	14,00	4,00	34
17	2	2	2	2	4	2,40	4	4	4	3	4	3	3,67	3	4	4	3,67	1,00	0,87	0,67	0,85	21,00	26,00	8,00	55
18	4	3	2	3	4	3,20	3	4	3	2	4	2	3,00	3	3	3	3,00	1,00	1,00	1,00	1,00	31,00	19,00	6,00	56
19	4	4	2	3	3	3,20	4	3	4	4	2	3	3,33	3	4	3	3,33	1,00	1,00	1,00	1,00	32,00	35,00	13,00	80
20	5	3	5	4	3	4,00	3	4	2	4	3	5	3,50	3	4	4	3,67	0,87	1,00	1,00	0,96	30,00	11,00	8,00	49
21	3	2	3	2	4	2,80	4	4	4	4	4	3	3,83	4	4	5	4,33	0,87	0,87	1,00	0,91	35,00	20,00	15,00	70
22	3	4	2	3	4	3,20	2	4	4	3	3	3	3,17	3	2	4	3,00	1,00	1,00	1,00	1,00	14,00	20,00	15,00	49
23	4	4	4	3	4	3,80	3	5	5	2	5	1	3,50	5	5	1	3,67	1,00	1,00	1,00	1,00	20,00	10,00	5,00	35
24	4	3	3	4	4	3,60	3	3	3	4	3	3	3,17	4	4	5	4,33	1,00	1,00	1,00	1,00	27,00	11,00	8,00	46
25	3	3	3	4	4	3,40	4	4	2	4	3	4	3,50	3	3	2	2,67	1,00	1,00	1,00	1,00	60,00	11,00	6,00	77
26	3	2	3	3	4	3,00	3	4	4	4	4	4	3,83	3	4	4	3,67	0,87	0,87	1,00	0,91	27,00	22,00	14,00	63
27	4	4	4	3	5	4,00	1	5	5	5	1	5	3,67	5	5	5	5,00	0,87	0,87	1,00	0,91	16,00	13,00	5,00	34
28	4	4	4	5	5	4,40	4	3	4	3	3	4	3,50	4	4	3	3,67	0,87	0,87	1,00	0,91	16,00	9,00	13,00	38
29	3	4	3	4	4	3,60	4	4	4	4	4	4	4,00	4	4	5	4,33	1,00	1,00	1,00	1,00	38,00	11,00	11,00	60
30	2	2	3	2	1	2,00	3	3	2	2	2	2	2,33	1	2	3	2,00	1,00	0,87	1,00	0,96	17,00	13,00	5,00	35
31	3	3	3	3	3	3,00	3	3	3	3	3	3	3,00	3	3	3	3,00	0,87	0,87	1,00	0,91	22,00	30,00	9,00	61
32	4	4	5	4	5	4,40	5	4	5	5	4	4	4,50	5	5	5	5,00	0,87	0,87	1,00	0,91	19,00	11,00	11,00	41
33	4	5	1	4	3	3,40	4	3	3	2	3	2	2,83	4	3	3	3,33	1,00	1,00	1,00	1,00	17,00	12,00	4,00	33
34	5	4	5	5	4	4,60	5	5	4	4	5	5	4,67	4	4	5	4,33	1,00	1,00	1,00	1,00	23,00	21,00	8,00	52
35	1	4	2	4	4	3,00	3	2	3	4	4	4	3,33	4	4	4	4,00	0,87	0,20	0,67	0,58	20,00	22,00	9,00	51
36	3	4	4	3	4	3,60	3	2	3	3	4	3	3,00	3	3	2	2,67	1,00	1,00	1,00	1,00	36,00	16,00	8,00	60
37	3	1	2	2	3	2,20	2	3	2	2	2	3	2,33	1	3	3	2,33	0,87	0,87	1,00	0,91	10,00	28,00	20,00	58
38	1	1	1	1	1	1,00	1	1	2	1	1	1	1,17	1	1	1	1,00	0,87	1,00	1,00	0,96	42,00	20,00	10,00	72
39	2	3	4	3	3	3,00	4	3	2	2	2	4	2,83	4	2	4	3,33	0,87	0,87	1,00	0,91	8,00	21,00	11,00	40
40	1	1	1	1	1	1,00	2	2	2	1	2	2	1,83	1	1	1	1,00	0,87	0,67	0,67	0,73	36,00	11,00	7,00	54
41	1	1	2	3	3	2,00	2	4	3	2	3	3	2,83	4	3	2	3,00	0,87	0,87	1,00	0,91	39,00	22,00	10,00	71

