ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"SIMULADOR PARA MINIMIZAR LA VIOLACIÓN DE ACUERDOS DE NIVELES DE SERVICIO (SLAs) EN ARQUITECTURAS DE NUBES BASADAS EN MECANISMOS DE MERCADO"

INFORME DE PROYECTO DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES
ORIENTACIÓN SISTEMAS MULTIMEDIA

Presentado por:

Ericka Esther Castro Montoya

José Luis Sumba Zhongor

GUAYAQUIL - ECUADOR

2015

AGRADECIMIENTO

A mis padres, por dármelo todo y

por ser mi mayor bendición.

A mi tía Sara, por su apoyo constante

durante toda mi vida.

A la PhD. Carmen Vaca, por ser más que

una profesora y ayudarnos con sus consejos.

A mi compañero de tesis, por su paciencia

y empeño durante esta etapa.

A mi amigo Juan León Mera, por ser como

mi hermano y escucharme en todo momento.

A nuestros profesores, por todos los

conocimientos y consejos transmitidos.

Ericka Castro M.

A mis padres, por su sacrificio

y apoyo incondicional.

A mi compañera de tesis, Ericka,

por su dedicación y

esfuerzo en esta etapa.

A Paola y Dennisse , por escucharme

y brindarme siempre su apoyo.

A la PhD. Carmen Vaca, por apoyarnos

con sus consejos.

A nuestros profesores, por los

conocimientos impartidos.

José Luis Sumba.

DEDICATORIA

A mi mamá, por ser la luz de mi camino.

A mi padre, por enseñarme que la honestidad

y el sacrificio me hacen un mejor ser humano.

A mi mami Cloti, por ser mi segunda mamá.

A mis hermanos, por ser mi fortaleza y

mis ganas de salir adelante.

A mis amigos, por enseñarme que el

camino es más fácil

cuando se tiene la compañia adecuada.

A todo aquel que sueña y lucha por

cumplir lo que se propone.

Ericka Castro M.

A mi madre, por enseñarme a

vencer las adversidades.

A mi padre, por ser ejemplo de dedicación

y entrega total a la familia.

A mis hermanas, por soportarme, por ser

mi inspiración y motivo de superación.

A mis amigos, por su constante apoyo.

A Logan, que más que una mascota

fue un amigo y motivo de alegría

para toda la familia

José Luis Sumba.

TRIBUNAL DE SUSTENTACIÓN

Ph.D. Sixto García

PRESIDENTE

MSc. Carlos J. Mera G.

DIRECTOR DE PROYECTO DE GRADUACIÓN

Ph.D. Carlos Monsalve A.

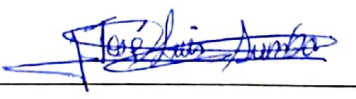
MIEMBRO PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este informe, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Ericka Esther Castro Montoya



José Luis Sumba Zhongor

RESUMEN

El presente proyecto consiste en diseñar un modelo que permita minimizar las violaciones a un SLA (Service Level Agreement por sus siglas inglés) dentro de una arquitectura en Nube basada en un mecanismo de mercado. Se denomina violación cuando el proveedor no cumple con los requerimientos acordados con el usuario dentro del SLA.

Se realizará un análisis de mecanismos de mercado y sistemas de reputación existentes. Dentro de la etapa de diseño del modelo, el uso del mecanismo de mercado y el sistema de reputación ofrecerá al usuario la oportunidad de seleccionar a un proveedor, no sólo considerando el precio del servicio, sino también la reputación del proveedor, con lo cual se espera que los servicios sean alojados por proveedores con una reputación alta y cuya probabilidad de falla sea baja.

La etapa de experimentación del modelo se realizará en un simulador implementado sobre el framework de CloudSim. Los datos para la simulación al igual que los resultados serán manejados en formato XML. Además, se presentarán los resultados gráficamente con la librería JFree-Chart para facilitar la interpretación de los mismos.

Finalmente, el uso del lenguaje estadístico R permitirá procesar los resultados y generar gráficas de las violaciones ocurridas en la simulación para los mecanismos de mercado y sistemas de reputación previamente seleccionados.

Índice General

Índice de figuras

Índice de Tablas

LISTA DE ACRÓNIMOS

DOM . . . . . *Document Object Model*

FCFS . . . . . *First-Come-First-Serve*

IaaS . . . . . *Infrastructure as a Service*

LSP . . . . . *Logic Scoring of Preferences*

MBA . . . . . *Market-Based Resource Allocation*

MBC . . . . . *Market-Based Control*

NIST . . . . . *National Institute of Standards and Technology*

PaaS . . . . . *Plataform as a Service*

QoS . . . . . *Quality of Service*

SaaS . . . . . *Software as a Service*

SLA . . . . . *Service Level Agreement*

VM . . . . . *Virtual Machine*

INTRODUCCIÓN

En la Nube existen distintos modelos de servicios: software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS)[23]. Los proveedores cobran por dichos servicios de acuerdo al uso y ofrecen la posibilidad de escalar dinámicamente según la demanda. Ellos deben garantizar que los servicios estén disponibles y cumplan con los requerimientos del usuario establecidos en el contrato denominado SLA. Para esto, requerimientos funcionales y no funcionales son definidos en dicho contrato. Adicionalmente, en él se establecen las obligaciones tanto del usuario como del proveedor y las sanciones en caso de incumplimiento. El incumplimiento del contrato es conocido como una violación al SLA.

