Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Configurador de la monitorización de calidad de servicios en Google App Engine

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro López Peris

Tutor: Silvia Mara Abrahao Gonzales

2015 - 2016

Agradecimientos

En primer lugar, a mi tutora del trabajo, Silvia Mara Abrahao Gonzales quiero agradecerle el apoyo y la dedicación que nos ha dado en los últimos meses, además de guiarnos en la elaboración del trabajo de fin de grado.

A mi familia tengo que darles las gracias por el apoyo que me han dado. Siempre que los he necesitado he podido contar con ellos.

Y por último a mis compañeros, y en especial a Andrés, por toda la ayuda recibida a lo largo de mi formación académica.

Resumen

Hasta ahora la nube se ha visto como un modelo de servicio para el ahorro de costes cuando en realidad lo que está permitiendo a las empresas es contar con una plataforma que impulsa la innovación al permitir experimentar sin riesgos con nuevos modelos de negocio y trabajar en un formato de desarrollo continuo de mejora y actualizaciones de productos y servicios en tiempo real. Y es precisamente esa necesidad de innovar y mejorar lo que hará aumentar la demanda de soluciones basadas en nube. Actualmente existe la necesidad de mecanismos de monitorización flexibles que permitan tanto al cliente como al proveedor evaluar la calidad de los servicios en la nube con el fin de proporcionar una adecuada provisión de los mismos. A la hora de contratar estos servicios, se crea un Acuerdo de Nivel de Servicios (*Service Level Agreenment*, SLA) el cual define las características de calidad mínimas que el proveedor del servicio tiene que ofrecer en sus servicios, las cuales tienen que cumplirse por obligación. El incumplimiento de los acuerdos en el SLA desembocaría en una compensación por parte del proveedor del servicio al consumidor.

El grupo de investigación ISSI de la UPV está trabajando en una infraestructura de monitorización de servicios cloud que utiliza modelos en tiempo de ejecución (models@run.time) para evaluar la calidad de los servicios y detectar incumplimientos en los acuerdos a nivel de servicio (SLA). En trabajos anteriores se ha propuesto una infraestructura de monitorización [5] [6] y las herramientas software que soportan dicha infraestructura [4] [10]. En particular, se ha propuesto un configurador y un middleware de monitorización que ha sido implementado en la plataforma Microsoft Azure [10].En este trabajo se propone un componente configurador de la monitorización encargado de proporcionar a un monitor la información necesaria para realizar la tarea de monitorizar servicios en la plataforma Google App Engine. El objetivo de este proyecto es el análisis, diseño e implementación sistema de monitorización de calidad de servicios específico para la plataforma Google App Engine. En particular, este proyecto se centra en definir un configurador que permita la monitorización de servicios en esta plataforma y generar un modelo de calidad en tiempo de ejecución que contiene las directivas de monitorización.

Este enfoque significará un avance respecto a las herramientas de monitorización actuales, ya que ayuda a la configuración de estas con el fin de obtener datos de bajo nivel de la plataforma Google App Engine.

Palabras clave: Computación en la nube, Calidad de Servicio, Acuerdo de nivel de servicio, Modelos en tiempo de ejecución, Monitorización, Google App Engine.

Abstract

Until now the cloud has been seen as a service model for cost savings when in fact what is allowing companies is to have a platform that encourages innovation by allowing experiment without risk with new business models and work in a format continuous improvement and development of product updates and real-time services. And it is precisely this need to innovate and improve what will increase demand for cloud-based solutions. There is now the need for flexible monitoring mechanisms to enable both the customer and the supplier to assess the quality of services in the cloud with the proper provision of the same order. When hiring these services, a Service Level Agreement (SLA) it's designed, which defines the minimum quality characteristics that the service supplier has to offer in the services provided, which are mandatory to comply. The non-fulfillment of the SLA agreements will result in compensation to the consumer by the supplier of the service.

The ISSI research group from the UPV is working in a monitoring infrastructure for cloud services which exploits models at runtime (models@runtime) to both evaluate the service quality and detect violations of SLAs. In previous works, it has been proposed a monitoring infrastructure [5] [6] and a Middleware component in charge of this task [4] [10]. In particular, it was proposed the design of a platform-independent middleware and an implementation of this middleware for the Microsoft Azure platform [4].This paper presents a configurator component monitoring manager monitor provide the information necessary to perform the task of monitoring the Cloud platform is proposed. The objective of this project is the analysis, design and implementation monitoring system specific service quality for Google App Engine platform. In particular, this project focuses on defining a configurator that allows monitoring services on this platform and create a quality model at runtime containing monitoring policies.

This approach will mean a great advance over current monitoring tools, as it helps these settings in order to obtain low-level data platforms.

Keywords : Cloud computing, Services quality, Service level agreement, Models at Runtime, Monitoring, Google App Engine.

Resum

Fins ara el núvol s'ha vist com un model de servei per a l'estalvi de costos quan en realitat el que està permetent a les empreses és comptar amb una plataforma que impulsa la innovació en permetre experimentar sense riscos amb nous models de negoci i treballar en un format de desenvolupament continu de millora i actualitzacions de productes i serveis en temps real. I és precisament aquesta necessitat d'innovar i millorar el que farà augmentar la demanda de solucions basades en núvol. Actualment hi ha la necessitat de mecanismes de monitorització flexibles que permetin tant al client com al proveïdor avaluar la qualitat dels serveis en el núvol per tal adequada provisió dels mateixos. A l'hora de contractar aquests serveis, es crea un Acord de Nivell de Serveis (Service Level Agreenment, SLA) el qual defineix les característiques de qualitat mínimes que el proveïdor del servei ha d'oferir en els seus serveis, les quals s'han de complir per obligació . L'incompliment dels acords en el SLA desembocaria en una compensació per part del proveïdor del servei al consumidor.

El grup d'investigació ISSI de la UPV esta treballant en una infraestructura de monitorització de serveis Cloud que utilitza models en temps de execució (model@run.time) per avaluar la qualitat dels serveis i detectar incompliments en els acords de nivell de servei (SLA). En treballs anteriors se ha proposat una infraestructura de monitorització [5] [6] y un component Middleware encarregat de aquesta tasca. En particular, se ha proposat el disseny d'un middleware independent de plataforma i una implementació de aquest middleware per a la plataforma Google App Engine.En aquest treball es proposa un component configurador del monitoratge encarregat proporcionar al monitor la informació necessària per realitzar la tasca de monitoritzar la plataforma Cloud. L'objectiu d'aquest projecte és l'anàlisi, disseny i implementació sistema de monitorització de qualitat de serveis específic per a la plataforma Google App Engine. En particular, aquest projecte se centra en definir un configurador que permeti la monitorització de serveis en aquesta plataforma i generar un model de qualitat en temps d'execució que conté les directives de monitorització.

Aquest enfocament significarà un gran avanç respecte a les eines de monitorització actuals, ja que ajuda a la configuració d'aquestes per tal d'obtenir dades de baix nivell de les plataformes.

Paraules claus : Computació al núvol, Serveis de qualitat, Acord de nivell de servei, Models en temps de execució, Google App Engine.

Tabla de contenidos

Tabla de Ilustraciones

1. Introducción

Este documento tiene a fin presentar el trabajo de desarrollo de un configurador de la monitorización de calidad de servicios desplegados en Google App Engine. A lo largo de este documento se expondrá los conceptos más relevantes para el proyecto, el desarrollo y resultado final del trabajo.

1.1. Motivación

La corriente del desarrollo de software en los últimos años, se ha encaminado a la creación de aplicaciones que tengan unas características esenciales: alta disponibilidad y accesibilidad desde cualquier lugar. Esto ha sido facilitado en gran medida por la globalización en el uso de Internet. Hoy en día, es muy difícil encontrar un hogar que no tenga un ordenador o una persona que no posea un dispositivo móvil con acceso a Internet.

La mayoría de usuarios de dispositivos informáticos, se decantan por el uso de aplicaciones web, antes que por la descarga de aplicaciones en sus dispositivos, lo cual puede ocasionar daños o cargas innecesarias en memoria. Debido a esto, la mayoría de desarrollos de aplicaciones tienden a ser aplicaciones o servicios web. Estos son aplicaciones que están desplegadas en la web, cuya lógica esta puesta en servidores que dan un servicio de despliegue y ejecución en la web.

La necesidad de una alta disponibilidad y accesibilidad, ha creado una nueva tendencia en el mercado: la creación de servicios en la nube. Se trata de una tecnología relativamente nueva que está cobrando una gran importancia en el desarrollo y uso de sistemas software, cuyo principal interesado son las empresas de desarrollo de productos software, junto con las encargadas de seguridad software y empresas desarrolladoras de aplicaciones para dispositivos móviles.

La importancia de la nube ha ido adquiriendo fuerza y relevancia a medida que las personas han tomado conciencia de la necesidad de guardar datos personales a buen recaudo, compartir contenido digital de forma rápida e instantánea, y al tiempo que los soportes “físicos” han sido menos accesible o limitados en sus capacidades de manejo de archivos. Si bien las memoria USB siguen siendo un medio físico muy factible para trasladar grandes cantidades de datos de forma puntual y para un propósito determinado, el uso de la nube se ha convertido en un medio más factible para compartir archivos sin necesidad de contacto físico entre las personas: aunque parezca una contradicción, la nube es una solución de almacenamiento en Red muy social, tanto que incluso proveedores como Dropbox se han integrado totalmente en Facebook, la red social por excelencia.

Por ello, muchas empresas han adoptado el modelo de negocio de la computación en la nube. Estas plataformas agrupan distintos tipos de servicios: Infraestructuras como servicios *(Infraestructure as a Service*, IaaS), Plataforma como Servicio (*Platform as a Service*, PaaS), o Software como Servicio (*Software as a Service*, SaaS). Este modelo de negocio posee un gran número de ventajas tanto para los clientes como para los proveedores, tales como la alta disponibilidad, estabilidad, modelo de pago por uso, máximo aprovechamiento de

recursos hardware, capacidad de escalar-reducir las aplicaciones bajo demanda y rápidamente, etc..

En este trabajo nos centraremos en servicios que utilizan Software como Servicio, SaaS. SaaS es un modelo de distribución de software donde el soporte lógico y los datos almacenados se alojan en servidores de una compañía TIC a los que se accede vía internet. La empresa TIC se ocupa del mantenimiento, operación diaria y del soporte usado por el cliente.

Un servicio SaaS, tiene un número de usuarios en ocasiones con picos de tráfico muy altos, por tanto, debe soportar el acceso de millones de usuarios con un rendimiento escalable que responda a esta variabilidad. Para determinar el nivel de cumplimiento de un servicio, se crea un Acuerdo de Nivel de Servicio (*Service Level Agreement*, SLA) entre las dos partes. Un SLA es un contrato entre el proveedor del servicio y el usuario de este. En este se acuerdan los requisitos mínimos que el servicio debe cumplir en la ejecución del servicio ofertado al usuario. Normalmente el proveedor de servicios cloud ofrece algunas garantías de rendimiento para servicios de cómputo y deja la detección de violaciones del SLA en manos del cliente. Además, pocos proveedores cloud recompensan al cliente automáticamente por estas violaciones, por lo que es necesario que el cliente pueda demostrar, con evidencias, estas violaciones encontradas.

De esta manera, es importante en la actualidad que el cliente cuente con herramientas que le permitan conocer el estado del comportamiento en tiempo real del servicio ofertado y comprobar que, en efecto, este servicio cumple los requisitos mínimos acordados en el SLA, para tener evidencias de sus violaciones, o la garantía de su correcto comportamiento. Igualmente, es interesante para la empresa proveedora ya que puede dar garantías de uso a sus clientes.

Para dar respuesta a esta necesidad varios métodos y herramientas de monitorización se han propuesto recientemente. Sin embargo, estas propuestas tienes varias limitaciones (ver sección 2).Por todo ello es necesario el uso de una herramienta de monitorización, que permita monitorizar el cumplimiento de las clausulas del SLA, medir los atributos de calidad de interés y que genere informes fiables del nivel de servicio que se provee.

El desarrollo de estas herramientas supone un reto dado que el despliegue y ejecución de sistemas software en infraestructuras altamente dinámicas introduce un nuevo conjunto de desafíos y requisitos en lo que respecto a la monitorización. En consecuencia, una aplicación de monitorización debe de poseer flexibilidad para adaptarse a cualquier cambio en los requisitos de monitorización y mantenerse según el servicio escale hacia arriba o hacia abajo dinámicamente.

Una solución que podría satisfacer estas necesidades es el desarrollo de software dirigido por modelos (Model Driven Engineering). Sin embargo, sería una solución estática para la integración del SLA con la monitorización de métricas, lo cual imposibilitaría el cambio en esta. Los requisitos acordados en el SLA pueden sufrir cambios a lo largo del tiempo por nuevos acuerdos entre las partes. Por ello, debemos considerar opciones más dinámicas. Para solucionar este problema, y como solución más viable, Cedillo *et al.* [5][6] plantea el uso de los modelos en

tiempo de ejecución (models@runtime). Estos modelos tienen el potencial de ser utilizados en tiempo de ejecución para supervisar y verificar aspectos particulares de los servicios cloud en tiempo de ejecución.

Esta tecnología especifica los datos que debe recoger la herramienta de monitorización, los cuales estarán basados en el SLA y en otros requisitos adicionales. Este modelo estará producido acorde a un modelo de calidad SaaS, que recogerá las características, subcaracterísticas, atributos y métricas que permitirán medir los atributos de calidad de los servicios en la nube.

Un modelo de calidad SaaS describe las características de un producto software, como se relacionan, como pueden ser medidas y que interpretación se puede dar a estas medidas.

Utilizando el modelo de calidad SaaS junto al SLA y otros requisitos adicionales, podremos crear un modelo con las características para la medición de los acuerdos y una estructura para su medición, el cual estará reflejado en el modelo en tiempo de ejecución (models@runtime).

Esta aproximación de monitorización de servicios cloud basada en modelos en tiempo de ejecución así como la infraestructura software que le soporta se ha propuesto en trabajos anteriores [4][5][6]. Aunque se ha propuesto un configurador de monitorización de servicios cloud [10] (Jimenez, 2014), este ha sido explotado para su uso únicamente en la plataforma Microsoft Azure.

1.2. Objetivos

El propósito de este trabajo es crear un configurador para la monitorización de la calidad de servicios cloud (SaaS) y detección de incumplimientos del SLA que soporte la infraestructura de monitorización propuesta en [5][6], centrándonos en el desarrollo de la aplicación web para la configuración de la monitorización de servicios desplegados en la plataforma Google App Engine.

Se han tratado de conseguir los siguientes objetivos:

- La construcción de una herramienta web que apoye la creación de un modelo en tiempo de ejecución que recoja los datos del SLA necesarios para establecer los requisitos no funcionales a monitorizar y clasificar estos requisitos utilizando un modelo de calidad SaaS e indicar en base a él los métodos de medición de esos requisitos.
- Instanciación de esta herramienta en la plataforma cloud Google App Engine, investigando las herramientas que esta aporta para la extracción de los datos propios de la plataforma y trabajando con estas.
- Definición de métodos para el cálculo de las métricas .

1.3. Contexto

Este proyecto es la creación del configurador de la monitorización de aplicaciones o servicios desplegados en la plataforma cloud Google App Engine

basado en el trabajo de fin de grado presentado por Javier Jiménez el cual define un configurador y un middleware de monitorización de servicios cloud para la plataforma Microsoft Azure [10][4].

El proyecto en general está dividido en dos trabajos de fin de grado: el trabajo de Andrés Páez Rosales, titulado "Configurador de la monitorización de calidad de servicios en Google App Engine" cuyo objetivo es crear un middleware para la monitorización de la calidad de servicios cloud (SaaS) que soporte la infraestructura de monitorización propuesta por Cedillo et al [5] [6], y el presente trabajo que proporcionará una herramienta para la configuración de la monitorización y generación del modelo en tiempo de ejecución que será la entrada para el middleware de monitorización. Todo esto para monitorizar la aplicaciones desplegados en la plataforma Google Appp Engine así como la detección de incumplimientos del SLA.

Asimismo, este proyecto forma parte del proyecto de investigación "Desarrollo Incremental de Servicios Cloud Dirigido por Modelos y Orientado al Valor del Cliente (Value@Cloud)" realizado por el grupo de investigación de Ingeniería del Software y Sistemas de Información (ISSI) del Departamento de Sistemas Informáticos y Computación (DSIC) y del Departamento de Organización de Empresas (DOE) de la Universitat Politécnica de Valencia. En particular, el trabajo desarrollado forma parte del paquete de trabajo "Definición y gestión de mecanismos de monitorización del valor en tiempo de ejecución".

1.4. Estructura del documento

En el primer capítulo se ha hecho una pequeña introducción a este trabajo, declarando la motivación para hacerlo, los objetivos que espera cumplir, el contexto alrededor del trabajo y la estructura que va a tener este documento.

Los siguientes capítulos se estructuraran de la siguiente manera:

En el Capítulo 2 se presenta el estado del arte acerca de los métodos relacionados con este trabajo: la evolución de las técnicas de monitorización y la monitorización de servicios en la nube.

En el Capítulo 3 se presenta el contexto a este trabajo en lo que refiere a Cloud Computing. Se explica los conceptos principales de la computación en la nube, los servicios que ofrece y se presenta los acuerdos de nivel de servicio que definen las relaciones legales entre el cliente y el consumidor del servicio.

En el Capítulo 4 se presenta el proceso de monitorización propuesto por Cedillo [5][6] exponiendo sus distintas fases y artefactos.

En el Capítulo 5 se presenta la estructura del Configurador de la Monitorización presentando sus componentes, funciones y la relación entre ellos, así como la contribución de este proyecto que consiste en la creación de un configurador del monitor propuesto en la plataforma Google App Engine. En particular, se presenta el diseño de la arquitectura del configurador en la plataforma Google App Engine y su implementación.

En el Capítulo 6 presenta un ejemplo de configuración de la monitorización de servicios en la plataforma Google App Engine haciendo uso del configurador propuesto.

Para finalizar, en el Capítulo 7 se describe las conclusiones generales, los problemas enfrentados en el desarrollo y los trabajos futuros.

2. Análisis del estado del arte

En esta sección se expondrán las tecnologías de monitorización halladas en la literatura. Primero se procederá a introducir la monitorización de servicios en la nube. A continuación, se pondrá en contexto la monitorización de servicios en la nube a través de sus orígenes en la monitorización de sistemas distribuidos. Tras ello se procederá a exponer los distintos acercamientos actuales encontrados en la literatura al problema, describiendo sus diferencias y tendencias. Por último se procederá a hacer una comparación entre las distintas herramientas de monitorización y se detallaran sus virtudes y carencias.

2.1. Introducción a la monitorización en la nube

Actualmente muchas empresas están adoptando tecnologías cloud como solución de provisión de recursos tecnológicos, para sus necesidades de infraestructura y software. Como consecuencia de esto, se hace necesario contar con mecanismos de monitorización flexibles, que permitan tanto al cliente como al proveedor, evaluar la calidad de los servicios ofertados con el fin de ofrecer una adecuada provisión de los mismos. Existen muchas soluciones en el mercado para la monitorización de servicios desplegados en la nube. Sin embargo, la mayoría provee métricas simples, que no están directamente relacionadas a los Acuerdos de Nivel de Servicios (SLA) y tampoco cuentan con mecanismos personalizados, que permitan especificar nuevas fórmulas para el cálculo de métricas complejas. Para administrar estos servicios y hacer posible que cumplan con los términos de acuerdo de servicio (SLA) es vital obtener información de rendimiento y calidad de los servicios desplegados.

Un proceso que complete y precisamente identifica la causa base de un evento mediante la captura de la información correcta en el momento correcto y con el menor coste para determinar el estado de un sistema y supervisar su estatus en una forma oportuna y significante.

Estas herramientas son vitales para proveedores y clientes de servicios en la nube ya que los métodos de pago por uso necesitan de datos precisos de rendimiento para poder realizar la facturación correcta. La monitorización es una parte necesaria para los procesos de provisión de recursos/servicios, planificación óptima de la calidad, comprobación de SLAs, gestión de la configuración, facturación y seguridad y privacidad. Para este cometido una gran variedad de herramientas y técnicas están disponibles. Estas utilizan distintos tipos de acercamiento a la monitorización. Estas herramientas pueden clasificarse según el tipo de datos que extraen y de cómo los obtienen.

2.1.1. Tipo de datos

Podemos clasificar los datos de un servicio cloud en tres capas, relacionadas con las tres formas de provisión de servicios cloud: IaaS, PaaS y SaaS. La infraestructura como servicio (***Infrastructure as a Service*, IaaS**) se encuentra

en la capa inferior y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran para manejar tipos específicos de cargas de trabajo. El ejemplo comercial mejor conocido es Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento esenciales (respectivamente).

Es la capa del medio, en inglés conocida como **Platform as a Service (PaaS)** consiste en la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.).

De esta forma, un arquetipo de plataforma como servicio podría consistir en un entorno conteniendo una pila básica de sistemas, componentes o APIs preconfiguradas y listas para integrarse sobre una tecnología concreta de desarrollo

En este nivel, los datos obtenidos por la herramienta de monitorización serán los referentes al entorno de máquinas virtuales sobre los que se ejecuta el servicio (sistema operativo y otras herramientas). Este es el caso de *Google Stackdriver*, la herramienta de monitorización propia de Google App Engine que permite obtener datos del sistema operativo Linux sobre el que trabajan las herramientas como el tiempo de ejecución de una petición.

Los datos de SaaS por otra parte serán los referentes a la implementación concreta del servicio y variarán dependiendo del dominio del servicio y la tecnología con la que esté implementado. Por ejemplo un servicio que gestiona una página de comercio electrónico o un servicio de gestión de pagos aportarán datos diferentes, en uno podremos obtener datos acerca del tiempo medio que un ítem pasa en el carrito hasta que es comprado y en otro el número de pagos realizados con una determinada sucursal en un día.