42	3	4	3	2	2	2,80	4	4	4	2	2	2	3,00	3	3	2	2,67	0,87	1,00	1,00	0,96	29,00	10,00	8,00	47
43	5	2	1	4	1	2,60	5	3	3	3	1	4	3,17	1	1	1	1,00	1,00	1,00	1,00	1,00	9,00	41,00	12,00	62
44	3	3	3	4	3	3,20	3	2	2	3	1	3	2,33	2	2	3	2,33	0,87	0,87	1,00	0,91	33,00	14,00	19,00	66
45	3	2	2	5	2	2,80	4	1	1	2	3	4	2,50	1	2	4	2,33	0,87	1,00	1,00	0,96	13,00	26,00	14,00	53
46	1	3	2	3	3	2,40	3	3	3	3	3	3	3,00	2	4	3	3,00	0,87	1,00	1,00	0,96	18,00	10,00	11,40	39
47	4	4	3	2	4	3,40	3	3	3	4	4	3	3,33	3	5	5	4,33	1,00	1,00	1,00	1,00	28,00	27,00	6,00	61
48	2	2	3	2	5	2,80	4	4	4	3	4	5	4,00	5	5	5	5,00	1,00	1,00	1,00	1,00	25,00	10,00	10,00	45
49	4	4	4	4	5	4,20	4	4	4	5	4	4	4,17	3	3	5	3,67	1,00	1,00	1,00	1,00	41,00	18,00	11,00	70
50	2	2	4	2	5	3,00	4	4	1	1	4	1	2,50	4	3	1	2,67	0,87	1,00	1,00	0,96	13,00	13,00	6,00	32
51	2	3	3	3	4	3,00	3	4	4	4	3	4	3,67	3	2	3	2,67	1,00	1,00	1,00	1,00	35,00	16,00	12,00	63
52	3	4	1	2	2	2,40	4	4	4	3	4	3	3,67	4	4	4	4,00	0,87	1,00	1,00	0,96	35,00	15,00	12,00	62
53	1	2	1	2	1	1,40	4	3	4	4	4	3	3,67	1	1	1	1,00	0,87	1,00	1,00	0,96	45,00	2,00	8,00	55
54	2	3	2	3	4	2,80	4	4	4	4	4	5	4,17	4	4	4	4,00	0,87	0,87	1,00	0,91	14,00	21,00	9,00	44
55	4	4	4	4	4	4,00	3	3	4	2	2	4	3,00	2	2	2	2,00	1,00	1,00	1,00	1,00	15,00	20,00	7,00	42
56	3	3	1	2	2	2,20	4	3	3	4	3	3	3,33	3	2	2	2,33	0,87	1,00	1,00	0,96	15,00	16,00	12,00	43
57	1	2	3	2	3	2,20	4	4	3	1	3	4	3,17	3	3	3	3,00	1,00	1,00	1,00	1,00	12,00	43,00	9,00	64
58	3	2	3	2	3	2,60	3	3	3	3	3	4	3,17	4	2	4	3,33	0,87	1,00	1,00	0,96	16,00	13,00	10,00	39
59	3	4	2	3	3	3,00	3	4	4	3	2	4	3,33	4	3	5	4,00	0,87	0,00	1,00	0,62	9,00	30,00	11,40	50
60	1	3	3	3	2	2,40	1	3	5	3	4	4	3,33	4	4	2	3,33	0,87	0,87	1,00	0,91	54,00	3,00	8,00	65

Fuente: https://unsplash.com/collections/261936/technology

Resumen

La computación en la nube ha traído consigo muchas ventajas derivadas de sus características particulares (auto-servicio bajo demanda, acceso amplio a la red, elasticidad, modelo de multitenencia, pago por uso, entre otros) proporcionando a sus usuarios varios beneficios pero también nuevos retos en el aprovisionamiento de servicios de hardware y software. Entre los desafíos más significativos está el aprovisionamiento adecuado y de alta calidad de los servicios que el proveedor ofrece a sus clientes. Dado el amplio número de proveedores de plataformas en la nube, se hace indispensable que éstos ofrezcan servicios de calidad, a fin de satisfacer las expectativas de sus clientes. Los métodos y herramientas de monitorización juegan un papel crucial en este contexto ya que proporcionan información sobre la utilización de los servicios y su nivel de calidad.

La hipótesis de esta tesis es que la utilización de modelos en tiempo de ejecución (models@run.time), una técnica que se enmarca en la Ingeniería Dirigida por Modelos, puede constituir una solución apropiada ya que estos modelos permitirán cambiar dinámicamente los requisitos de monitorización sin la necesidad de cambiar la infraestructura de monitorización. Por tanto, el principal objetivo de esta tesis doctoral es la definición y validación empírica de un método de monitorización de servicios cloud (Cloud MoS@RT) que explote los modelos en tiempo de ejecución para hacer frente a los desafíos de monitorización de servicios cloud previamente mencionados.