2.2. Monitorización en sistemas distribuidos tradicionales

El germen común de los inicios de monitorización de servicios en la nube, se tienen también en las tecnologías que marcaron los inicios de la nube, esto son los sistemas distribuidos de clústeres de cómputo y grids. De estas tecnologías emergieron también las herramientas de monitorización que se aplicarían a la nube.

La computación en la nube se diferencia de la de los clústeres en que el sistema es considerado como una unidad y la tarea a desarrollar es la misma en cada nodo de los clústeres. Además los clústeres son un grupo de ordenadores conectados entres si por una red local (LAN). Esto no es así en la computación en la nube donde en cada nodo diferentes máquinas virtuales comparten recursos cada una pudiendo servir a un propósito diferente y la escalabilidad es más amplia pudiendo estar geográficamente distribuidos. La comparación con el grid es más compleja dado que comparten varios puntos en común y de hecho la infraestructura de la computación en la nube puede verse como un grid. Según Hassan [9] la diferencia radica en los servicios ofrecidos por ambas plataformas,

mientras que los grids están diseñados para ser de propósito general y las interfaces se mantienen a un nivel más bajo, la nube ofrece un conjunto más limitado de características pero a un nivel más alto ofreciendo servicios y recursos más abstractos (como por ejemplo un sistema operativo).

Aunque las características de este tipo de sistemas no sean idénticas a las condiciones de la nube, tanto como por requisitos a ser monitorizados como por las características de la tecnología que los constituyen, algunos de los sistemas y técnicas de monitorización han sido adaptadas para utilizarse en la nube y han surgido otras específicas para este tipo de entornos. Por ejemplo la herramienta de monitorización Nimsoft originalmente era utilizada para extraer datos relacionados con la infraestructura (red, servidores, bases de datos…) pero después evolucionó para ofrecer sus servicios como un Monitor de Nube Unificado para monitorizar servicios alojados externamente.

Categoria	Grid	Cluster	Cloud
Requerimientos SLA	Alto	Muy Alto	Bajo
Tipo de trabajo	Repetitivo	Muy repetitivo - MPP	Not MPP-Flujo de trabajo repetitivo
Dependencia de datos	Media	Alta	Alta
Dependencia I/O	Media	Alta	Baja
Dependencias externas	Media	Ninguna	Ninguna
Tiempo de integracion	Largo	Largo y en marcha	Corto
Tolerancia a fallos	No deseable pero posible	No	No
Unidades tipicas de trabajo	Segundos a minutos	Muy corto. Algunos minutos max.	Largo. Desde minutos a horas

Tabla 1. Comparación de sistemas

MPP*: Massively parallel

2.3. Herramientas de monitorización en la nube

Actualmente existen numerosas herramientas académicas y comerciales que cumplen con las funciones de monitorización de servicios. Algunas ofrecen solamente medios de extracción de datos del servicio monitorizado mientras que

otras ofrecen soluciones integrales que se ocupan desde la extracción de datos hasta la generación de informes y su visualización.

Según la revisión realizada en 2014 por Kaniz Fatema et al [11], se pueden clasificar estas herramientas según el alcance del tipo de datos que pueden obtener del servicio a ser monitorizado, tal y como se ha comentado la sección 2.1 de este capítulo. Muchas de las herramientas de monitorización de servicios en la nube como Amazon Cloud Watch [20], Azure Watch, Nimsoft y Datadog son capaces de monitorizar tanto al nivel de la infraestructura como de la aplicación. Sin embargo, otras herramientas solo son capaces de monitorizar o bien al nivel de la infraestructura, por ejemplo BMC TrueSight Pulse.

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, siendo estas cada día más necesarias. Un ejemplo de herramienta multiplataforma es ZENOSS .

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. Es el caso de Amazon Cloud Watch el monitor de los Amazon Web Services en Amazon EC2, incluyendo instancias de servicios y las diferentes bases de datos ofertadas por Amazon[17]; Azure Diagnostiscs [21] en el caso de Microsoft Azure y sus servicios, instancias de servicios de Windows Azure, bases de datos SQL de Azure y almacenamiento Azure; o Google Cloud Monitoring la herramienta de Google Cloud Platform que permite la monitorización de servicios en la nube, máquinas virtuales y otros servidores de código abierto como Apache, MongoDB, Nginx...

En las siguientes subsecciones se llevará a cabo una revisión de las herramientas, técnicas y tendencias más populares, concluyendo con una comparación de las herramientas más populares y una exposición de los puntos débiles de estas técnicas.

2.3.1. Monitorización basada en agentes

Con respecto a las técnicas utilizadas por estas herramientas para la recuperación de información, la más común presente en propuestas académicas es Nagios y PandoraFMS. La monitorización por agentes es típica de la monitorización multipropósito. Este tipo de monitorización consiste en instalar una pieza de software denominada agente en cada servicio a ser monitorizado. Cada agente se encarga de recuperar datos del servicio en el que se ejecutan y, dependiendo de la solución, procesarlos construyendo métricas de más alto nivel. Cada uno de los agentes se encarga de que periódicamente esos datos propios de cada servicio sean enviados a un segundo componente software llamado servidor de monitorización cuya tarea principal es recuperar los datos recolectados por los agentes, tratarlos si fuese necesario y almacenarlos para su análisis posterior. Este servidor suele encargarse también del ciclo de vida de los agentes, detectando agentes desconectados y regulando el comportamiento de estos según el estado en el que se encuentre el sistema (por ejemplo reduciendo la actividad de los agentes en momentos de elevada carga) y además de lanzar alertas si fuese necesario.

Con el propósito de mejorar el rendimiento y características de calidad como la disponibilidad, se han establecido diferentes propuestas que varían la topología de la monitorización. Por ejemplo, para evitar que la monitorización se vea comprometida por un fallo del servidor del nodo que actúa como servidor de monitorización se han propuesto sistemas de comunicación en los que el rol de servidor de la monitorización puede ser adoptado por cualquiera de los agentes llegado el caso. También con el propósito de mejorar la eficiencia de la monitorización Meng et al. [13] proponen el uso de árboles de agentes que puedan desarrollarse o ser podados de forma dinámica según el servicio varía en su aprovisionamiento según las condiciones de elasticidad así lo requieran.

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. [14] ofreciendo distintas soluciones a nivel de arquitectura y de funciones de cálculo sobre estos datos.

2.3.2. Monitorización como Servicio (Monitoring as a Service, MaaS)

Una tendencia que se hace patente en las últimas publicaciones sobre la materia es ofrecerlos servicios de monitorización como un servicio en la nube. La mayoría de herramientas de monitorización ofrecen sus servicios como SaaS, es decir el servicio es alojado y mantenido en una plataforma externa a la del servicio que se quiere monitorizar y un tercero contrata esos servicios de monitorización.

La provisión de un servicio de monitorización de esta manera supone grandes ventajas. Meng et al. [13] son expuestas algunas:

Primero, este tipo de provisión de servicios de monitorización ayuda a minimizar el coste de propiedad de estas herramienta permitiendo obtener los mismos resultados de forma más barata ya que el consumidor de estos servicios ya no tendrá que hacerse cargo de los costes de sobre trabajo que generan este tipo de aplicaciones a la hora de monitorizar los servicios, evitando así costes extras de almacenamiento y procesamiento, y la sobrecarga en instantes de alta demanda.

Además este modelo permite beneficiarse de las actualizaciones centralizadas de la plataforma de monitorización, proveyendo software de mayor calidad de manera constante.

También este sistema hace más sencillo para los propietarios de los servicios desplegar monitores a distintos niveles de los servicios en la nube en comparación con la obligación de crear cada uno de estos monitores específicos para tu servicio.

Además de ello, MaaS se beneficia de las ventajas del modelo de negocio de SaaS, de manera que permite el modelo de facturación pago por uso (pay-per-use). De esta forma, los costes iniciales necesarios para el uso de estas herramientas sería más bajo que en el formato tradicional, ajustándose así a proyectos de menor envergadura con presupuestos reducidos que podrán beneficiarse de este tipo de herramientas.

Este modelo también permite a los proveedores de servicio consolidar las demandas de monitorización a distintos niveles para conseguir una monitorización eficiente y escalable. Teniendo la capacidad de ceder más recursos según sea necesario para la monitorización de servicios que hayan crecido debido a la alta demanda y así evitando que los servicios de monitorización acaparen más tiempo de procesador del necesario en momentos críticos de la provisión del servicio.

Por último, MaaS incita a los proveedores a invertir en nuevas herramientas de monitorización y ofrecer siempre un servicio de monitorización de la más alta calidad.

Sin embargo, las ventajas ofrecidas por MaaS tienen su contrapartida en la dificultad de implementación. El uso de servicios MaaS es problemático cuando el acercamiento que se usa para la extracción de datos de la monitorización es el de agentes. Esta técnica hace necesaria que los servicios en la nube a ser monitorizados sean capaces de soportar la instalación y ejecución de los agentes de los servicios SaaS encargados de extraer y transmitir los datos del servicio al núcleo de la monitorización. Por ello, estos agentes suelen estar disponibles en código abierto en el lado del cliente para facilitar la incorporación a nuevos servicios.

Estos servicios además suelen proveer funciones extra como un sistema de alertas que avisa a los clientes de incidencias con el servicio por medio de mensajes SMS, correos electrónicos y otros medios de comunicación. Así como herramientas de visualización del estado de los servicios monitorizados.

2.3.3. Monitorización del SLA

Uno de los objetivos de la monitorización de servicios en la nube consiste en vigilar el cumplimiento del Acuerdo de Nivel de Servicio (Software Level Agreement, SLA). Un SLA es un contrato estricto entre un proveedor de un servicio y su cliente en el cual se fijan los términos referentes a la calidad de la provisión del servicio. En él, se establecen y consensan los términos de nivel de calidad de servicio de todos los requisitos no funcionales (RNFs), por ejemplo, garantizar que la latencia del servidor de aplicaciones web sea menor que 100 ms. Las principales partes involucradas en este acuerdo son las partes signatarias: cliente y proveedor del servicio. Además, puede haber terceras partes involucradas que cumplen el rol de auditores encargados de comprobar que los términos del servicio de calidad sean cumplidos. Para que sea posible comprobar el cumplimiento del SLA es necesario contar con las herramientas y procedimientos apropiados que permitan medir los atributos correspondientes a los RNFs expresados en el documento. Para ello es necesario un monitor que sea capaz de extraer datos en el nivel de aplicación ya que los términos del SLA habitualmente asumen que serán garantizados a este nivel. Sin embargo este es un trabajo complicado para los monitores convencionales, ya que los datos extraídos de la capa de plataforma o la

de infraestructura no pueden asociarse de manera obvia con las métricas requeridas en la capa de aplicación para la medición de los RNFs expresados en el SLA [16]. Como se ha comentado en el punto anterior, una aplicación puede compartir una o varias máquinas virtuales así como ejecutarse en varias máquinas físicas a la vez, hecho que dificulta la asociación de la semántica de los datos de bajo nivel con respecto a las características de más alto nivel que quieren ser monitorizadas.

Debido a ello, las técnicas de monitorización del SLA, más que necesitar de un acercamiento a la extracción de datos específico, se centran en aspectos de más alto nivel. Su objetivo es conseguir establecer las relaciones pertinentes entre las especificaciones del SLA (muchas veces escritas en lenguaje natural) y el tipo de datos a obtener del servicio en la nube que permitan medir esos RNFs. De forma que el posterior análisis de los datos obtenidos permita determinar el cumplimiento o no de los requisitos establecidos en el SLA.

2.3.4. Comparación de herramientas de monitorización

Para la comparación de las herramientas de monitorización se han tomado un conjunto de soluciones comerciales y académicas populares, incluyendo las propias de los grandes proveedores de servicios en la nube. Se ha comparado la capacidad de ser utilizada en distintas plataformas, el tipo de tecnología que utilizan, el tipo de datos que son capaces de extraer de los servicios, si ofrecen monitorización y detección de violaciones del SLA y si son capaces de incorporar a su proceso de monitorización métricas personalizadas. La Tabla 1 presenta un resumen de la comparativa de estas herramientas.

	Multiplataforma	Datos de infraestructura	Datos de plataforma	Datos de aplicación	Basada en agentes	MaaS	Monitorización del SLA	Métricas personalizables	Tipo de solución
Azure Watch /Diagnostics	**No**	**Si**	**Si**	**Si**	*	**No**	**No**	**Si**	**Comercial**
Amazon Cloud Watch	**Si**	**Si**	**Si**	**Si**	**Si**	**No**	**No**	**Si**	**Comercial**
Google app engine/ Stackdriver	**Si**	**Si**	**Si**	**Si**	**Si**	**Si**	**No**	**Si**	**Comercial**
Nagios	**Si**	**Si**	**Si**	**Si**	**Si**	**No**	**No**	**Si**	**Comercial, GPL v2**
ZMON	**Si**	**Si**	**Si**	**Si**	**Si**	**No**	**No**	**Si**	**Comercial**

Avaya	**Si**	**Si**	**Si**	**Si**	**Si**	**No**	**Si**	**Si**	**Comercial**

Tabla 2 - Comparación de herramientas de monitorización

2.4. Conclusiones

Las herramientas analizadas y la revisión de la literatura al respecto de la monitorización de servicios en la nube destacan el esfuerzo realizado por encontrar métodos de la 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, pero en casos que esto no sea posible existen herramientas que utilizan otras técnicas. 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.

Así mismo, la monitorización de los SLAs no aparece en la mayoría de aplicaciones comerciales, el usuario debe interpretar los datos de monitorización con el fin de corroborar el cumplimiento o no de los acuerdos de nivel de servicio. A excepción de las herramientas propias de las plataformas de provisión de servicios en la nube, el resto de herramientas hacen especial hincapié en estar disponibles para diversas tecnologías

3. Cloud computing

En esta sección se ahondará en los conceptos relativos a la computación en la nube (Cloud computing). Primero daremos una definición general del concepto de Computación en la nube. Seguidamente expondremos las ventajas y desventajas de la utilización de servicios de computación en la nube. En tercer lugar, clasificaremos los servicios cloud que se ofrecen en la actualidad. Y para finalizar esta sección explicaremos el significado del Acuerdo de nivel de servicio (*Service Level Agreement* SLA) en el entorno de la computación en la nube.

3.1. Definición de computación en la nube

La computación en la nube es una nueva tecnología añadida a la jerga de las tecnologías de la información a comienzos de 2007. Esta se refiere tanto al servicio ofertado en internet como al hardware y sistemas software en los data centers que ofrecen el servicio [12]. La utilidad de la computación en la nube es la de dar acceso por demanda a recursos informáticos. En esta estructura, los clientes siguen un modelo de pay-as-you-go que provee acceso a cuantos todos los recursos informáticos que se necesiten desde cualquier lugar [2].

Es un modelo que permite el acceso mediante la red a distintos recursos informáticos configurables, los cuales pueden ser provistos y desplegados con un esfuerzo de gestión e interacción con el proveedor mínimo [1]. Este modelo cloud se compone de las siguientes características esenciales:

- **Autonomía:** El consumidor puede unilateralmente proveerse de las características computacionales, tales como tiempo del servidor o almacenamiento de red, sin necesidad de requerir interacción humana con cada proveedor de servicio.
- **Acceso mediante la red:** Los recursos están disponibles a través de la red y accesible mediante mecanismos estandarizados que promueven su uso mediante un amplio rango de tipo de plataformas (p. ej., dispositivos móviles, tablets, ordenadores portátiles, ordenadores de sobremesa y estaciones de trabajo).
- **Agrupación de recursos:** Los recursos provistos por el proveedor están agrupados para servir múltiples consumidores utilizando un modelo de simultaneidad de múltiples usuarios, con distintos recursos dinámicos, tanto físicos como virtuales, y reasignados acorde a la demanda del consumidor. Hay un sentido de localización independiente en la que el consumidor no tiene control o conocimiento sobre la localización exacta de los recursos ofrecidos por el proveedor, aunque si sobre una localización a un nivel más abstracto, como puede ser Ciudad o País.
- **Elasticidad:** Las características pueden ser provistas y publicadas con elasticidad, en muchos casos automáticamente, para ser escalado externamente e internamente dependiendo de su demanda. Para el consumidor, las características disponibles a menudo aparecen de forma ilimitada y puede ser adquirida en cualquier momento.
- **Servicios medibles:** Los sistemas en la nube controlan y optimizan automáticamente los recursos influenciando las características medidas a

un nivel de abstracción apropiado al tipo del servicio. El uso de recursos puede ser monitorizado, controlado y reportado, proveyendo transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

- **QoS predefinidos:** Los términos de calidad van expresados en los acuerdos de nivel de servicio (SLA) los cuáles expresan los parámetros de calidad entre los cuales el servicio va a ser ofrecido.
- **Modelo de computación bajo demanda:** Las organizaciones ya no tienen la necesidad de construir y mantener sus propios centros de datos. Sus necesidades están cubiertas por los recursos contratados a un proveedor que posee unos recursos mucho mayores que los que podría obtener la empresa a un precio menor.

3.2. Ventajas y riesgos del cloud computing

En esta sección se expondrán las ventajas y desventajas de la computación en la nube, las cuales entran en distintos ámbitos tales como económicos, como de rendimiento, como de seguridad y muchos más

3.2.1. Ventajas del cloud computing.

Las ventajas del cloud computing las podemos clasificar según el punto de vista del uso de esta, es decir, desde el punto de vista de los proveedores de servicios de cloud computing y desde el punto de vista de consumidores de este.

3.2.1.1. Punto de vista de los proveedores

- **Utilización de hardware más óptima**: En la mayoría de organizaciones, la utilización de recursos hardware raramente está a su capacidad máxima. El valor de la utilización de estos recursos es extremadamente minimizado en contra del coste de su obtención. La computación en la nube puede ayudar a las organizaciones con grandes inversiones en recursos hardware a alquilar los recursos no utilizados a otras organizaciones.
- **Ingresos mayores**: Da la oportunidad a especialidades que no se encontraban antes en el Mercado de ejecutar nuevos modelos de negocios para recibir mayores ingresos económicos. Además, la posibilidad de arrendar el hardware no usado da a la organización la posibilidad de producir beneficios extras.
- **Mercados Software más amplios:** Los proveedores software pueden entregar sus aplicaciones en forma de suscripciones básicas. Esta característica pueden fomentar el uso de su aplicación a los consumidores, lo cual también produce minimizar el uso de software pirata, permitiendo al usuario recibir ingresos mayores.
- **Monitorización de actividad:** Los proveedores son capaces de monitorizar las acciones y actividades de los clientes. Con esto, pueden promover los servicios y productos más utilizados por los clientes, con la posibilidad de ganar más dinero.
- **Mejor gestión de publicación:** Los proveedores SaaS tienen la libertad de enviar diferentes parches, releases y actualizaciones a cualquier cliente

por separado. Dado que las aplicaciones software están alojados en servidores, las actualizaciones pueden ser automáticamente aplicadas sin la intervención del cliente.

3.2.1.2. Punto de vista de los consumidores

- **Costes reducidos**: La computación en la nube permite a pequeñas y medianas empresas recibir servicios de startups low cost que arriendan servicios de los proveedores cloud, en vez de tener su servicio propio. Además, grandes empresas pueden aprovecharse de la computación en la nube como solución táctica para hacer frente a situaciones puntuales sin gastar grandes sumas de dinero en adquirir recursos que no van a estar inactivos la mayoría de tiempo.
- **Tiempo de configuración menor**: Las organizaciones pueden adquirir y operar los recursos necesarios sin necesidad de invertir mucho tiempo en la instalación y configuración de estos recursos.
- **Problemas de instalación/ actualización inexistentes**: Las organizaciones se evitan tener problemas de instalación o actualización de los recursos. Los proveedores de los recursos o servicios son los encargados de proporcionar y gestionar la instalación y actualización de los recursos.
- **Mayor escalabilidad**: Las organizaciones pueden instalar cualquier número de instancias hardware/software sin mayor esfuerzo. Además, los clientes pueden eliminar las instancias no utilizadas para ahorrar costes. Esta elasticidad tiene dos ventajas principalmente: Primero, libera a las organizaciones de gastar grandes costes en recursos informáticos que no se van a utilizar en un futuro. Segundo, permite a las organizaciones afrontar pequeños inconvenientes con flexibilidad añadiendo nuevos recursos en el momento que lo necesite.

3.2.2. Riesgos del cloud computing.

La computación en la nube todavía es una tecnología joven. Las organizaciones normalmente prefieren adoptar metodologías contrastadas a lo largo del tiempo y por la utilización de las mejores prácticas de anteriores consumidores. Algunos de los riesgos de la computación en la nube incluyen:

- **Estándares**: La computación en la nube carece de estándares necesarios para perder acoplamiento entre proveedores y clientes. Cada cliente deberá utilizar APIs ofrecidas por los proveedores, para permitir que sus aplicaciones puedan utilizar los servicios disponibles. Esto quiere decir que cada proveedor tiene su propia tecnología y estándares, lo cual hace imposible para los clientes moverse de un proveedor a otro.
- **Dependencias**: Las organizaciones no quieren invertir en soluciones informáticas que en un futuro decidan dejar el mercado.
- **Transparencia**: Debido a que los proveedores tienen el control total de los recursos de la nube, estos pueden realizar cambios en la infraestructura y en los servicios sin necesidad de notificar a sus clientes. Estas cuestiones deben quedar escritas en el SLA para garantizar la continuidad y fiabilidad de las soluciones que el cliente utiliza.

- **Seguridad**: Las organizaciones no pueden imaginar alojar información crítica fuera de sus fronteras. Ellos piensan que perder el acceso físico y el control de los servidores que alojen esta información significa perder la información de por sí. Estas cuestiones hacen que información sensible pueda ser vulnerable en cuestiones de seguridad y que puedan llegar a agencias de inteligencias u organizaciones competidoras.
- **Conexión a internet**: Para la utilización de los recursos alojados en la nube es imperativo tener conexión a internet. Por ello depende siempre de que se posea esta, limitando el ámbito de utilización de estas.
- **Disponibilidad**: Esto es una característica crucial para la estabilidad y éxito de las empresas. Los proveedores clave de computación en la nube invierten cientos de millones de dólares en su hardware para garantizar el alto nivel de servicio provisto a sus clientes. No obstante, la fiabilidad y la disponibilidad de los servicios en la nube no están garantizadas al 100% debido a circunstancias inesperadas. La carencia en este sentido provoca que las organizaciones tengan una copia de seguridad de sus datos localmente para situaciones de emergencia, lo que equivale a un coste extra.
- **Legislación**: Las leyes relacionadas con situaciones referentes a la computación en la nube, tales como la confiabilidad de dichas soluciones, la disponibilidad de los proveedores, la seguridad de su información, así como situaciones económicas están aún por determinar.

3.3. Clasificación de servicios cloud

En esta sección se expondrán los distintos métodos de clasificación de los servicios en la nube, primero atendiendo al criterio de la forma de despliegue y en segundo lugar al modelo de servicio ofrecido:

3.3.1. Modelos de despliegue

El despliegue de servicios cloud incluye aquellos procesos mediante los cuales se procede a la puesta en marcha de aquella infraestructura y software necesario para que el servicio entre en la fase de producción, es decir, esté listo para su utilización. Según las características de la infraestructura de despliegue se pueden clasificar a un servicio como:

- **Nubes Privadas (Private Cloud):** La infraestructura en la nube es provista para el uso de una única organización con muchos consumidores. Puede pertenecer, ser gestionada y ser operada por una organización, por terceros, o por una combinación de las anteriores.
- **Nubes Comunitarias (Community Cloud):** La infraestructura en la nube es provista exclusivamente para una comunidad de consumidores que tienen asuntos compartidos. Puede pertenecer, ser gestionada y ser operada por una organización, por terceros, o por una combinación de las anteriores.
- **Nubes Públicas (Public Cloud):** La infraestructura en la nube es provista para el uso libre del público en general. Puede pertenecer, ser gestionada y ser operada por empresas, organizaciones gubernamentales o académicas, o una combinación de las anteriores.

- **Nubes Híbridas (Hibrid Cloud):** La infraestructura en la nube es una composición de dos o más infraestructuras cloud distintas que mantienen sus entidades únicas pero que se juntan por estándar o tecnologías que permiten la portabilidad de datos.

3.3.2. Modelos de servicio

Teniendo en cuenta qué tipo de servicio ofrece el proveedor podemos encontrar:

- **Software como servicio (Software as a Service - SaaS):** Las características que se provee al consumidor son las del uso de las aplicaciones de los proveedores las cuales son ejecutadas en infraestructuras cloud. Las aplicaciones se pueden acceder desde distintos dispositivos mediante una interfaz de cliente, tal como navegadores web, o programas específicos para cada dispositivo. El consumidor no gestiona o controla la infraestructura de las aplicaciones.
- **Plataforma como servicio (Platform as a Service - PaaS):** El servicio provisto por este modelo es el del despliegue en infraestructuras cloud de aplicaciones creadas usando lenguajes de programación, librerías, servicios y herramientas soportadas por el proveedor. El consumidor no gestiona ni controla la infraestructura de las aplicaciones.
- **Infraestructura como servicio (Infraestructure as a Service - IaaS):** Provee procesamiento, almacenamiento, red y otros recursos informáticos fundamentales donde el consumidor es capaz de desplegar y ejecutar software arbitrario, el cual puede incluir sistemas operativos o aplicaciones. El consumidor no controla ni gestiona la infraestructura, pero tiene control de los recursos provistos.

3.4. El acuerdo de nivel de servicio

El mayor problema actual del modelo de computación en la nube es la desconfianza que existe entre el cliente y el proveedor debido a los riesgos de esta tecnología y a la novedad de esta. En esta sección se describirá el rol que juega el acuerdo de nivel de servicio para regular la relación entre ambas partes y en el que se regulan sus derechos y obligaciones.

Los términos de un servicio en la nube contratado viene definido por 2 partes que constituyen un mismo documento: El acuerdo de servicio y el Acuerdo de Nivel de Servicio

El acuerdo de servicio es un documento que establece la relación legal entre el cliente y el proveedor de servicios Cloud. En este documento se establecen las reglas legales de la relación entre consumidor y proveedor.

El Acuerdo de nivel de servicio (Service Level Agreement, SLA) establece la calidad del servicio (Quality of Service, QoS) acordada entre clientes y proveedores. Es decir establece los límites mínimos y máximos para distintas propiedades acordadas, así como el procedimiento para reclamar al proveedor del servicio en el caso que se incumpla alguno de ellos, especificando además las compensaciones que se recibirán por el incumplimiento del acuerdo.

El mayor beneficio de la computación en la nube son los recursos compartidos, lo cual esta soportado por la naturaleza de su infraestructura. El SLA es ofrecida por los proveedores de servicio cloud como un acuerdo de servicio.

Cualquier estrategia de gestión del SLA consideran 2 fases: (1) La negociación del contrato y (2) la monitorización en tiempo real del servicio ofrecido. Así, la gestión del SLA engloba el esquema con los parámetros de la calidad de servicio (QoS, Quality of Service).

El punto principal es el de construir una capa sobre la red, la nube, o el middleware SOA capaz de crear un mecanismo de negociación entre los consumidores y los proveedores del servicio.

Resumiendo, el SLA describe la calidad mínima de servicio ofrecida por el proveedor cloud. Por normal general en ámbitos comerciales, los proveedores ofrecen unos términos no negociable que el cliente puede aceptar o buscar otro proveedor de servicios más acorde a sus necesidades. Solamente en casos en la que la organización tenga mucho peso en la contratación se puede llevar a cabo un contrato personalizado.

Este contrato puede ser suspendido en cualquier momento por cualquiera de las dos partes ya sea por alguna causa o por ningún motivo en concreto.

El SLA está formado por 3 partes básicas: (1) Una colección de promesas hechas al consumidor cloud, (2) una colección de promesas específicamente no hechas al consumidor y (3) un conjunto de obligaciones que el cliente tiene que cumplir.

En el caso de que el proveedor falle a la hora de cumplir alguna de los puntos acordados en el SLA, el proveedor deberá dar alguna compensación al cliente del servicio. Esta compensación estará especificada en el SLA.

La desventaja de la computación de la nube en relación con los SLAs es la dificultad de determinar la causa raíz en el caso de que no se cumpla alguna de las clausulas escritas en estas debido a la compleja naturaleza de su infraestructura.

La cláusula referente a la preservación de los datos concierne a los datos por parte del cliente que el servicio cloud puede almacenar y cuánto tiempo pueden estar almacenados en el proveedor. Por norma general, si el cliente rompe alguna de la cláusulas del SLA el proveedor no tiene ninguna obligación de guardar los datos guardados por el cliente. En el caso en el que el cliente sea el que quiera terminar el servicio, sus datos podrían conservarse durante 30 días.

En lo que concierne al uso de la información del cliente. Los proveedores se suelen comprometer a no vender, licenciar o hacer públicos cualquier dato que al cliente concierna con la excepción de que el cliente se lo pida explícitamente.

Por último, las limitaciones en el SLA describen las excepciones a la hora de tratar las clausulas acordadas en el contrato. Generalmente estas suelen ser por paradas de servicios programadas, las cuales no cuentan como una violación en las promesas del servicio.

En la mayoría de los casos el proveedor del servicio cloud no se hace cargo de la monitorización del cumplimiento de estos acuerdos expuestos en el SLA y debe ser el cliente el que debe encargarse de obtener estos datos y realizar las pertinentes reclamaciones. Aun así, el proveedor puede ofrecer métodos o aplicaciones para la monitorización de sus servicios.

3.5. Google App Engine

Google App Engine o también conocido más comúnmente como GAE o App Engine nos abre la infraestructura de producción de Google de forma gratuita como plataforma de desarrollo y hospedaje de aplicaciones web.

El servicio fue lanzado el 7 de abril del 2008 como un servicio de cloud pero a diferencia de otros servicios en la nube como Amazon Web Services o Azure Services Platform de Microsoft, el servicio ofrecido por Google es un servicio de Plataforma como Servicio y no de Infraestructura como Servicio.

GAE soporta de manera oficial los lenguajes de programación Python y Java de manera estable y en modo de beta testing en lenguaje de programación Go creado por ellos mismos. Al soportar Java, es posible además utilizar cualquier lenguaje JVM o lo que es lo mismo, cualquier lenguaje que pueda ejecutarse sobre una máquina virtual de Java, aunque eso sí, con serias limitaciones.

App Engine provee servicios y APIS tales como almacenes de datos NoSQL, memoria caché y APIs de autentificación de usuario, comunes en la mayoría de aplicaciones.

App Engine trabaja con herramientas populares de desarrollo tales como Eclipse, IntelliJ, Maven, Git, Jenkins y PyCharm. Puedes construir tus aplicaciones con la herramienta que más te guste sin cambiar tu flujo de trabajo.

3.5.1. Google Stackdriver.

Google Stackdriver provee herramientas de monitorización, registros y diagnóstico de aplicaciones cloud. Equipa al usuario de visión en la salud, rendimiento y disponibilidad de las aplicaciones, permitiendo al usuario encontrar y solucionar problemas fácilmente. Stackdriver provee una gran cantidad de métricas, dashboards, alerta, gestión de registros, reportes y herramientas de rastreo.

3.5.2. Restricciones

En cuanto a su uso, Google App Engine presenta ciertas limitaciones mencionadas a continuación:

- Las aplicaciones solo tienen permisos de lectura a los archivos del sistema de archivos. Para almacenar datos y archivos en modo lectura y escritura es necesario utilizar un sistema de archivos virtual sobre el DataStore.
- Solo se puede ejecutar código a través de consultas HTTP.
- Las aplicaciones Java solo pueden usar el conjunto considerado seguro de clases del JRE estándar. (Comprueba el Listado de clases)
- Las aplicaciones no pueden crear nuevos hilos de ejecución

- Los usuarios de Python pueden subir módulos para su uso en la plataforma pero no aquellos que están completamente desarrollados en C o Pyrex
- El soporte para SSL solo está disponible par dominios *.appspot.com
- Un proceso iniciado en el servicio para responder a una consulta no puede durar más de treinta segundos
- No soporta sesiones persistentes, solo sesiones replicadas a las que además se les aplican ciertos límites.
- No se pueden abrir sockets, por lo tanto, no se puede usar Twisted

3.6. Otras plataformas Cloud

En esta sección se haremos una breve introducción a las plataformas cloud existentes: Microsoft Azure y Amazon AWS y de las aplicaciones de monitorización que estas plataformas ofrecen

3.6.1. Microsoft Azure

La plataforma Microsoft Azure fue anunciado por primera vez en el evento PDC (Professional Developers Conference) en 2008 como Windows Azure Platform y lanzado en 2010 como Windows Azure es un servicio IaaS y PaaS ofrecido por Microsoft para construir, desplegar y gestionar aplicaciones y servicios alojados en los centros de datos de Microsoft .

Este servicio 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.

Microsoft Azure utiliza un modelo de pago de pay-as-you-go, en el cual el cliente paga el servicio según el número de datos utilizados.

3.6.1.1. Microsoft Azure Diagnostics

Microsoft Azure Diagnostics es un software y conjunto de APIs se encarga de "capturar datos del sistema y de logs procedentes de las máquinas virtuales y las instancias de las máquinas virtuales que se ejecutan en un servicio cloud de Azure y transfieren esos datos a una cuenta de almacenamiento de la elección del usuario". Es por ello el encargado de recuperar los datos de rendimiento de bajo nivel procedentes de los servicios. Estos datos están modelados en forma de instancias de la clase llamada Performance Counter. Un Performance Counter contiene datos relativos a únicamente una métrica de bajo nivel. Estos están clasificados por un nombre que representa la ruta de la procedencia de esta información.

3.6.2. Amazon Web Services (AWS)

Amazon Web Services es una colección plataforma de computación en la nube proporcionada por Amazon.com Los primeros servicios de AWS se lanzaron en 2006 para proporcionar servicios online para páginas webs y aplicaciones cliente. [15]

AWS proporciona algunos servicios incluidos:

- CloudDrive, que permite al usuario subir música, videos, documentos y fotos a aplicaciones conectados a la red. Este servicio solo permite a los usuarios a hacer streaming de la música en sus dispositivos.
- CloudSearch, una servicio de búsqueda usado normalmente para integrar búsquedas personalizadas en otras aplicaciones,
- DynamoDatabase, un servicio de base de datos NoSQL

3.6.2.1. Amazon CloudWatch

AmazonCloudWatch es un servicio de monitorización de los recursos de la nube de AWS y de las aplicaciones que se ejecutan en AWS. Se puede utilizar Amazon CloudWatch para recopilar y realizar el seguimiento de métricas y logs, establecer alarmas y reaccionar automáticamente a los cambios en sus recursos AWS. Amazon CloudWatch puede monitorizar recursos de AWS como, por ejemplo, instancias de Amazon EC2, tablas de Amazon DynamoDB e instancias de base de datos de Amazon RDS, así como métricas personalizadas generadas por las aplicaciones y los servicios, y los logs generados por las aplicaciones. Puede utilizar Amazon CloudWatch para obtener visibilidad para todo el sistema sobre la utilización de recursos, el desempeño de las aplicaciones y el estado de funcionamiento. Puede usar esta información para iniciar y mantener la ejecución de la aplicación sin problemas.

4. Arquitectura de Monitorización basada en Modelos en Tiempo de Ejecución

En esta sección se presenta la arquitectura de monitorización basada en modelos en tiempo de ejecución concebida por Cedillo et al [4] y refinada en posteriores trabajos [5] [6]. Esta arquitectura es la utilizada para la implementación tanto del monitor de servicios desplegados en la nube como en el configurador para la monitorización.

Primero presentaremos la arquitectura propuesta, a continuación se presenta el proceso de monitorización y se finaliza describiendo sus componentes más importantes.

4.1. Presentación de la Arquitectura de Monitorización basada en Modelos en Tiempo de Ejecución

Como hemos comentado anteriormente, la tendencia en la actualidad nos lleva al uso más común de los servicios que nos ofrece la computación en la nube. Siendo esta una tecnología bastante novel, tiene todavía algunos resquicios que hacen que falte mucho por solucionar.

Por esto, la evolución de la computación de la nube está promoviendo el desarrollo de nuevas tecnologías para proveer servicios de alta calidad. La infraestructura de la computación en la nube, en concreto el modelo de software como servicio, provee tanto herramientas que pueden ser usadas por los usuarios como servicios alojados en las plataformas cloud.

Debido a la naturaleza de la computación en la nube, la forma en la que los servicios se construyen y se despliegan ha cambiado. Como resultado de eso, es necesario cumplir una serie de requisitos no funcionales, incluyendo aquellas características específicas de la nube.

Uno de los resquicios mayores de la computación en la nube es el cómo poder determinar qué características se deben ofrecer para que el usuario reciba un servicio de calidad y como, de parte del usuario, se puede tener una certitud de que el servicio que se le ofrece y por el cual está pagando, en el caso en que sea de pago por uso, cumple unos requisitos mínimos. De esto, surge el Acuerdo de nivel de servicio (Service Level Agreement, SLA), el cual se define como un documento formal y negociado entre el proveedor del servicio y el cliente o consumidor del servicio, en el que se detalla el servicio mínimo que se va a ofrecer al consumidor, es decir, las características mínimas que debe cumplir el servicio. Estas características se miden según una serie de métricas, las cuales tienen que estar detalladas en el SLA, y por tanto tienen que poder ser medidas.

Con esto surge un interés común entre las dos partes del contrato, la monitorización del servicio ofertado. Por parte del consumidor, por tener constancia de que se cumple lo acordado en el SLA y en el caso de que no se cumpla poder reclamar y tener una compensación por ello, y, por parte del

proveedor, certeza de calidad de su servicio, pudiendo demostrar que este siempre cumple con los requisitos mínimos que se ofrecen.

Las opciones de monitorización tradicionales están restringidas a un entorno estático y homogéneo, y por ello, no pueden ser aplicadas correctamente a los entornos cloud. En el desarrollo de software tradicional muchas presunciones en el contexto están descritas en su diseño. La computación en la nube conlleva nuevos retos y necesidades a lo hora de evaluar y medir, debido a la a las características especiales de la computación en la nube, tales como la latencia, elasticidad y la escalabilidad.

Cedillo et al [4] propone el uso de Ingeniería dirigida por modelos (Model Driven Engineering, MFE) para enfrentarse a este problema presentando una arquitectura de monitorización basada en modelos de ejecución. Este proceso se centra en comprobar que los requisitos mínimos acordados en el SLA se cumplan así como los requisitos no funcionales que se desee monitorizar.

4.1.1. Modelo en tiempos de ejecución (Models@run.time)

En la Ingeniería dirigida por modelos, un modelo es una abstracción o representación de un sistema construido por un objetivo específico. Los modelos en tiempo de ejecución deben representar el sistema, sus estados y su comportamiento. Si el sistema cambia, los modelos tienen que cambiar también, y viceversa. Es esencial que también se representen por sí mismo. Los modelos en tiempo de ejecución son modelos con un alto nivel de abstracción, en concreto, son modelos que están conectados relativamente con el espacio del problema.

Los modelos en tiempo de ejecución, por lo tanto, se construyen en el reflejo pero buscan dibujar el problema desde la solución de este. Esto nos lleva a la siguiente conclusión: Un modelo en tiempo de ejecución es una auto representación casualmente conectada con el sistema asociado que hace énfasis en la estructura, el comportamiento o los objetivos del sistema desde una perspectiva del espacio del problema.

Así como en los modelos de desarrollo tradicionales, un modelo en tiempo de ejecución soporta el razonamiento. Los sistemas de usuario pueden utilizar los modelos en tiempo de ejecución para soportar la monitorización de estados dinámicos y el control de sistemas durante su ejecución, o para observar el comportamiento de los sistemas en tiempo de ejecución. Un modelo en tiempo de ejecución puede soportar potencialmente la integración semántica de elementos de software heterogéneos en tiempo de ejecución.

En una visión más global, podemos prever la utilización de los modelos de ejecución para arreglar errores o para tomar nuevas decisiones de diseño en sistemas en ejecución para soportar su diseño continuo.

4.1.2. Empleo de los modelos en tiempo de ejecución en la monitorización de servicios cloud

El uso de modelos en ámbitos de la monitorización de servicios permite en tiempo de ejecución detallar y clasificar los requisitos no funcionales a monitorizar, los cuales estarán definidos según unas métricas las cuales pueden ser extraídas

desde la plataforma a monitorizar. Sin embargo no todos los requisitos no funcionales pueden ser medibles en tiempo de diseño, por lo que es necesaria una tecnología que sea capaz de romper una barrera entre el diseño del sistema y el tiempo de ejecución.

Cedillo et al [4] sostiene que resulta útil el modelo en tiempo de ejecución para este propósito, dado que el desarrollador no tendrá que implementar un nuevo modelo y nueva implementación cada vez que aparecen nuevos requisitos, al contrario, con el modelo en tiempo de ejecución, se puede cambiar los elementos a monitorizar y como medirlos sin tener que parar la ejecución de la monitorización, lo cual representa menor coste de despliegue, de construcción y de ejecución. Además este modelo asegura la continuidad de la monitorización sin que tenga un coste muy elevado.

En este modelo, los atributos de la calidad de servicio están directamente conectados con la arquitectura de la monitorización, ya que esta depende de los requisitos no funcionales de estos. Por esta razón, es estrictamente necesario que exista una visión de los requisitos no funcionales a monitorizar con los datos que hacen posible la monitorización de alto nivel.

Por estas razones se decide realizar este proceso tomando como modelo el modelo en tiempo de ejecución.

4.2. El proceso de monitorización

El proceso de monitorización propuesto por Cedillo et al [4] consiste en 3 tareas, la cual se subdividen en una serie de actividades particulares. Este proceso se basa en la técnica de control de bucle autonómico (*autonomic control loop*). La idea de esta técnica es la de medir los parámetros de sistema, analizarlos, diseñar medidas correctivas en el caso de que sea necesario y ejecutar estas acciones en orden de mejorar el sistema. Un beneficio de esta técnica es el de la reducción de interacción humana para lidiar con la baja abstracción, el mantenimiento, y problemas de reusabilidad.

Las tareas a realizar por el proceso de monitorización son las siguientes (1) La Configuración de la monitorización, (2) el proceso de medición y (3) el análisis de los resultados. Como podemos ver en la Figura 1, el proceso de monitorización comienza con la configuración de la monitorización cuyo resultado es el modelo en tiempo de ejecución que va a ser utilizado para el proceso de medición y posteriormente para el análisis del resultado.

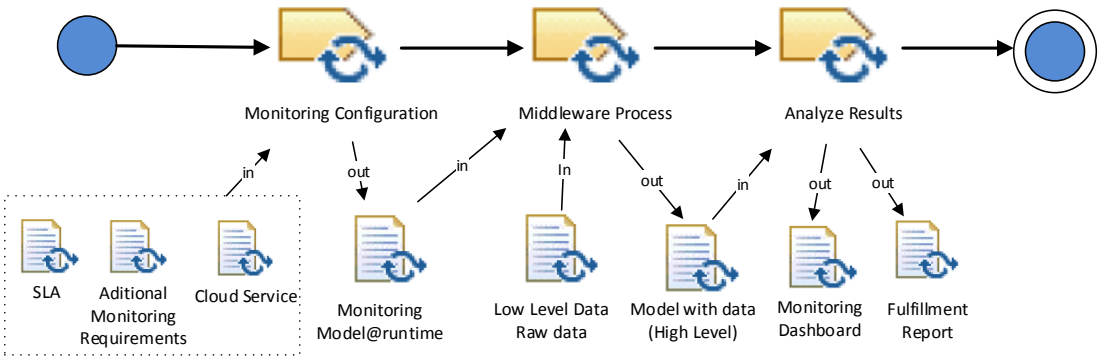


Figura 1. Proceso de Monitorización

El proceso de medición captura la información de bajo nivel de los servicios en ejecución utilizando técnicas que transformaran los datos en datos de mayor nivel los cuales permitirán a la tarea de Análisis de Resultados obtener un informe sobre la calidad del servicio

El proceso de Análisis de resultado utilizara los datos generados en el proceso de medición y los comparara con los requisitos no funcionales especificados crear un reporte en el que se detalle cuando ha fallado.

En el punto siguiente explicaremos as características de cada una de estas tareas así como sus entradas y sus resultados.

4.2.1. Configurador de la Monitorización

El configurador de la monitorización es responsable de la preparación del modelo en tiempo de ejecución. Genera el código mediante transformaciones. Este código va a ser utilizado por el middleware de monitorización para operar con la información sacada de la nube. La Figura 2 especifica el proceso de configuración de la monitorización.

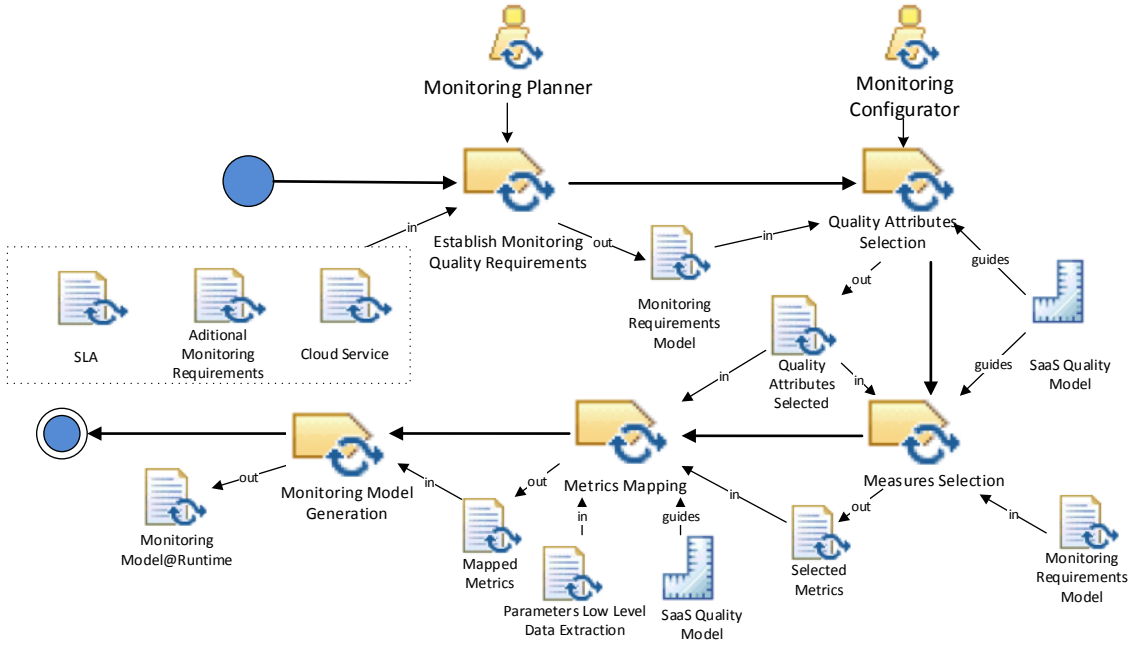


Figura 2. Configuración de la Monitorización

4.2.1.1. Tareas

A continuación se especificarán las tareas que intervienen en la configuración de la monitorización:

1) Establecer los requisitos de calidad para monitorizar (Establising Monitoring Quality Requirements)

Es la primera tarea de este proceso. Esta tarea tiene como entrada tres artefactos: (1) El acuerdo de nivel de servicio (*SLA*) con los requerimientos no funcionales, (2) los requisitos no funcionales adicionales a monitorizar (*Aditional Monitoring Requirements*), y (3) Los artefactos que van a ser analizados en el proceso de monitorización (*Artifacts*).

El resultado de este proceso son las Especificaciones de los Requisitos a monitorizar (*Monitoring Requirements Especification*). Este artefacto contiene las características, métricas y atributos a monitorizar. En el apéndice A se muestra el metamodelo del Modelo de Requisitos de Monitorización.

2) Selección de Atributos de calidad (*Quality Atributes Selection*)

Las características en las que se dividen la calidad y sus medidas asociadas pueden ser de gran utilidad para la evaluación de un producto software. En este trabajo, la calidad de un servicio Cloud se expresa mediante un modelo de calidad alineado con el estándar ISO/IEC 25010 [24] que descompone la calidad de servicio en Características, subcaracterísticas, métricas y atributos de calidad. El modelo también contiene métricas (con una o más operacionalizaciones).

En este proceso, se toma como entrada el documento de Especificación de Requisitos de Monitorización y, utilizando como guía un Modelo de calidad SaaS (*SaaS Quality Model*) y relaciona cada Requisito de Monitorización con el atributo

de calidad SaaS apropiado. En el apéndice A se muestra el metamodelo del Modelo de Calidad SaaS.

El resultado de este proceso es el artefacto de Atributos de Calidad Seleccionados (*Quality Attribues Selected*).

3) Selección de Medidas (*Measures Selection*)

A los atributos de calidad que sacamos en el proceso anterior, es necesario especificar una forma de medirlos, para ello habrá que elegir las métricas para poder medir los atributos.

La Selección de métricas (*Measures Selection*) también utiliza el modelo de calidad SaaS y dependiendo de la perspectiva del usuario, selecciona las métricas apropiadas a aplicar. Es importante incluir la criticidad relativa a los atributos, para poner prioridades a tener en cuenta a la hora de ejecutivas las respectivas medidas correctivas.

El resultado de este proceso sería el conjunto de medidas asociadas a los atributos de calidad seleccionados identificadas como el artefacto Métricas Seleccionadas (*Selected Metrics*)

4) Asociación de métricas

Para monitorizar un servicio desplegado en la nube dependemos de las métricas dependientes de la aplicación. El resultado del proceso anterior es un artefacto con las métricas independientes de la aplicación, por tanto tendremos que mapear las métricas sacadas en el paso anterior con las métricas dependientes de la plataforma.

El resultado de esta tarea es el del artefacto de métricas asociadas (*Mapped Metrics*)

5) Generación del modelo de calidad en tiempo de ejecución (*Monitoring Model Generation*)

Las métricas ya escritas de manera dependientes de plataforma pasarán a formar parte del modelo en tiempo de ejecución (*Monitoring Model@Run.Time*), siendo este el producto final del configurador y la entrada del middleware de monitorización. Por lo tanto, el modelo en tiempo de ejecución contiene todas las directivas de monitorización: los requisitos a monitorizar, las características, atributos de calidad y métricas (tanto las métricas independientes de la plataforma como las específicas de la plataforma cloud), los mecanismos de extracción de datos, etc. En el apéndice A se muestra el metamodelo del modelo en tiempo de ejecución.

4.2.1.2. Artefactos

En esta sección explicaremos los artefactos implicados en el proceso de configuración de la monitorización:

- **SLA (*Service Level Agreement*):** Este artefacto representa al documento del acuerdo del servicio entre el proveedor del servicio cloud y el consumidor de este. En este documento se especifican los niveles mínimos y máximos acordados acerca del rendimiento del servicio contratado así como el procedimiento para la reclamación por incumplimiento de alguno de estos acuerdos, especificando a su vez la compensación por ello.
- **Requisitos adicionales de la monitorización (*Additional Monitoring Requirements*):** Artefacto que representa los requisitos de monitorización no incluidos en el SLA pero incluidos en la monitorización debido a interés del consumidor.
- **Servicio en la nube (*Cloud Service*):** Este artefacto representa a la información necesaria relativa al servicio el cual va a ser monitorizado (tipo de servicio, datos de acceso, nombre del servicio, configuración, etc…).
- **Modelo de requisitos de la monitorización (*Monitoring Requirements Model*):** Este artefacto representa a los requisitos que deberán tenerse en cuenta durante la monitorización, los límites que deben mantener y la forma en lenguaje natural en la que deben medirse. Este modelo se puede encontrarse detallado en la sección 1.1 del Anexo
- **Modelo de calidad SaaS (*SaaS Quality Model*):** Este artefacto representa el modelo de calidad específico para la medición de la calidad de servicios en la nube provisionados como SaaS. Este modelo se toma como referencia a la hora de elaborar métricas válidas para la medición de atributos de calidad. Se puede encontrar el detalle de este modelo en la sección 1.2 del Anexo.
- **Atributos de calidad seleccionados (*Quality Attributes Selected*):** Este producto es el producto intermedio de la tarea. Contiene aquellos atributos que se han seleccionado para representar los requisitos no funcionales presentes en el modelo de requisitos de monitorización.
- **Métricas seleccionadas (*Selected Metrics*):** Este artefacto contiene las métricas independientes de plataforma escogidas para la medición de los atributos de calidad seleccionados.
- **Parámetros de bajo nivel de extracción de datos (*Parameters Low Level Data Extraction*):** Este artefacto contiene la información relativa a los diferentes tipos de datos que pueden extraerse del servicio y la plataforma donde se despliega y que podrán emplearse para medir atributos de calidad.

- **Métricas asociadas (*Mapped Metrics*):** Este artefacto es un producto intermedio de la tarea. Representa las métricas dependientes de la plataforma creadas para ser equivalentes a las métricas seleccionadas para medir atributos de calidad de los requisitos no funcionales seleccionados.
- **Modelo de monitorización en tiempo de ejecución (*Monitoring Model@Run.Time*):** Es el producto final del configurador de la monitorización. Este modelo contiene toda la información referente a la

monitorización, es decir, los datos del servicio cloud, y los atributos, métricas, fórmulas, etc... Para la monitorización del servicio cloud.

4.2.1.3. Roles

En esta sección explicaremos los roles implicados en el proceso de configuración de la monitorización:

- **Planificador de la monitorización**: Este rol debe de ser desempeñado por un usuario con conocimientos sobre los acuerdos de nivel de servicio alcanzados. A su vez debe tener conocimientos generales de la plataforma cloud y ser capaz de formular estos conceptos facilitando su medición y estableciendo los límites aceptados.
- **Configurador de la monitorización**: Este rol es desempeñado por un usuario experto del Modelo de Calidad Saas, con conocimientos de calidad de software y cuyos conocimientos sobre las métricas dependientes de plataforma sean bastante amplios para poder hacer una asignación entre las métricas del modelo de atributos de calidad SaaS y las métricas dependientes de la plataforma.

4.2.2. Proceso de Medición

El proceso de medición está incluido en el middleware de monitorización que recupera la información de los servicio y aplicaciones y provee información acerca de la monitorización a los usuarios y proveedores de servicios cloud. Usa el modelo en tiempo de ejecución descrito anteriormente (Ver punto 4.1.1) y usa un motor de medición para medir los atributos. La comunicación entre los servicios y el middleware es implementada mediante técnicas que permiten la comunicación bidireccional entre el middleware de monitorización y los servicio cloud.

El motor de análisis recibe la información del motor de medición y la compara con la información aportada por el SLA y los otros requerimientos no funcionales. El middleware provee los resultados los cuales pueden ser útiles a la hora tomar acciones a la hora de mejorar la calidad de la nube y la calidad del SLA.

4.2.3. Análisis de los resultados

El motor de análisis es parte del middleware de monitorización, su trabajo es el de comparar los valores obtenidos por el proceso de monitorización con los requerimientos no funcionales, analizar los resultados y hacer un reporte del análisis. Los resultados obtenidos por el sistema de monitorización se pueden utilizar para planear una estrategia para cambiar la infraestructura utilizando arquitecturas de reconfiguración que utilicen un sistema experto o un conocimiento base, adaptando su propio sistema y soportando el cumplimiento de los requisitos no funcionales, cerrando el bucle de control autonómico.

4.3. Componentes del Proceso de Monitorización

En esta sección se representara la estructura de los componentes de alto nivel que compondrán la estructura del monitor de servicios cloud. En la Figura 3. Se destacan dos grandes componentes:

- El Configurador del Middleware (*Middleware Configurator*)

- El Middleware de Monitorización y Análisis (*Monitorig and Analysis Middleware*).

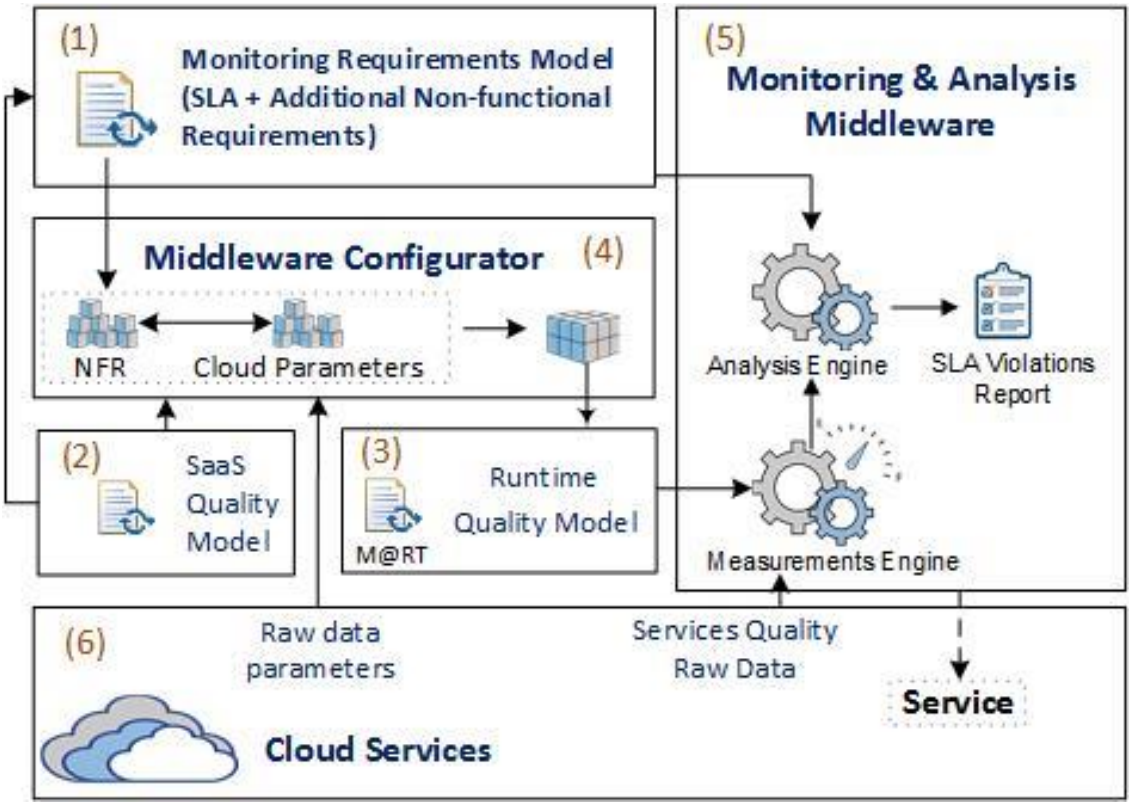


Figura 3. Componentes del proceso de monitorización

A continuación detallaremos cada uno de estos componentes.

4.3.1. Configurador del Middleware

En este proceso, un usuario con conocimientos respecto al modelo de calidad SaaS, al acuerdo de nivel de servicio (SLA) y a los parámetros dependientes de la plataforma, escoge una serie de parámetros para crear el modelo en tiempo de ejecución (*Monitoring Model@Run.Time*). Este usuario carga los datos del servicio cloud a monitorizar, carga el SLA para mapear los requerimientos con lo acordado con el usuario en el SLA, escoge los requisitos no funcionales del SLA y escoge del modelo de calidad SaaS las características, subcaracterísticas, atributos y métricas independientes de plataforma y posteriormente con estos escogerá las métricas dependientes de plataforma para monitorizar. Con esto creará las funciones para poder hacer los cálculos y mediciones para analizar los datos. Con todo esto se creará el documento XML el cual contiene el modelo en tiempo de ejecución que será el documento de entrada del middleware de monitorización.

4.3.2. El Motor de Monitorización y Análisis

El motor de monitorización y análisis es una parte esencial del proceso de monitorización, en él se producen las actividades nucleares del proceso de extracción y cálculo de datos.

El motor de Monitorización y Análisis se encarga de extraer los datos relacionados con los atributos de calidad del servicio y realizar las operaciones necesarias para obtener los valores necesarios a comparar con los datos a analizar.

Este proceso está dividido en tres componentes especializados: El Motor de Medición (*Measurement Engine*), el Mecanismo de recolección de datos de la plataforma (*Platform Data Retrieval Mechanism*) y el Motor de Análisis (*Analysis Engine*).

4.3.2.1. El Motor de Medición

El motor de medición es el encargado de realizar los cálculos sobre los valores extraídos por el Mecanismo de Recolección de datos de la plataforma.

Este componente calcula las métricas basadas en las operacionalizaciones detalladas en el modelo en tiempo de ejecución para la monitorización, después de calcularlas las guarda para su posterior análisis.

4.3.2.2. El Motor de Análisis

El Motor de análisis es el encargado de reportar si alguno de los datos recolectados se sale de los límites establecidos en el SLA.

Este componente analizara los cálculos obtenidos del motor de medición con los límites acordados en el acuerdo de nivel de servicio. Creará un reporte en el cual dejará si alguno de estos cálculos de los requerimientos no funcionales sobrepasa los límites establecidos, para que bien el usuario o el proveedor del servicio puedan tomar medidas al respecto en el asunto. Por parte del usuario, la reclamación y posterior cobro de la compensación y por parte del proveedor la posibilidad de mejora de su servicio.

4.3.2.3. El Mecanismo de recolección de datos de la plataforma

En este proceso se encuentran los mecanismos para recolectar los datos de la plataforma. Es decir las llamadas a la aplicación correspondiente y trata los datos de la manera pertinente. Este proceso depende de la plataforma cloud en la cual está desplegada el servicio, ya que está pondrá a disposición una API a la cual este proceso deberá llamar con una nomenclatura especial. Además se encargará de almacenar los datos para que estos puedan ser medidos y que quede constancia de estos.

5. Definición del Configurador de monitorización de calidad de servicios de aplicaciones desplegadas en Google App Engine

En este capítulo se expone la implementación del proceso de configuración de la monitorización de servicios en la nube. Primero, se expone las herramientas y tecnologías utilizadas para llevar al cabo el desarrollo del configurador. A continuación, se expone un pequeño análisis de la plataforma seleccionada para la implementación y la influencia que tendrá en la implementación de este proceso. En el siguiente punto, se expondrá el configurador de la monitorización en detalle y finalmente su utilización por el Middleware de Monitorización.

5.1. Patrón de diseño

El configurador está organizado basado en el patrón de diseño de interfaces de usuario del Asistente (Wizard) .Este patrón tiene como objetivo dividir una tarea en subtareas más pequeñas de manera que sea más sencillo para el usuario completar la tarea principal, en este caso crear el modelo de calidad en tiempo de ejecución.

Tomando como guía el diagrama del proceso de configuración de la monitorización resulta evidente el carácter lineal del proceso de generación del modelo. Así mismo, el proceso es complejo y largo debido al tipo y número de decisiones que el usuario tiene que tomar por lo que disminuir la complejidad de las interfaces es una prioridad de primer nivel.

De igual manera, debido a la complejidad de la acciones a realizar y de los conceptos empleados, es necesario que al usuario se le guie a través de la aplicación indicándole las acciones que debe tomar a continuación.

Así mismo, la posibilidad de incluir la totalidad del proceso en una sola interfaz fue puesto a prueba en versiones preliminares (véase Figura 4) y se demostró que la cantidad de elementos y posibilidades podría crear confusión al usuario y reducir las funciones de la aplicación con lo que quedó descartada.

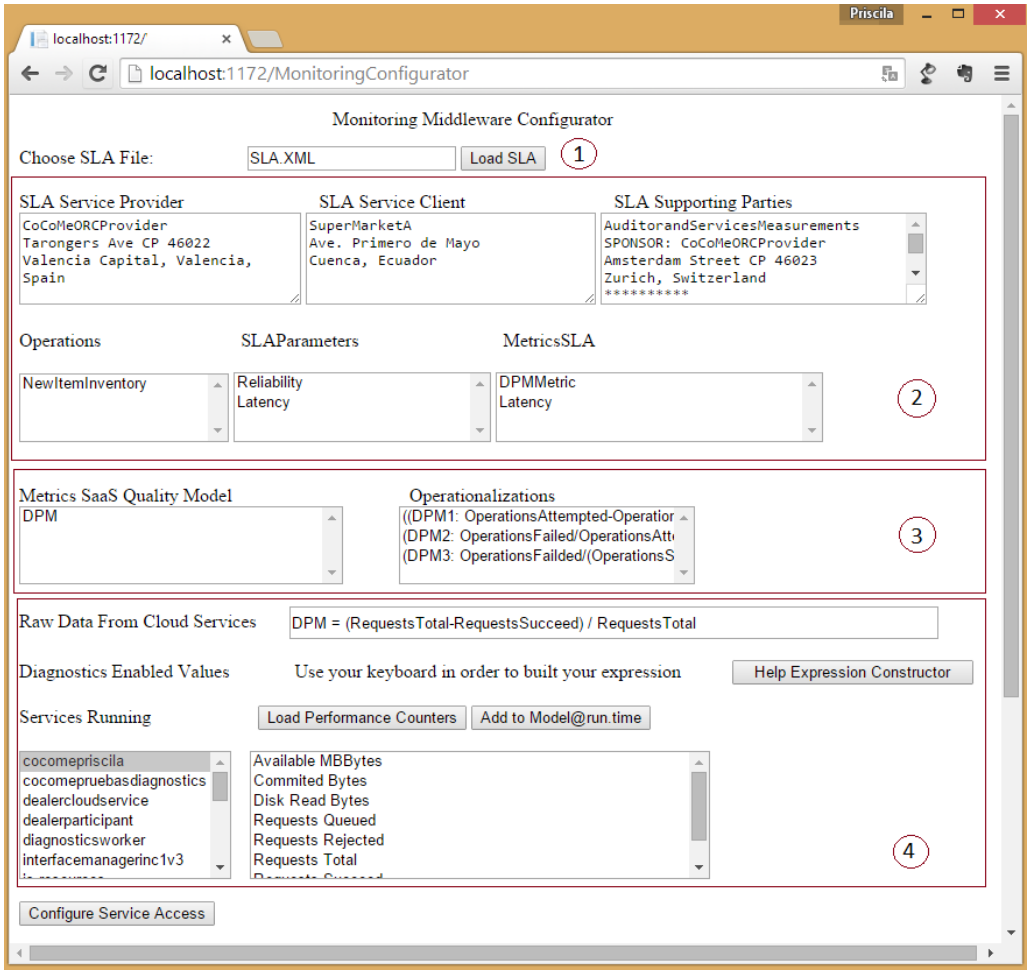


Figura 4 - Prototipo del Configurador de la Monitorización.

El diseño final de la aplicación presenta el habitual diseño de interfaz presente en aplicaciones que utilizan este patrón, ocupando la parte central de la pantalla los formularios a rellenar y situando en la parte inferior los botones de avance y retroceso para navegar entre tareas como puede verse ilustrado en la Figura 5.

Figura 5 - Ejemplo de interfaz con el patrón Asistente.

A continuación, se procede a realizar un recorrido guiado de la aplicación detallando los puntos más importantes de la misma. El flujo de la aplicación puede verse ilustrado en la Figura 6.

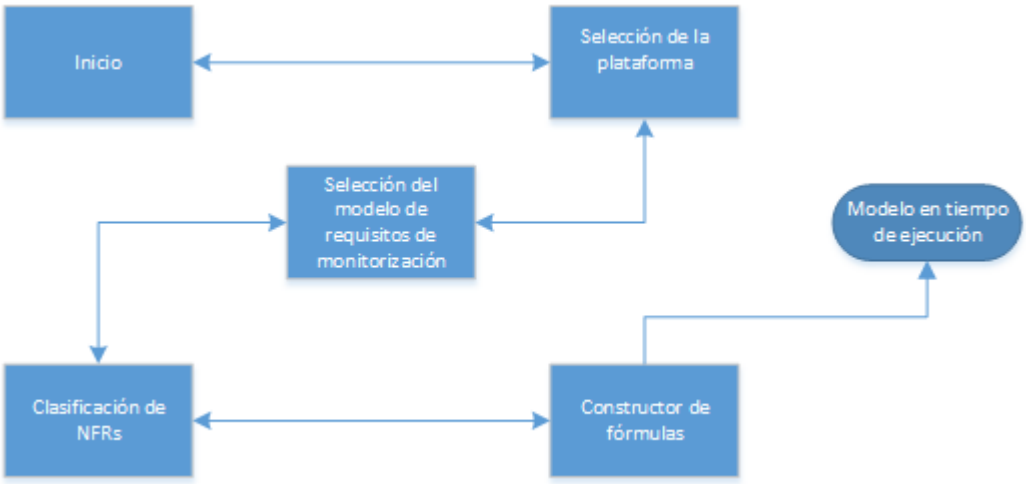


Figura 6 - Flujo de la aplicación web

5.2. Introducción a las herramientas y tecnologías utilizadas

Para definir las tecnologías a utilizar para el desarrollo del configurado de la monitorización de aplicaciones desplegadas en Google App Engine, definimos las siguientes subcategorías según el nivel de detalle de estas y su uso:

- Despliegue
- IDE
- Lenguaje de programación (Backend)
- Frameworks
- Interfaz de Usuario
- Base de datos
- Frontend

5.2.1. Despliegue

La implementación del middleware requería de una plataforma en la nube debido a que la instancia de este requiere de alta disponibilidad dado que este proceso debe de estar activo de forma continua para que el procesamiento de resultados aportados en tiempo real por el servicio a ser monitorizado se realice con el mínimo retraso posible de manera que situaciones de riesgo sean detectadas a tiempo.

La plataforma en la nube escogida ha sido Google App Engine, una destacada plataforma como servicio (PaaS). Se ha escogido esta plataforma por distintas razones, tanto como por seguir la línea de proyecto (Utilización del mismo servicio a analizar), hasta por temas de rendimiento, versatilidad en el uso de diferentes lenguajes de programación y facilidad de despliegue directo desde Entornos de desarrollo tales como Eclipse. Además Google App Engine ofrece a los desarrolladores frameworks para la utilización de su API de monitorización para los lenguajes de programación que este soporta, facilitando su desarrollo en estos mismos y siendo Google App Engine la mejor opción de despliegue.

Por lo tanto, el configurador ha sido implementado en la misma plataforma y con las mismas tecnologías utilizadas por el monitor, para facilitar la interoperabilidad de ambas herramientas.

5.2.2. Entorno de desarrollo

El Entorno de desarrollo (IDE) es una aplicación que contiene diferentes herramientas para la facilitación en el desarrollo o programación de aplicaciones. Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Se ha elegido este IDE por la variedad de herramientas que facilitan la creación de aplicaciones, como la que proporciona Google App Engine para su despliegue directo y que se incorpora a Eclipse.

5.2.3. Lenguaje de programación

El lenguaje de programación utilizado es Java. Lenguaje de propósito general, concurrente, orientado a objetos.

La elección de Java para el desarrollo de la parte backend de la aplicación se determinó por su robustez, su amplia documentación encontrada en la red y su dinamismo a la hora de crear aplicaciones.

Además Java es uno de los lenguajes soportados por Google App Engine.

5.2.4. Frameworks

El proyecto del configurador de la monitorización de servicios desplegados en Google App Engine es un proyecto Maven Spring MVC (Modelo Vista Controlador).

5.2.4.1. Spring

Spring es un framework para el desarrollo de aplicaciones, y contenedor de inversión de control, de código abierto para Java. Una de sus funcionalidades es el Modelo Vista Controlador, que permite gestionar las distintas partes del proyecto, gestiona las distintas dependencias entre clases y tipos de archivos.

Además gestiona las llamadas REST, facilitando el uso de distinto tipos de anotaciones que crea toda la lógica de detrás de la aplicación.

Spring incorpora distintos módulos que simplican el tratamiento de objetos y el mapeo de estos con distintos tipos de archivos (JSON, XML), como por ejemplo Spring's Object/XML Mapping que ayuda a la transformación de documentos XML en objetos y clases java y viceversa.

5.2.4.2. Maven

Maven es una herramienta software para la gestión y construcción de proyectos Java. Maven utiliza un Project Object Model para describir el proyecto software a definir, sus dependencias a otros módulos y componentes externos y el orden de construcción de los elementos.

5.2.5. Base de datos

La base de datos utilizada es MySQL, base de datos relacional la cual está incorporada en los servicios ofertados por Google App Engine.

Para la conexión, recolección y tratamiento de los datos desde la aplicación se utiliza Hibernate que facilita la utilización y el mapeo entre los datos guardados en base de datos y los objetos de la aplicación.

5.2.5.1. MySQL

MySQl es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y esta consideradao como la base de datos opern source más popular del mundo y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

5.2.5.2. Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

5.2.6. Frontend

Para la creación de la interfaz de usuario se ha utilizado HTML para la creación de la estructura, Javascript como lenguaje para el tratamiento de datos y JQuery para funcionalidades de interacción con la aplicación.

5.2.6.1. HTML

Lenguaje de Marcas de Hipertexto (*Hypertext Markup Language, HTML*) es un lenguaje marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web es sus diferentes versiones, define una estructura básica y un código para la definición de una página web como texto, imágenes, etc.... Es un estándar a cargo del World Wide Web Consortium (W3C) [19].

5.2.6.2. Javascript

Lenguaje de Marcas de Hipertexto (*Hypertext Markup Language, HTML*) es un lenguaje marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web es sus diferentes versiones, define una estructura básica y un código para la definición de una página web como texto, imágenes, etc.... Es un estándar a cargo del World Wide Web Consortium (W3C) [19].

5.2.6.3. JQuery

JQuery es una biblioteca de Javascript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

5.3. Google App Engine

Google App Engine es una aplicación para construir aplicaciones web escalables y backend para aplicaciones móviles. App Engine provee servicios y APIS tales como almacenes de datos NoSQL, memoria caché y APIs de autentificación de usuario, comunes en la mayoría de aplicaciones.

App Engine trabaja con herramientas populares de desarrollo tales como Eclipse, IntelliJ, Maven, Git, Jenkins y PyCharm. Puedes construir tus aplicaciones con la herramienta que más te guste sin cambiar tu flujo de trabajo.

Sobre esta plataforma se desplegarán los servicios de monitorización y la página web que componen este proyecto.

5.4. El configurador de la Monitorización

El configurador de la monitorización es una aplicación web cuyo objetivo es asistir al usuario durante la tarea de "configuración de la monitorización" del proceso de monitorización (ver Figura 2). Esta herramienta ha sido diseñada para dar soporte a las distintas subtareas que componen la Configuración de la Monitorización cuyo objetivo es la elaboración del modelo en tiempo de ejecución que será empleado posteriormente por el monitor para evaluar la calidad del servicio y comprobar el cumplimiento del acuerdo SLA.

El usuario de esta interfaz no podrá ser el usuario de la aplicación web, sino más bien un usuario experto que tenga conocimientos del modelo de calidad de servicio, de las características de la plataforma cloud en la cual está desplegada la aplicación o servicio a monitorizar, y de los acuerdos de nivel de servicio, ya que será el encargado de hacer un mapeo entre los requisitos no funcionales del acuerdo de nivel de servicio, las características de calidad y métricas del servicio y las métricas que dependen de la plataforma cloud.

Para conseguir este objetivo, se ha optado por la implementación de una aplicación web dado que una característica propia de un proyecto cloud es la capacidad de acceso desde cualquier parte del mundo al servicio, por lo que una herramienta que de soporte a un servicio de estas características es recomendable que comparta esta propiedad, en contra de las características de una posible versión de escritorio. Así lo demuestra la tendencia actual de trasladar las aplicaciones de escritorio a la nube, incluso una herramienta tan especializada como la presente. Además, la tecnología web hace posible el uso de la aplicación desde cualquier dispositivo.

Según las subtareas de configuración de la monitorización la aplicación debería ser capaz de:

- Recoger los Requisitos de Calidad para la monitorización especificados por un planificador de la monitorización.
- Ser capaz de contener un modelo de calidad SaaS que ofrezca asistencia al usuario para realizar la configuración.
- Clasificar los requisitos de monitorización relacionándolos con los atributos de calidad presentes en el modelo de calidad SaaS.
- Seleccionar de entre las métricas del modelo de calidad y asociar a cada atributo de calidad asociado a un requisito de monitorización una métrica.

- Ser capaz de expresar las métricas procedentes del modelo de calidad SaaS utilizando las propiedades específicas de la plataforma.

- Generar un modelo en tiempo de ejecución que será consumido por el Middleware de Monitorización para realizar la configuración de la monitorización del servicio.

5.4.1. Selección de plataforma

Se trata del primer paso del configurador de la monitorización y está relacionado con los datos de la aplicación o servicio desplegado en la nube, el cual va a ser monitorizado.

En este paso, el usuario debe de introducir los datos del servicio que desee monitorizar (Figura 7) y escoger la aplicación a monitorizar. Para empezar, se deberá seleccionar la plataforma en la que el servicio está desplegado. En este prototipo se da soporte únicamente a Google App Engine dado el requerimiento de un profundo conocimiento de la plataforma para realizar la configuración de la plataforma.

A continuación, se deberán de introducir las credenciales propias del servicio: Nombre de la aplicación a monitorizar, el identificador del despliegue y la cadena de conexión. El identificador del despliegue determina únicamente el servicio dentro de la plataforma, mientras que la cadena de conexión incluye la información necesaria para acceder a los recursos de almacenamiento de Google, de esta manera se concede el acceso, necesario, a la aplicación para realizar operaciones de comunicación con los datos de monitorización del servicio y el Middleware de Monitorización.

Al ser todos los campos obligatorios (ya que si no la plataforma no podrá ser monitorizada), en caso de dejar algún campo vacío, la aplicación dará un error obligándote a rellenar todos los campos.

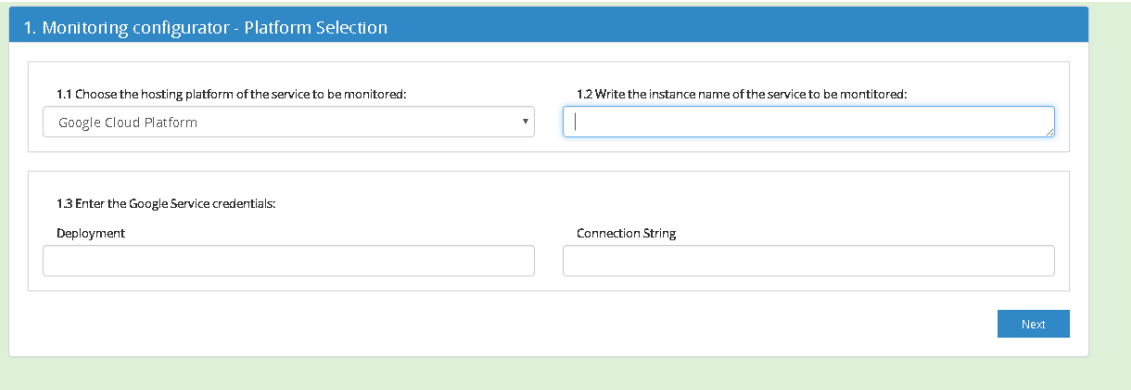


Figura 7 - Selección de la plataforma

5.4.2. Selección del modelo de requisitos de monitorización

El modelo de requisitos de monitorización se realiza fuera de los límites de la aplicación. Este documento contiene la información relativa a los requisitos no funcionales a monitorizar que será necesaria para establecer las métricas capaces de medir estos requisitos.

En esta implementación, cuyo diseño se observa en la Figura 8, se ha optado por aceptar este modelo realizado en formato XMI (*XML Metadata Interchange*, XML de Intercambio de Metadatos) un formato estándar ISO/IEC 19509:2014 que tiene como objetivo "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".

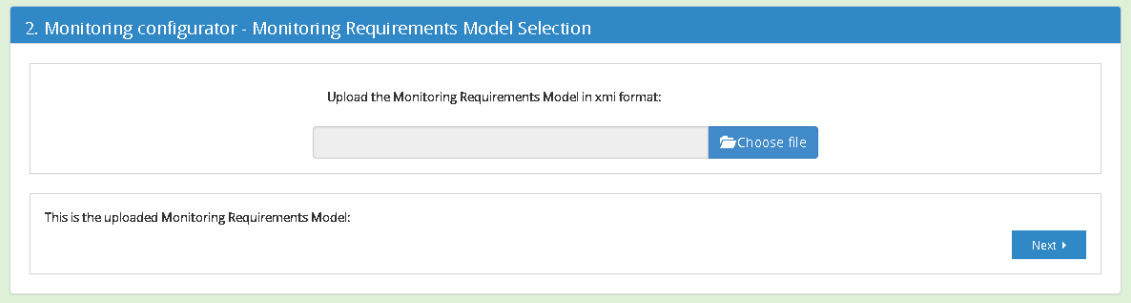


Figura 8 - Selección del modelo de requisitos de monitorización.

Aunque en esta ocasión el empleo de XMI está ligeramente desviado de su intencionalidad, pues en esta aplicación se utiliza como contenedor de los datos estructurados de requisitos de monitorización, se ha creído conveniente su uso dado que la aplicación debe trabajar conjuntamente con otras que modifiquen este tipo de modelos. En este caso los modelos son generados desde Eclipse a falta de una herramienta propia para el diseño de los mismos.

Una vez importado el modelo haciendo clic en Choose File y seleccionándolo en el explorador, si se ha importado correctamente en la parte inferior aparece una tabla con la información, resumida, más relevante para el usuario (ilustrado por la Figura 9). En caso de no ser un archivo valido, el sistema dará un aviso como que el archivo no es válido. En este caso se deberá de elegir un archivo válido:

- **NFR Name**: Nombre del requisito no funcional.
- **Threshold**: Expresión del umbral, mayor, menor o igual que el requisito no funcional debe cumplir por acuerdo de los stakeholders. Puede estar expresada como un porcentaje o un número decimal.
- **Metric Name**: Nombre de la métrica o métricas elegidas como posibles candidatas a medir el RNF asociado.
- **Metric Formula**: Fórmula, en lenguaje natural que expresa la forma en la cual se puede medir una métrica. Una métrica puede medirse con una o más fórmulas.

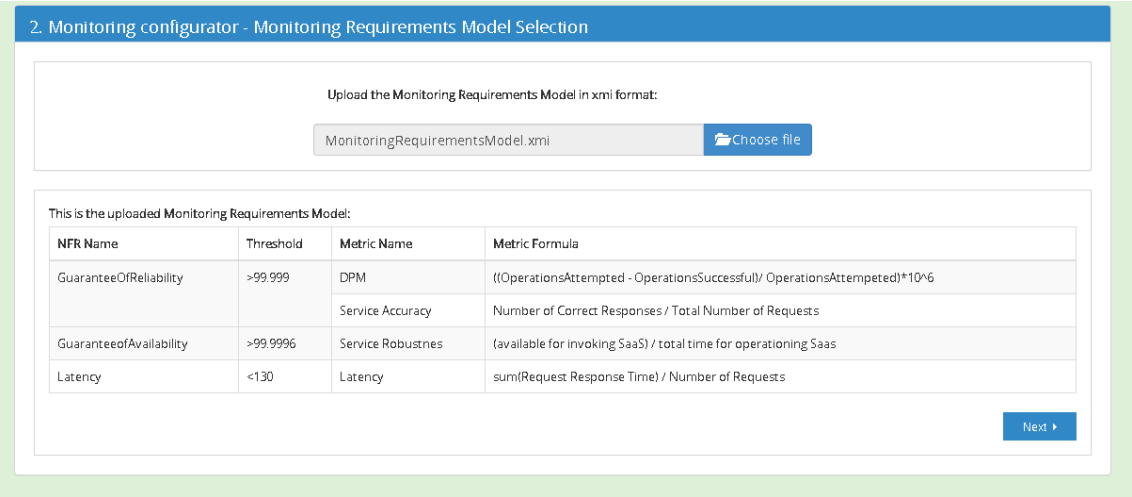


Figura 9 - Modelo de requisitos de la monitorización cargado en la aplicación.

5.4.3. Clasificación de requisitos no funcionales

En esta interfaz, el usuario tiene por objetivo seleccionar y clasificar los requisitos no funcionales a monitorizar usando un Modelo de Calidad SaaS como guía. Para ello, el usuario deberá seleccionar uno por uno los requisitos no funcionales, clicando sobre una tabla idéntica a la descrita en el punto anterior situada bajo la explicación del paso 3.1 (véase en la Figura 10). Una vez seleccionado, se procederá a realizar la clasificación en el paso 3.2. En este se procederá a realizar la clasificación dentro de las ramificaciones del modelo de calidad SaaS, tal y como indica la segunda tarea del modelo de proceso, seleccionando para ello primero la característica a la que pertenece de la lista situada bajo el título *Characteristic*, la subcaracterística (en el caso de que sea necesario y exista una para la característica seleccionada aplique) navegando en el árbol de subcaracterísticas, el atributo y finalmente la métrica en sus respectivos controladores (Figura 11).

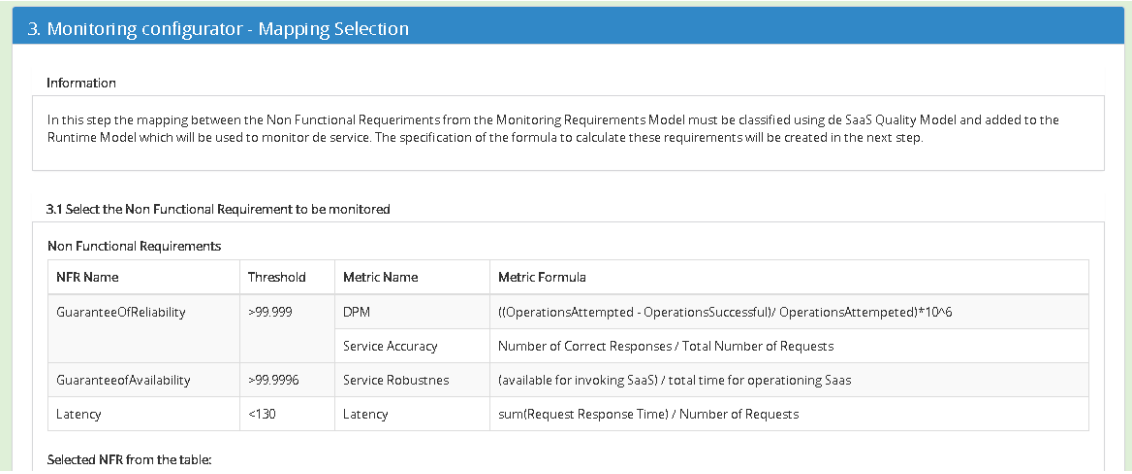


Figura 10 - Asociación de Métricas: Los RNFs.

El modelo de Calidad SaaS ya está introducido internamente como datos de la aplicación y de igual manera que el Modelo de Requisitos de Monitorización está

portado en formato XMI por lo que puede ampliarse o modificarse sin modificar la implementación. Se ha utilizado un modelo de calidad SaaS prototipo ya que la creación de un modelo de calidad SaaS completo se encuentra en el momento de escribir este trabajo en fase de investigación.

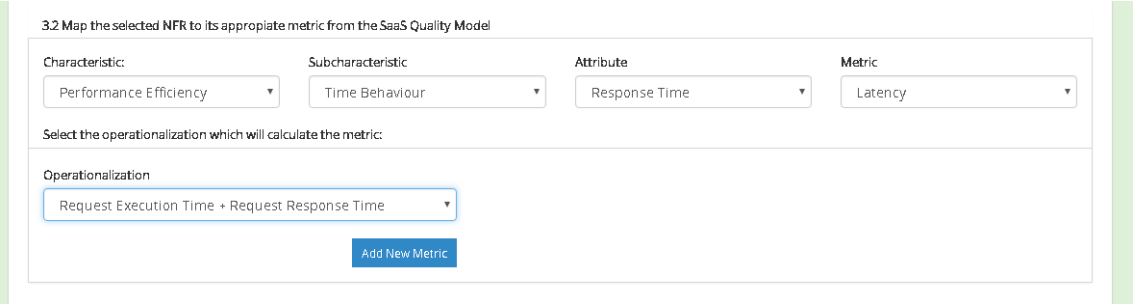


Figura 11 - Asociación de Métricas: Clasificación en el modelo de calidad.

En el momento de realizar la clasificación, el usuario deberá tener conocimiento teórico sobre este modelo de calidad y deberá ser capaz de, tomando los datos del modelo de requisitos, ser capaz de encontrar la clasificación más apta para los RNFs usando los conceptos del modelo de calidad.

Una vez hecha la clasificación, siguiendo la subtarea tres del modelo de proceso, la Selección de Métricas, se procede a seleccionar la operacionalización del controlador para ello habilitado bajo el título Operationalization. Esta operacionalización describe la forma de medir la métrica de manera general, en términos independientes de la plataforma. Si la clasificación es correcta, se procederá a añadirla al modelo en tiempo de ejecución resumido en la tabla bajo el título: "Selected metrics added to the Model@Runtime". En caso de haber un error en la clasificación en este mismo apartado puede seleccionarse la métrica erróneamente clasifica clicando en la papelera que aparece para proceder al borrado del mismo.

Una vez terminada la clasificación de todos los requisitos no funcionales a monitorizar se procede al siguiente paso pulsando siguiente. En la tabla situada en la sección del modelo en tiempo de ejecución (Figura 12) aparecerá entonces una nueva fila conteniendo la información de la clasificación:

- **NFR#**: Número de requisito no funcional. Identifica al requisito no funcional de forma única.
- **Attribute**: Nombre del atributo bajo el cual se ha clasificado el RNF. No es necesaria la información pertinente a las características y subcaracterísticas dado que cada atributo es la hoja del árbol de características por lo que la clasificación ya las incluye. No debe haber dos atributos de mismo nombre en el modelo de calidad SaaS.
- **Metric**: Nombre de la métrica en la cual se ha clasificado el RNF.
- **Operationalization**: Nombre de la operacionalización, en términos independientes de la plataforma, que ha sido elegida para medir la métrica.

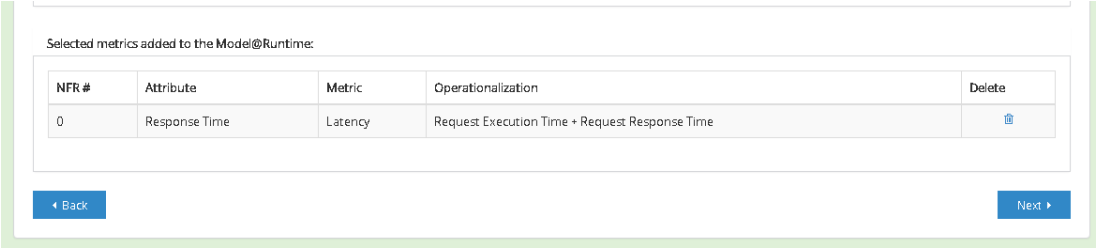


Figura 12 - Asociación de métricas: El modelo en tiempo de ejecución.

5.4.4. Constructor de fórmulas

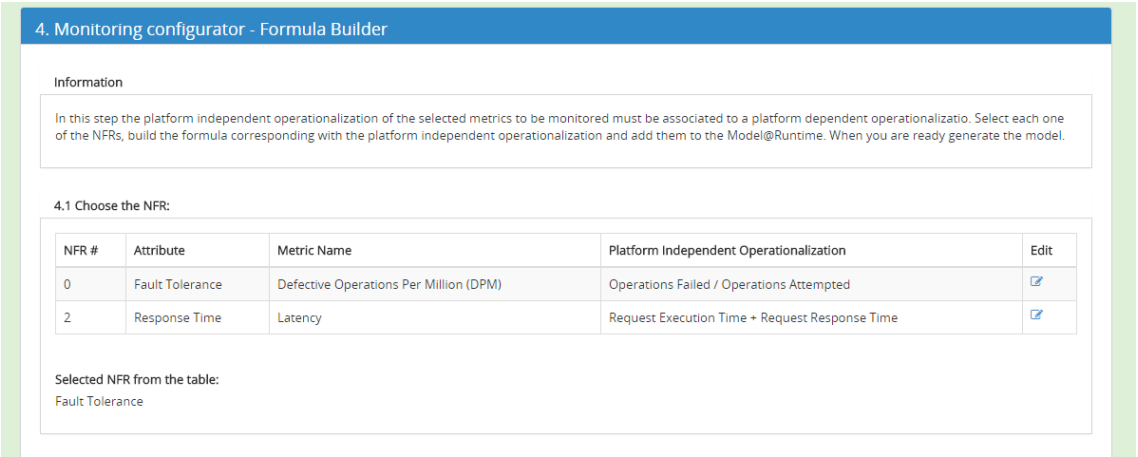
Esta interfaz da soporte a la última parte de la cuarta subtarea del modelo: la Asociación de Métricas. Una vez clasificadas y asignada una operacionalización independiente de la plataforma a la métrica asociada a un requisito no funcional, el usuario debe de escribir esta operacionalización usando términos propios de la plataforma.

En el caso del presente prototipo, eso implica traducir las fórmulas en lenguaje natural en fórmulas especificando los *Performance Counters* procedentes de *las métricas de la Api de monitorización de Google App Engine* apropiados o en su defecto los contadores personalizados creados con este propósito. Esta fórmula será la que el motor de medición interpretará para extraer la información de rendimiento y realizar las operaciones dispuestas en ella y que constituirán una medida de un atributo de calidad.

Para realizar este proceso primero se debe seleccionar uno de los RNF expuestos en la tabla bajo el apéndice 4.1 clicando sobre el *Checkbox* de la fila del RNF deseado (véase Figura 13).

En esta tabla aparece 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 Independent Operationalization**: Operacionalización seleccionada formulada en términos independientes de la plataforma.



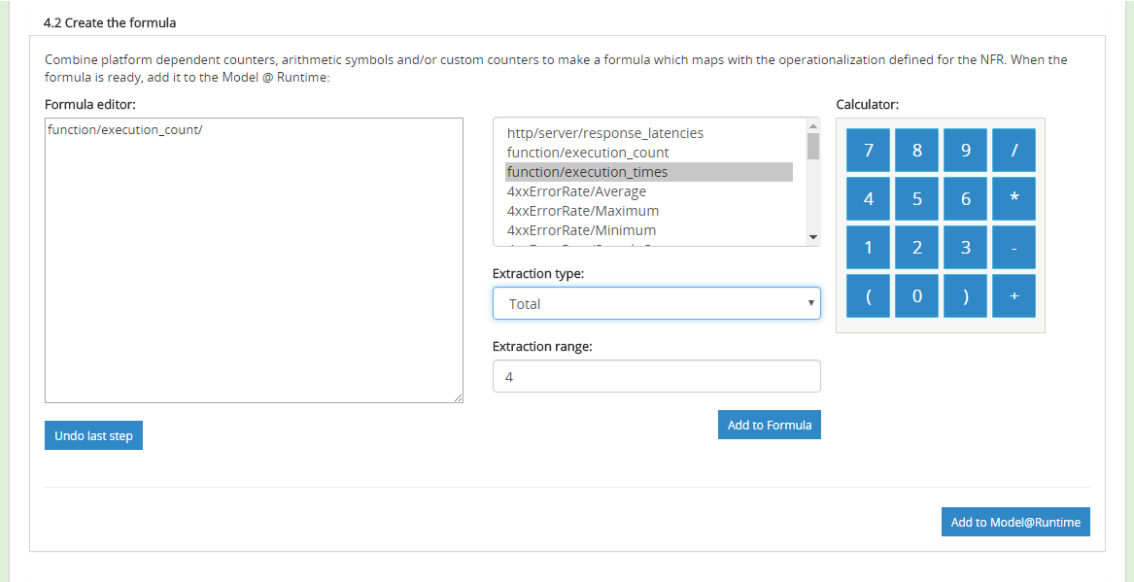


Figura 13 - Constructor de fórmulas.

Una vez seleccionado el RNF se procederá al segundo apartado del formulario (4.2) dónde el usuario seleccionará de entre los *Performance Counters* de Google app Engine, los términos de la calculadora (símbolos y números naturales) y los *Custom Counters* (Contadores Personalizados) disponibles para realizar la formula dependiente de la plataforma tomando como referencia la fórmula independiente tomada del modelo de calidad SaaS y que puede observarse en la tabla del apartado 4.1 (Figura 13).

Para añadir un contador (ya sea un *Performance Counter* o un *Custom Counter*) deberá seleccionar la opción del menú clicando 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 clicando sobre él y:

1. 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).
2. 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.
3. Pulsar *Add ToFormula*: El contador aparecerá en el cuadro de texto de la fórmula y podrá elegirse otro ítem para continuar con la construcción de la fórmula siguiendo el mismo proceso.

En el caso de producirse errores al introducir nuevos elementos en la fórmula puede deshacerse el último elemento añadido pulsando el botón *Undo last step*. En caso de que la fórmula 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 (Figura 14) 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**: Fórmula construída en los pasos anteriores con elementos dependientes de la plataforma.

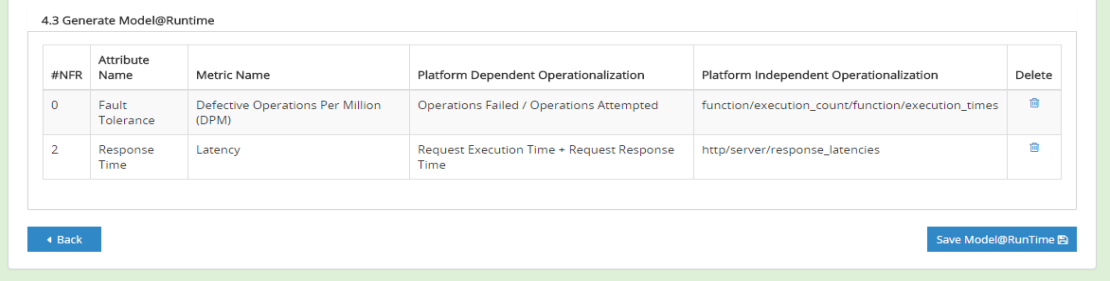


Figura 14 - Constructor de fórmulas: Resultado en el modelo en tiempo de ejecución.

Este proceso debe repetirse con todos los RNF que quieran ser monitorizados. Una vez todas las operacionalizaciones dependientes de la plataforma se hayan cargado en el modelo en tiempo de ejecución pulsando sobre *Save Model@Runtime*.

5.4.5. Generación del modelo en tiempo de ejecución

Una vez realizado el último paso, el modelo en tiempo de ejecución será mandado al monitor para inicializar la monitorización y una copia en XML podrá ser descargada. La estructura del XML generado será mediante las siguientes etiquetas:

- **CloudService:** Es la etiqueta encargada de abrir la información relacionada con la plataforma.
  - **Name:** Nombre de la aplicación que va a ser monitorizada.
  - **connectionString:** Cadena de texto para conectarse a la fuente de datos.
  - **deploymentId:** Identificador del servicios.
- **Characteristics:** Etiqueta encargada de abrir la sección de características a monitorizar.
  - **Name:** Nombre de la característica.
- **Atribute:** Etiqueta encargada de abrir la sección de atributos.
  - **Atibute;Name:** Nombre del atributo
- **Metric:** Etiqueta encargada de abrir la sección de métricas.
  - **Name:** Nombre de la métrica.
- **Operationalization:** Etiqueta encargada de abrir la sección de operalizaciones.
  - **Name:** Nombre de la operalizacion.
- **Function:** Etiqueta encargada de abrir la sección de las funciones.

    - **Formula:** Formula generada para el cálculo del cumplimiento del requisito dependiente de plataforma.
- **DirectMetric:** Etiqueta encargada de abrir la sección de las métricas.
    - **Name:** Nombre de la métrica.
    - **Identifier:** Numero único para identificar la métrica. Este será usado a la hora de crear la formula.
    - **extractionRate:** 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).
    - **extractionType:** Este es un valor numérico (en segundos) en el cuál se extrae un dato del contador seleccionado del servicio.3
- **NFR:** Etiqueta encargada de abrir la sección de los requisitos no funcionales.
    - **Name:** Nombre del NFR.
    - **NFRReference:** Identificador del requisito no funcional.

A continuación se muestra un ejemplo de la operacionalizacion representada en XML. Se puede ver un modelo en tiempo de ejecución generado en el caso de prueba realizado en el Anexo 2.

```
<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>([34]/[36])*1000000</formula>
                <operands>
                  <DirectMetric>
  <name>  function/execution_times </name>
                    <identifier>[36]</identifier>
                    <extractionRate>4</extractionRate>
                    <extractionType>0</extractionType>
                  </DirectMetric>

                  <DirectMetric>
                    <name>function/execution_count</name>
                    <identifier>[34]</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>
```

Figura 15 - Ejemplo de operacionalización representada en XML.

5.5. El Middleware de Monitorización

El Middleware de Monitorización es la herramienta software encargada de dar soporte a la segunda tarea del proceso de Monitorización. Esta herramienta es la encargada de recibir el modelo de calidad en tiempo de ejecución creado a partir del Configurador y realizar la monitorización del servicio. A partir del modelo en tiempo de ejecución, extrae los datos de monitorización utilizando métodos para recuperar los datos de la plataforma mediante llamadas a la API de la plataforma, almacena estos resultados, y genera un reporte en forma de gráficas accesibles para el usuario de los resultados obtenidos en la monitorización del servicio cloud. También es posible generar un informe de violaciones de las clausulas del SLA a partir de los datos de monitorización.

5.5.1. Descripción del funcionamiento

En esta sección se detallará el funcionamiento de una ejecución del Middleware de Monitorización.

Primero, el monitor se encuentra desplegado y ejecutándose en una instancia de la plataforma Google App Engine. Una vez lanzado a ejecución, se crea un objeto Monitor que se encargará en un primer momento de buscar un modelo de monitorización en tiempo de ejecución con el cual iniciar el proceso de monitorización. Una vez cargado este módelo, como salida del Configurador de la Monitorización, el *Monitor* lo mapeara como objeto y lo guardara sus atributos en la base de datos mediante la utilización del *DataManager*.

Con el modelo en tiempo de ejecución ya cargado, el Monitor creará una conexión estable con la API de monitorización de Google App Engine, introduciendo los datos de autentificación para que la API permita extraer datos de ella

Seguidamente el *QueueManager* se encargará de crear las tareas programadas para la recolección de los valores desde la API. Las tareas programadas llamarán al Monitor con las distintas operaciones y cálculos a realizar. Cuando la operación a hacer sea extraer datos de la aplicación desplegada en la nube, el monitor llamará al extractor con los datos necesarios el cual pasara a la Operationalization y hará los cálculos necesarios. El resultado de esto será el valor final de la métrica.

El monitor llamará al *DataManager* para guardar en base de datos el valor de la métrica obtenido. Al finalizar esta operación analizará este valor con el valor rescatado de la aplicación y si hay alguna anomalía creara un informe con esta.

Este proceso se repetirá de forma cíclica hasta que se actualice el modelo de monitorización en tiempo de ejecución o se cancele la monitorización.

En el caso de querer actualizar los datos de monitorización, cargará el nuevo modelo en tiempo de ejecución y mediante el *DataManager* actualizará los valores en la base de datos, sin parar la ejecución del middleware de monitorización.

6. Caso de estudio

En este capítulo se presentará un escenario como caso de estudio para ejemplificar el funcionamiento del configurador de la monitorización. En primer lugar se presentará el caso de estudio, a continuación se expondrá el contexto anterior para la creación del modelo de calidad en tiempo de ejecución, es decir, los pasos seguidos en el configurador de la monitorización y para finalizar explicaremos el funcionamiento del Middleware de Monitorización para la monitorización de ese servicio.

6.1. Presentación del caso

Una empresa de subastas Subastas y más S.L. ha decidido crear un servicio online de subastas en la línea y ha decidido desplegarlo en la nube. 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. Estos sitios necesitan contar con una serie de características de calidad, entre las cuales se contemplan altos niveles de disponibilidad, elasticidad, etc... Los servicios de subasta pueden tener distintos formatos, los más populares son las subastas directas e inversas.

Este servicio se encuentra desplegado en la plataforma Google App Engine en la forma de una aplicación web. Se ha decidido esta plataforma debido a su facilidad de despliegue, alta disponibilidad, integración con Eclipse y por la facilidad de precios que esta ofrece, la cual se basa en la cantidad de datos que recibe la aplicación.

La empresa de subastas considera críticas para el servicio la disponibilidad, la eficiencia y la latencia. Subastas y más S.L. está interesada en que los tiempos de respuesta del servicio sean los suficientemente bajos para que al cliente no le resulte desconfiable pujar por mercancías o servicios o poner en subasta sus productos. A su vez, también es vital que durante el proceso de subastas se generen el menor número de errores posibles para evitar frustraciones en los usuarios que pujen y también errores que puedan suponer menos pujas para la subasta. Por esto la Subastas y más S.L y el proveedor del servicio incluyen estas condiciones en el acuerdo de nivel de servicio.

Con mayor detalle en el acuerdo de nivel de servicio se acuerdan los requisitos no funcionales siguiente:

1) El tiempo de respuesta del servicio (*Response Time*) 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 fórmula:

$$Latency = Request\ Execution\ Time + Response\ Time$$

2) La confiabilidad (*Reliability*) será medida a través de las operaciones defectuosas por millón (*DPM*). En este caso, el servicio tendrá un máximo

de 10 operaciones defectuosas por millón (99.9999% de fiabilidad del servicio). Este requisito será calculado utilizando la métrica correspondiente (*Defective Operations per Million, DPM*).

$$DPM = \frac{Operations\ Failed}{Operations\ Attempted} * 10^6$$

Estos requisitos no funcionales serán monitorizados para velar por su cumplimiento. En el caso de que no se cumplan, el proveedor del servicio deberá reembolsar a Subastas & mas S.L. el coste del servicio y el coste de las pérdidas.

6.2. Configuración de la monitorización

6.2.1. Introducción de los datos de la plataforma.

En la primera interfaz, *Platform Selection* mostrada en la Figura 16, se seleccionará en el punto 1.1 la plataforma en la cual se ha desplegado el servicio, en el este caso Google App Engine. Una vez seleccionada aparecen dos campos más con los datos propios de los servicios de Google App Engine.

En el campo 1.2 se introduce el nombre del servicio, en nuestro caso el servicio es la aplicación web Subastas & mas S.L. 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 Subastas & mas S.L., 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.

Figura 16 - Caso de Estudio: Selección de Plataforma

6.2.2. Introducir el modelo de requisitos

El modelo de requisitos de monitorización se realiza fuera de los límites de la aplicación. Este documento contiene la información relativa a los requisitos no funcionales a monitorizar que será necesaria para establecer las métricas capaces de medir estos requisitos. El modelo de requisitos de monitorización para este ejemplo se puede ver en el Apéndice A.

En este paso el usuario deberá subir a la aplicación el acuerdo de nivel de servicio que contenga los requisitos no funcionales:

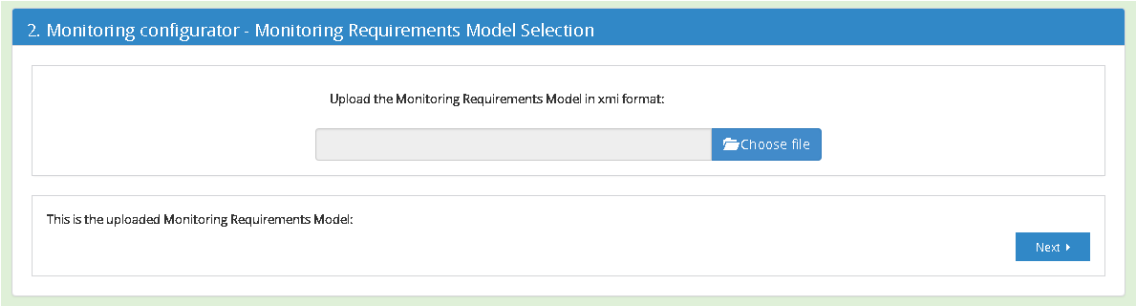


Figura 17 - Caso de Estudio: Subida del Modelo de Requisitos de Monitorización

Este archivo debe de estar en formato XMI. La aplicación transformará el documento XMI y guardara los datos de los requisitos no funcionales para posteriormente poder mapearlos con los atributos del modelo de calidad SaaS para poder monitorizarlos. En esta página se mostrará, una vez cargado el acuerdo de nivel de servicio, en forma de tabla. La importancia de este paso pasa por la transformación de un documento XMI en una serie de requisitos funcionales para poder monitorizar.

Una vez subido el fichero con éxito, se ilustrará el resultado final del modelo subido:

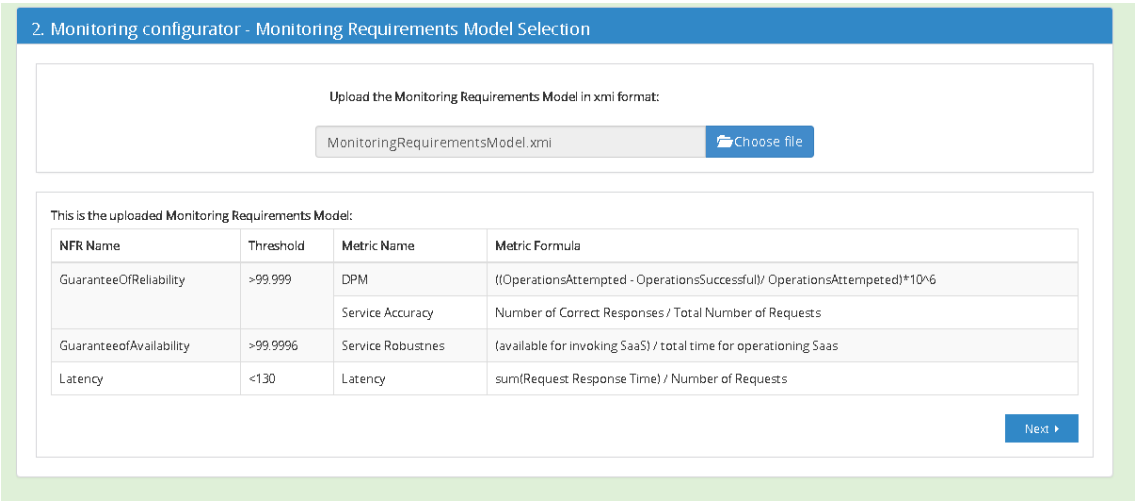


Figura 18 - Caso de Estudio: El modelo de especificación de requisitos.

Una vez subido este modelo de forma correcta, se activará el botón *Next* que se seleccionará a continuación para llegar hasta el siguiente paso.

6.2.3. Asociación de métricas

En este paso se deberá asociar las métricas del modelo de calidad SaaS con los requisitos no funcionales acordados en el acuerdo de nivel de servicio. En este caso utilizaremos la Latencia y la confiabilidad, con la métrica correspondiente del modelo de calidad SaaS (véase Apéndice A).

El usuario tendrá que escoger los requisitos no funcionales que quiere monitorizar de la aplicación o servicio desplegado en la nube. Cada uno de los requisitos escogidos deberá asociarlos con la correspondiente característica, subcaracterística, atributo, métrica y operacionalización para poder monitorizarlo. Al hacer la pertinente asociación este requisito no funcional se añadirá provisionalmente al modelo en tiempo de ejecución, aunque posteriormente se tendrán que crear las funciones necesarias para poder calcular los datos para la monitorización.

Tomando primero la Latencia. Se selecciona en la tabla dicho NFR y se procede a realizar la asociación. En este caso la clasificación se realizará de la siguiente manera: la característica a la que pertenece es Eficiencia de Funcionamiento (*Performance Eficiency*), la subcaracteristica será comportamiento temporal (*Time Behaviour*). El atributo será el tiempo de respuesta (*Response Time*), la métrica Latency y la operacionalizacion Latency.

Al terminar este paso pasaremos al último paso, el cual se trata de la construcción de fórmulas. En la siguiente imagen (Figura 19) se muestra cómo quedaría la configuración finalizada. Una vez configurado habrá que añadirla al modelo en tiempo de ejecución.

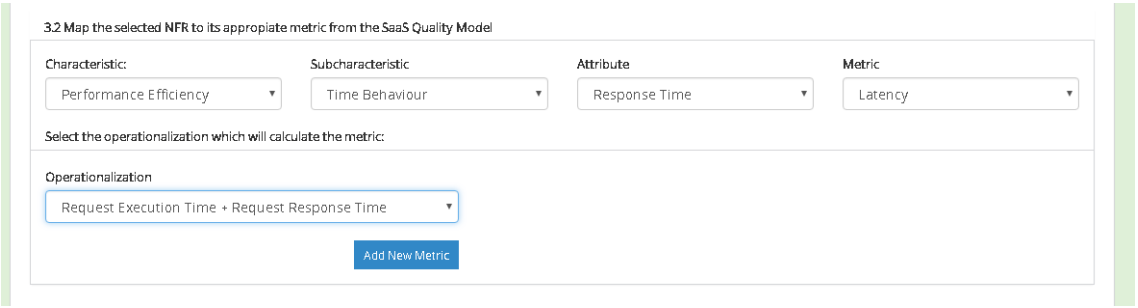


Figura 19 - Caso de Estudio: Clasificación de la latencia.

A continuación, se procede a realizar el mismo proceso con el RFN de confiabilidad. Al igual que en el paso anterior se seleccionará de la tabla dicho RFN y se procederá al mapeo de las características. En este caso, la característica a la que pertenece la confiabilidad es *reliability,* el atributo *fault tolerance* (tolerancia a fallos) y la operacionalización es ña correspondiente a DPM quedándose de la siguiente manera (Figura 20).Una vez configurado habrá que añadirla al modelo en tiempo de ejecución.

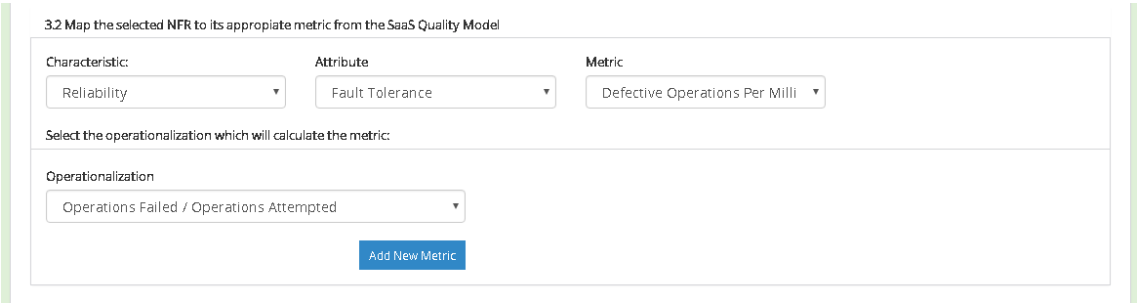


Figura 20 - Caso de Estudio: Clasificación de la confiabilidad.

En este momento, tendremos los RNF's clasificados con unos atributos determinados y unas métricas procedentes del modelo de calidad SaaS independientes de la plataforma. Una vez revisados que los datos son correctos y que ambos están en la tabla, se procederá a pulsar el botón *next* para continuar con el proceso.

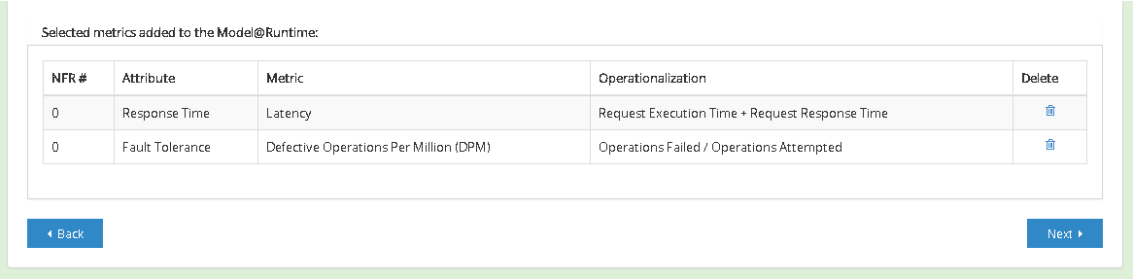


Figura 21 - Caso de Estudio: Clasificación de la confiabilidad.

6.2.4. Construcción de fórmulas

En este proceso ya vendrán definidas los atributos, métricas y operacionalizaciones que se requieren para la monitorización, pero estas definidas a alto nivel, es decir todas estas propiedades definidas serán independientes de plataforma. Como la aplicación o servicio a monitorizar se encuentra desplegada en una plataforma cloud concreta, es indispensable para el monitor tener las fórmulas para recuperar los datos con atributos dependientes de la plataforma.

Por lo tanto, en el siguiente paso habrá que transformar las operalizaciones independientes de la plataforma en dependientes de plataforma. Para ello, con la guia de las operacionalizaciones SaaS se deben componer las formulas con las que se trabajara para calcular las métricas de cada RNF.

En primer lugar se toma de la tabla la métrica de operaciones defectuosas y se realiza la fórmula, tomando los *Performance Counters* que se puedan asociar a cada uno de los elementos de la fórmula del espacio reservado a ellos en la aplicación. En este caso *Operations Attempted* pasa a ser el contador *Requests Total* mientras que *Operations Succesful* pasa a ser *Requests Succeeded*. Para añadir el contador Requests Total se busca en la lista y como parámetros se selecciona como tipo de extracción Total y el periodo de extracción 3 segundos. En el caso de *Requests Succeeded* también se toma este período y tipo de extracción, dado que ambos son contadores de funcionamiento similar. Puede observarse la construcción completa en la Figura 21. Para finalizar, habrá que añadir la formula y esta se queda almacenada.

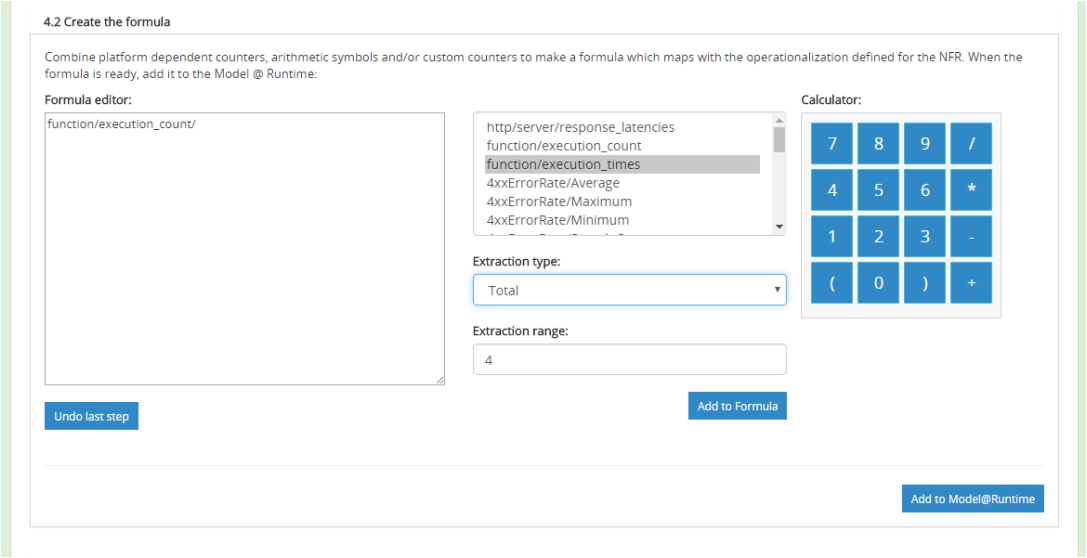


Figura 22 - Caso de Estudio: Construcción de la operacionalización de la confiabilidad.

Una vez añadida esta fórmula, se añade al modelo en tiempo de ejecución y se procede a añadir la siguiente métrica, la latencia. En este caso *Request Execution Time* (miembro de la fórmula independiente de la plataforma) tiene correspondencia directa con los parámetros de la plataforma en el nombre. Se procede a añadir este *Performance Counter* tal y como se vio en el caso anterior, solamente cambiando los parámetros. Esta vez se selecciona como tipo de extracción *Average*, dado que este contador representa el valor únicamente de la última petición medida, por lo que no es necesario obtener el último valor sino un valor medio. El tiempo de extracción en este caso será de 2 segundos dado que se necesitarán más muestras para conseguir el mayor espectro de resultados posibles. En la Figura 22 se muestra el resultado de esta configuración. También habrá que finalizar, añadiendo la formula.

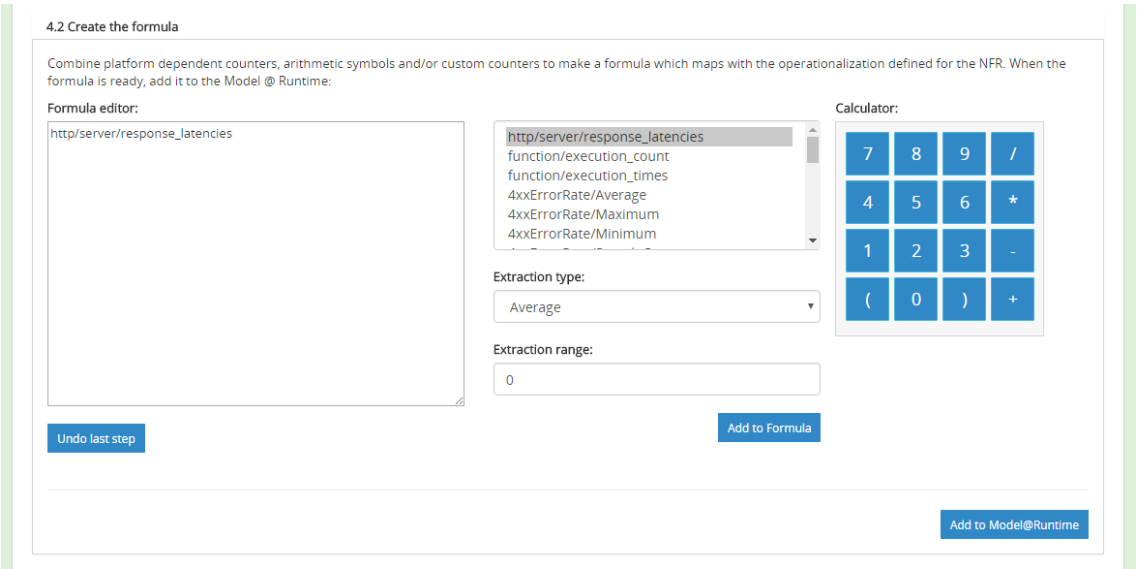


Figura 23 - Caso de Estudio: Construcción de la operacionalización de la latencia.

Una vez añadido esta última fórmula al modelo en tiempo de ejecución se procede a generarlo. Pudiendo descargar una copia en formato XML y mandando

al Middleware de Monitorización la información de monitorización presente en el modelo, así configurando el servicio y comenzando el proceso de extracción de datos.

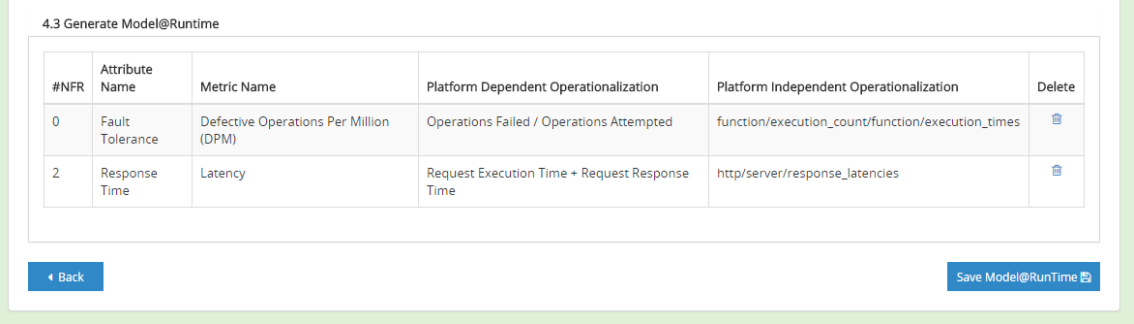


Figura 24 - Caso de Estudio: Resultado de la configuración.

6.2.5. Modelo en tiempo de ejecución XML

Una vez generado el modelo en tiempo de ejecución, esta se descargará para que el usuario pueda comprobar que los datos son correctos antes de mandarlos al monitor. El XML generado, contendrá todos los datos necesarios por el monitor para proceder a la monitorización. Las datos que el usuario ha ido seleccionando a lo largo de todo el proceso de configuración, serán los que aparezcan dentro de las etiquetas correspondientes (explicadas en el punto 5.4.4).

A continuación, se muestra una pequeña parte del modelo en tiempo de ejecución generado (ver figura 25). Además se puede consultar en el Anexo 2 el XML generado por el caso de estudio.

```
<?xml version="1.0"?>
<RuntimeModel xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <CloudService>
    <Name>Subastas [ mas S.L.</Name>
    <connectionString>DRIVER=MapRDrillODBC driver;AdvancedProperties={HandshakeTimeout=0;QueryTimeout=0;TimestampTZDisplayTimezone=utc;ExcludedSchemas=sysINFORMATION_SCHEMA;[
    OS1]};Catalog=DRILL;Schema=hivestg;ConnectionType=Direct;Port=8080</connectionString>
    <deploymentID>uU2DEmWWXX5a72oakyF1i2S2O</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>DPMK</name>
                <function>
                  <formula>([34]/[36])*1000000</formula>
                  <operands>
                    <DirectMetric>
    <name>  function/execution_times </name>
                      <identifier>[36]</identifier>
                      <extractionRate>4</extractionRate>
                      <extractionType>0</extractionType>
                    </DirectMetric>
                    <DirectMetric>
                      <name>function/execution_count</name>
                      <identifier>[34]</identifier>
                      <extractionRate>4</extractionRate>
                      <extractionType>0</extractionType>
                    </DirectMetric>
                  </operands>
                </function>
              </operationalization>
            </Metric>
          </metrics>
```

Figura 25 – XML generado por el configurador.

6.3. El Middleware de Monitorización

Una vez generado el modelo en tiempo de ejecución este se introducirá en el middleware de monitorización (Trabajo de Fin de Grado de Andrés Paez). Se

analizará el modelo en XML y se transformará la información recibida en instancias de la clase de Monitor Model@Run.Time y se almacenaran o se actualizarán los valores del modelo de monitorización en tiempo de ejecución en la base datos.

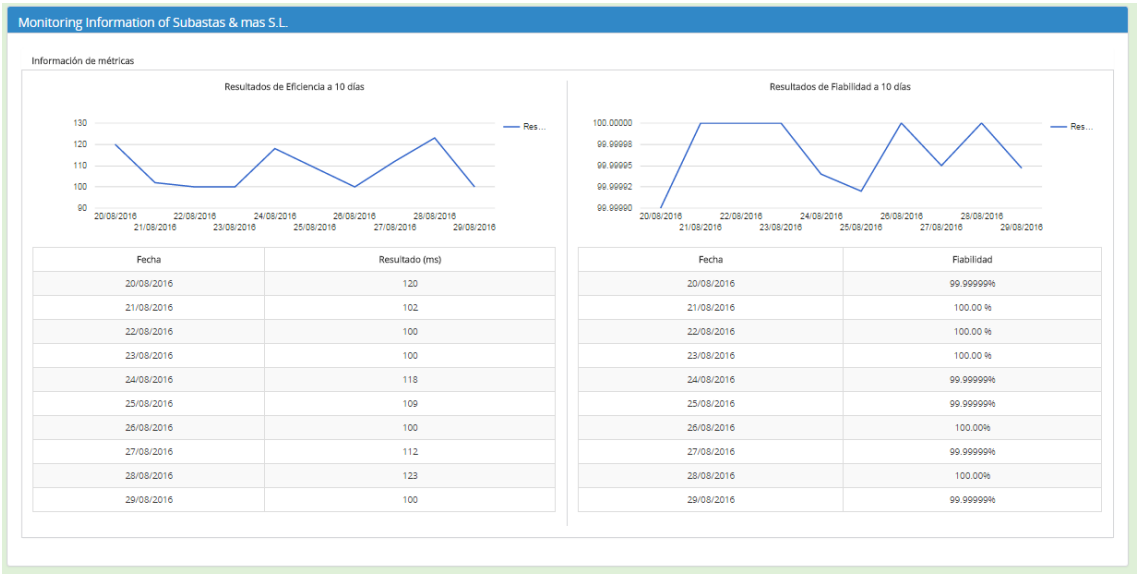
Una vez almacenado el modelo de monitorización en tiempo de ejecución, el middleware de monitorización analizará el modelo en tiempo de ejecución y creará una tarea programada para extraer de la API de monitorización de Google App Engine los datos de Operations Attempted, Opperations Failed y Request Execution Time.

Las tareas programadas se encargarán de recuperar en un proceso las métricas de Operations Attempted y Operations Failed y hacer el cálculo de la métrica, y en otro proceso recuperar la métrica de Request Execution Time. Una vez realizadas las operaciones anteriores, se almacenaran los cálculos de las métricas en base de datos y se analizarán los resultados. Para ello recuperara los datos del modelo de monitorización en tiempo de ejecución y se compararán los valores. En el caso de que los resultados se salgan de los límites establecidos, el monitor creara un reporte del fallo, creando una instancia de alerta en base de datos especificando, el tiempo y valor anómalo.

El middleware continuará recogiendo datos y calculando las métricas hasta que un nuevo modelo en tiempo de ejecución sea recibido y se actualicen los parámetros o hasta que se cancele el servicio de monitorización del servicio o aplicación desplegada en la nube.

Figura 26 - Caso de estudio: Introducción de datos de autentificación

Cuando el usuario entra en la interfaz de usuario del middleware de monitorización introducirá el nombre de la instancia de la monitorización (ver Figura 26) podrá ver un historial en forma de tabla de los datos recogidos en el proceso de monitorización y una gráfica de los resultados de esta (ver Figura 27).



Fecha	Resultado (ms)
20/08/2016	120
21/08/2016	102
22/08/2016	100
23/08/2016	100
24/08/2016	118
25/08/2016	109
26/08/2016	100
27/08/2016	112
28/08/2016	123
29/08/2016	100

Fecha	Fiabilidad
20/08/2016	99.99999%
21/08/2016	100.00 %
22/08/2016	100.00 %
23/08/2016	100.00 %
24/08/2016	99.99999%
25/08/2016	99.99999%
26/08/2016	100.00%
27/08/2016	99.99999%
28/08/2016	100.00%
29/08/2016	99.99999%

Figura 27 - Caso de estudio: Dashboard

7. Conclusiones y trabajos futuros

En este capítulo se recogen las conclusiones generales del presente trabajo, a continuación se presenta los trabajos futuros propuestos para continuar con este trabajo y para concluir los resultados obtenidos.

7.1. Conclusiones

En la actualidad, el Cloud Computing se ha convertido en la principal tendencia en el ámbito de las Tecnologías de la Información de los últimos años. En este 2015, todo indica que no sólo se mantendrá en un sitial relevante sino que llegará a transformarse en una opción imprescindible para los departamentos de TI, toda vez que será la plataforma esencial para acceder a servicios y soluciones tecnológicas que permitan a las organizaciones soportar el inmenso volumen de datos con el que convivirán cotidianamente en los próximos años.

Debido a estas tendencias, el uso de herramientas de monitorización de servicios cloud conllevaría un gran beneficio a los proveedores y consumidores de estos mismos. En el caso de los consumidores, para poder controlar el correcto uso del servicio contratado. Además, el usuario podrá comprobar que las características que el proveedor de servicio, estipuladas en el SLA se cumplen, teniendo que ser compensado económicamente en caso de incumplimiento. En el caso de los proveedores, para poder demostrar a sus clientes el cumplimiento de los servicios que ofrecen, pudiéndose posicionar por delante de sus competidores a la hora de luchar por la cuota del mercado.

Con la herramienta presentada se verifica la viabilidad del proceso de monitorización presentado por Cedillo et al [4] demostrando que este proceso es capaz de llevarse a cabo en un entorno de producción, enriqueciéndose del conocimiento a bajo nivel de este trabajo y refinándose para responder a las necesidades actuales de motorización de servicios cloud.

Como se ha explicado anteriormente, la mayoría de herramientas de monitorización actuales, son dependientes de la plataforma. Es decir, solo se podrán monitorizar las plataformas para las cuales se ha desarrollado. Por lo tanto, con la solución planteada, se podrá llegar a hacer una configuración para cualquiera de las plataformas existentes (en caso de mejorar la aplicación añadiendo estas)

7.2. Tecnologías

En esta sección se exponen algunos de los inconvenientes presentados durante la realización de este trabajo relacionados con las tecnologías utilizadas.

7.2.1. El Uso de Modelos en tiempo de ejecución

El uso de modelos en la ingeniería del software es una técnica muy útil para agilizar el tiempo de desarrollos de aplicación. En este proyecto se han empleado modelos en tiempo de ejecución como herramienta de establecimiento de los datos a analizar las cuales contienen el mapeo entre los requisitos no funcionales con las operacionalizaciones dependientes de plataforma necesarias para la monitorización. Esta tecnología supone varias ventajas a la hora de modelar los datos tales como la estructuración de la información, su carácter transformable, su

capacidad de estructurar expresiones complejas y la flexibilidad que aporta al middleware. En concreto estos modelos, permiten actualizar las fórmulas de métricas específicas en tiempo de ejecución sin la necesidad de alterar el proceso de monitorización.

La implementación de estos han traído consigo una serie de complicaciones, la mayor de ella es la de representar de forma precisa una realidad específica. Esto es debido a que no está pensado para que sea un elemento funcional sino un elemento más abstracto, lo cual hace que el desarrollador tenga complicaciones a la hora de definir este nivel más abstracto para modelos que dependan directamente de una plataforma o implementación específica.

Sin embargo, con el análisis continuo y las herramientas y tecnologías apropiadas estas inconvenientes desaparecen dejando solamente las virtudes de estas herramientas.

7.2.2. Mapeo de XMI

Este formato de documento ha ocasionado algunas dificultades a la hora de desarrollar las aplicaciones, necesitando invertir un alto número de horas de esfuerzo en el desarrollo de procesos que transformen estos modelos en objetos específicos de la implementación del configurado. Debido al poco uso de este modelo, el mapeo del XMI para generar los NFR con sus atributos ha tenido que hacerse mediante programación, pasando el XMI a formato JSON, uno de los más utilizados actualmente, y posteriormente generando las listas de los NFR con sus correspondientes atributos.

7.2.3. Google App Engine

Aunque la utilización de tecnologías tales como Java, HTML, Javascript, Spring, jQuery no eran desconocidas a la hora de realizar este trabajo, los frameworks y tecnologías específicas de la plataforma Google App Engine si lo eran. Esto significa que la implementación con estas tecnologías supuso un tiempo de investigación bastante alto y la implementación específica para esta plataforma un tiempo de implementación moderado.

Además, Google App Engine utiliza su propio plugin para la plataforma Eclipse, generando un proyecto que no coincidía con las tecnologías que se conocían. Para poder transformar el proyecto, se han tenido que dedicar muchas horas de investigación para poder suplir todos los errores que esto daban a la hora de realizar el cambio.

La falta de información específica encontrada en la red no ayudo a que esto sea más sencillo, la mayoría de información encontrada en la red era demasiado abstracta ya que era proporcionada por Google. Al ser esta tecnología bastante novel además hacia que encontrar soluciones a problemas específicos sea más complicado, ya que no existe mucha gente que haya realizado implementaciones parecidas anteriormente.

Otro problema que se encontró es a la hora de desplegar las aplicaciones. Esto fue debido a la versión de JAVA que Google App Engine te obliga a utilizar para poder desplegar una aplicación en la plataforma *cloud*. En este caso al tener el

ordenador al dio con la última versión, hubo que hacer un *downgrade* de la versión de JAVA.

7.3. Herramienta en pruebas

El configurador de la monitorización es la herramienta software implementadas para dar soporte al método de monitorización. Esta herramienta consistía en una aplicación web que permitiera la realización de un modelo en tiempo de ejecución.

Por una parte ello implicaba un conocimiento profundo de los modelos que intervenían en el proceso, el modelo de requisitos de monitorización, el modelo de calidad SaaS y el modelo de monitorización en tiempo de ejecución (todos ellos presentes en el Anexo). Estos modelos debían de ser comprendidos perfectamente para la elaboración de la aplicación, lo cual implicaba un estudio de la teoría referente a la calidad de software y la terminología de la medición. A su vez, al tratarse de modelos en fase de investigación se tuvieron que revisar y corregir en aquellas ocasiones en las que el modelo no bastase para representar la realidad de la configuración.

Al final, se ha conseguido llegar al presente prototipo a pesar de las trabas técnicas que se encontraron. Por ello, el presente prototipo todavía presenta margen de mejora en lo que usabilidad y diseño se refieren

7.4. Trabajos futuros

Este proyecto abre la posibilidad de diferentes trabajos para implementar en un futro tales como:

- **Mejora del configurador de Monitorización**: Como se ha dicho anteriormente, el configurador de monitorizacion aún tiene mucho margen de mejora. Lo que se pretende con la mejora propuesta es que, la aplicación aprenda del usuario experto como este crea las formulas con el fin que, se simplifique mucho la creación del modelo en tiempo de ejecución, hasta niveles que, cualquier usuario pueda crearlos sin la necesidad de tener grandes conocimientos.
- **Evaluación del configurador con usuarios finales**: Se pretende realizar un estudio de usabilidad con usuarios finales.
- **Extensión del configurador a otras plataformas**: Debido que el configurador está desarrollado con tecnologías estándar (Java, HTML, CSS..) este puede ser extendido a otras plataformas, añadiendo las APIS necesarias para gestionar la creación de formulas
- **Unión del configurador y el middleware**: Hacer que la aplicación del middleware y la del configurador de la monitorización sea una aplicación conjunta, por la cual no es necesario la descarga del modelo de monitorización en tiempo de ejecución del configurador y la posterior subida al middleware de monitorización de plataformas cloud. Las aplicaciones solo serían parte del proceso. A la finalización de la configuración la monitorización ya se queda configurada automáticamente.

8. Referencias

[1] Aceto, Giuseppe, et al. "*Cloud monitoring: A survey*." Computer Networks 57.9 (2013): 2093-2115. https://www.researchgate.net/profile/Alessio_Botta/publication/257582212_Cloud_monitoring_A_survey/links/551e59000cf29dcabb03b3bd.pdf

[2] Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM 53.4 (2010): 50-58.* *http://delivery.acm.org/10.1145/1730000/1721672/p50-armbrust.pdf?ip=90.74.200.161&id=1721672&acc=OPEN&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&CFID=659078997&CFTOKEN=26105083&__acm__=1471973554_7aa5121f613efa7fc717d04ede64ba0b*

[3] Blair, G. & Bencomo, N. & France, R. "Models@run.rime". *http://www.nellybencomo.me/publications/2009/ModelsatRunTime_EditorialOct09.pdf*

[4] Cedillo P., Jimenez-Gomez J., Abrahão S., Insfrán E. Towards a Monitoring Middleware for Cloud Services. 12th IEEE International Conference on Services Computing (IEEE SCC 2015), 27 June - 2 July, 2015, New York, USA.

[5] *Cedillo P., González-Huerta J., Abrahão S., Insfrán E. 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.*

[6] Cedillo, P., Gonzalez-Huerta, J., Abrahao, S., & Insfran, E. (2014). *"Towards Monitoring Cloud Services Using Models@ run. time. 9th Workshop on Models@run.time, 31–40".* http://st.inf.tu-dresden.de/MRT14/papers/mrt14_submission_5.pdf

[7] De Volder, Kris. "*JQuery: A generic code browser with a declarative configuration language."* International Symposium on Practical Aspects of Declarative Languages. Springer Berlin Heidelberg, 2006.

[8] Flanagan, D. (1996). "*JavaScript: The Definitive Guide*". ftp://ftp.micronet-rostov.ru/linux-support/books/programming/JavaScript/[O%60Reilly]%20-%20JavaScript.%20The%20Definitive%20Guide,%206th%20ed.%20-%20[Flanagan].pdf

[9] Hassan, Q. F., & Computers, F. (2011). "Demystifying Cloud Computing." Cross-Talk, *The Journal of Defense Software Engineering, 24(1), 16–21.* http://static1.1.sqspcdn.com/static/f/702523/10181434/1294788395300/201101-Hassan.pdf

[10] Jimenez, J. (2014). "Middleware para la monitorización de la calidad de servicios cloud". https://riunet.upv.es/bitstream/handle/10251/55548/JIM%C3%89NEZ%20-%20Middleware%20para%20la%20monitorizaci%C3%B3n%20de%20la%20calidad%20de%20servicios%20cloud.pdf?sequence=1

[11] Kaniz Fatema, Vincent C.Emeakaroha, Philip D.Healy, John P.Morrison, Theo Lynn, "A survey of Cloud monitoring tools: taxonomy, capabilities, and objectives" J. Parallel Distrib. Comput. 74 (2014) 2918–2933

[12] Mell, P. & Grance, T. (2011). "The NIST Definition of Cloud Computing" Recommendations of the National Institute of Standards and Technology. *NIST Special Publication 800 – 145*. http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

[13] Meng, S., & Liu, L. (2013). Enhanced monitoring-as-a-service for effective cloud management. IEEE Transactions on Computers, 62(9), 1705–1720. http://doi.org/10.1109/TC.2012.165

[14] Moldovan, D., Copil, G., Truong, H.-L., & Dustdar, S. (2013). MELA: Monitoring and Analyzing Elasticity of Cloud Services. 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, 80–87. http://doi.org/10.1109/CloudCom.2013.18

[15] Rouse, M. (2013). "Amazon Web Services (AWS)". http://whatis.techtarget.com/definition/Amazon-Web-Services-AWS

[16] V. Emeakaroha, T. Ferreto, M. Netto, I. Brandic, C. De Rose, CASViD: application level monitoring for SLA violation detection in clouds, in: Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual, 2012, pp. 499–508. http://dx.doi.org/10.1109/COMPSAC.2012.68.

[17] Amazon. Amazon Cloud Watch. 2014. https://aws.amazon.com/es/cloudwatch/ (accessed Agosto 2016).

[18] .Boundary. Boundary - Server and applications monitoring. 2015. http://www.boundary.com/ (accessed Agosto 2016).

[19] "HTML", Wikipedia, https://es.wikipedia.org/wiki/HTML

[20] Google. What is Google Cloud Monitoring? 2015. https://cloud.google.com/monitoring/docs (accessed Agosto 2016).

[21] Microsoft. *Collecting data with Microsoft Azure Diagnostics*. 2014. https://msdn.microsoft.com/en-us/library/azure/gg433048.aspx (accessed Agosto 2016).

[22] Monitis. *Network & IT Systems Monitoring- Monitis*. 2015. http://www.monitis.com/ (accessed Agosto 2016).

[23] Nimsoft. *Nimsoft*. 2015. https://support.nimsoft.com/ (accessed Agosto 2016).

[24] ISO/IEC 25010:2011-Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models, 2011

[25] Eclipse. https://www.eclipse.org/ (accessed Agosto 2016).

[26] Google monitoring API. https://cloud.google.com/monitoring/api/v3/ (accessed Agosto 2016).

Anexo A

1. Metamodelos de los modelos empleados por la infraestructura de Monitorización

1.1. Metamodelo del Modelo de Calidad SaaS (fuente: Cedillo et al [5])

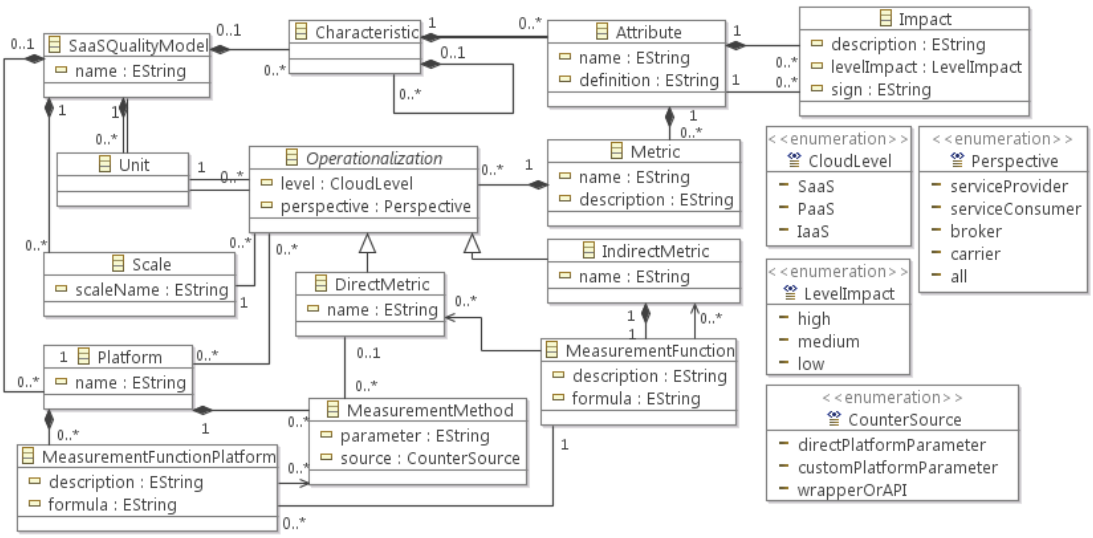


Figura 28. Anexo - Metamodelo del Modelo de Calidad SaaS

1.2. Metamodelo del Modelo de Requisitos de Monitorización (fuente: Cedillo et al [5])

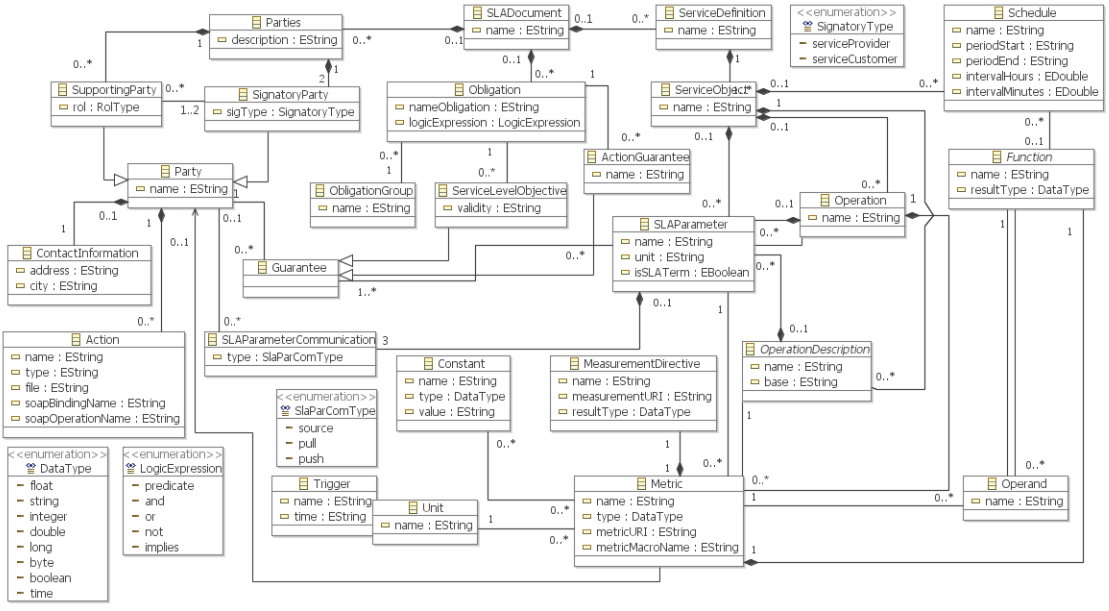


Figura 29. Anexo - Metamodelo de Reauisitos de Monitorización

1.3. Metamodelo del Modelo Monitorización en tiempo de ejecución (fuente: Cedillo et al [5])

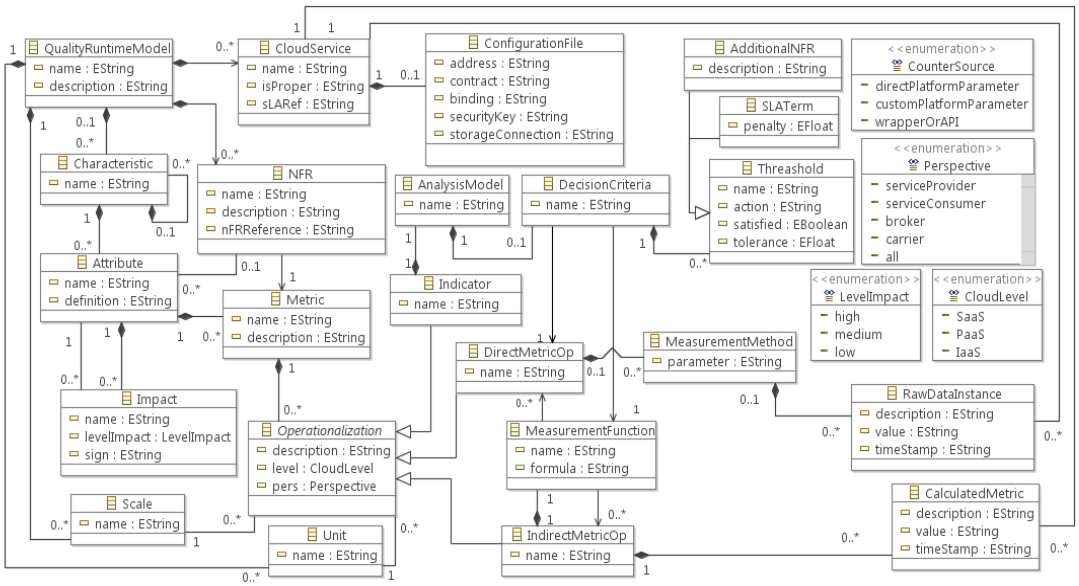


Figura 30. Anexo - Metamodelo del Modelo de Monitorización en tiempo de ejecución

2. Modelos empleados en el caso de estudio

2.1. Modelo de Requisitos de Monitorización

Nombre	Métrica	Formula	Umbral	SLA Specification
Eficiencia	Latencia	$Latencia = Tiempo\ de\ Ejecución\ de\ Peti + Tiempo\ de\ Respuesta$	<130ms	El tiempo de respuesta no puede ser superior a 130 ms
Confiabilidad	DPM	$DPM = \frac{Operaciones\ Fallidas}{Nº\ Total\ de\ Operaciones}$	<0.01	El número de operaciones defectuases por millón de operaciones es máximo 1

Tabla 3. Anexos - Modelo de Requisitos de Monitorización

2.2. Modelo de Calidad SaaS

Características	Sub-Características	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	(Operations Attempted – Operations Successful)/

			Per Million (DPM)	Operations Attempted
				Operations Failed / Operations Attempted
				Operations Failed / (Opeations Successful + Operations Failed)
		Service Stability	Coverage of Fault Tolerande (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
Performance Efficiency	Time Behaviour	Response Time	Latency	Request Execution Time + Request Response Time
			Request Execution Time	Execution Time
			Time Behaviour (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

Tabla 4. Anexo - Modelo de calidad SaaS

2.3. Modelo en tiempo de ejecución 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>Subastas & mas S.L.</Name>
    <connectionString>DRIVER=MapRDrillODBC
driver;AdvancedProperties={HandshakeTimeout=0;QueryTimeout=0;Timest
ampTZDisplayTimezone=utc;ExcludedSchemas=sysINFORMATION_SCHEMA;[OS1
]};Catalog=DRILL;Schema=hivestg;ConnectionType=Direct;Port=8080</co
nnectionString>
    <deploymentID>uU2DEmWWxX5a72oakyF1i2S2O</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>([34]/[36])*1000000</formula>
                <operands>
                  <DirectMetric>
                    <name>  function/execution_times </name>
                    <identifier>[36]</identifier>
                    <extractionRate>4</extractionRate>
                    <extractionType>0</extractionType>
                  </DirectMetric>

                  <DirectMetric>
                    <name>function/execution_count</name>
                    <identifier>[34]</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> http/server/response_latencies </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>
```