Se analizarán algunos de los mecanismos de mercado existentes, los cuales se basan en el precio para la asignación de recursos y permiten definir políticas conocidas como Market-Based Resource Allocation, y establecer mecanismos de control conocidos como Market-Based Control (MBC)[8]. De los mecanismos a ser analizados, se seleccionarán dos: subasta de oferta publicada(Posted Offer) y subasta inversa(Reverse Auction). Implementando este sistema en la Nube se garantizaría que el proveedor con la mejor oferta y el precio más bajo para el usuario, sea el

que aloje el servicio, pero esto no garantizaría que se eviten violaciones al SLA. Por lo tanto, con el propósito de minimizar tales violaciones se hará uso de indicadores de desempeño y confiabilidad del proveedor, los cuales se denominan reputación. La reputación puede ser definida por métricas que se basan en el historial y en la calificación que los usuarios dieron a los servicios alojados. Del mismo modo, serán analizados algunos sistemas de reputación existentes y se escogerán dos para ser utilizados en nuestro proyecto: Blurred Beta Reputation y Beta Based Reputation. Adicionalmente, propondremos una nueva métrica: Average Combinated Reputation. Ésta reputación la hemos denominado Metric Based Reputation porque está basada en métricas. Por otro lado, la reputación también puede ser definida por el cumplimiento de los parámetros QoS (Quality of Service) establecidos en el SLA. Dichos parámetros son indicadores de calidad y para el presente proyecto se utilizarán cuatro: Availiability, Reliability, Performance y Cost. Con la finalidad de obtener una única reputación resultado de la combinación de dichos QoS se empleará un método conocido como Logic Scoring of Preferences(LSP) descrito en [27], el cual permitirá unificar dichos criterios y hallar un valor único. A ésta reputación la hemos denominado QoS Based Reputation.

En nuestros días, realizar estos experimentos en Nubes reales requiere tiempos extensos y resulta costoso, por tales motivos hemos optado por implementar un simulador sobre el framework de CloudSim que cuenta con componentes para modelar y simular la Nube de una manera relativamente sencilla y de bajo costo.

Durante el proceso de simulación los datos de usuarios, proveedores, servicios a alojarse en la Nube y resultados serán manejados utilizando archivos XML. En el simulador se permitirá importar y exportar los datos en dicho formato. Además, los resultados obtenidos serán representados gráficamente a fin de facilitar su interpretación. Posteriormente, dichos resultados serán procesados con el lenguaje R para generar las tablas y curvas de las violaciones ocurridas en simulación con el fin de obtener evidencias con respecto a dichas violaciones a lo largo de los diferentes experimentos.

CAPÍTULO 1

1. Antecedentes

1.1 Descripción del problema

La computación en la Nube[1] (Cloud Computing) es un modelo que provee infraestructura y aplicaciones como un servicio escalable de acuerdo a la demanda y que se encuentra basado en un paradigma de negocio de pago por uso. En dicho entorno los proveedores deben garantizar que los servicios que ofrecen estén disponibles al ser solicitados por los usuarios. Además, estos deben cumplir con requerimientos acordados entre el proveedor y el usuario. Dicho acuerdo es definido en un SLA, mismo que considera las obligaciones y sanciones en caso de incumplimiento a los requerimientos no funcionales o parámetros de calidad, conocidos

[1] Entiéndase como Nube al entorno que en inglés se denomina Cloud.

como parámetros QoS (Quality of Service por sus siglas en inglés), tales incumplimientos se conocen como violaciones a un SLA; éstas pueden presentarse debido a diversos factores, uno de ellos es la carga de trabajo que deben soportar y los cortes imprevistos que pueden ser ocasionados por software, hardware o fallas de red [11][13]. Sin duda alguna, minimizar las violaciones al SLA con la finalidad de garantizar el cumplimiento de los parámetros QoS ocupa un lugar clave dentro de la relación cliente-proveedor. En consecuencia, el objetivo del presente trabajo es experimentar con un modelo que permita disminuir las violaciones a un SLA dentro de una arquitectura en Nube basada en un mecanismo de mercado.

1.2 Objetivos

1.2.1 General

Experimentar con un modelo que permita disminuir la violación de un SLA dentro de una arquitectura en Nube basada en un mecanismo de mercado.

1.2.2 Específicos

1. Analizar algunos modelos de sistemas de reputación existentes y diseñar un modelo, que permita hacer más precisa la asignación de reputación a provee-

dores.

2. Implementar un simulador extendiendo el framework de CloudSim.

3. Establecer escenarios de simulación y métricas que permitan comparar los resultados esperados con los resultados del experimento.

4. Representar gráficamente los resultados obtenidos de manera que se facilite su interpretación.

1.3 Justificaciones

Los sistemas de mercado utilizan mecanismos basados en precio para asignación de servicios, mientras que los sistemas de reputación manejan un historial de evaluaciones de servicios prestados que refleja el desempeño de un proveedor de servicio, permitiendo establecer indicadores para usuarios futuros.

En nuestros experimentos, se va a combinar estos dos sistemas para diseñar un modelo que se ajuste de manera más eficiente a las necesidades del usuario de la Nube y a los servicios ofrecidos por los proveedores considerando el precio y la reputación. Debido a que la implementación en una plataforma real es de costo elevado y de tiempo de desarrollo prolongado, se ha optado por el uso de un simulador que permita reflejar los resultados de dichos experimentos. El uso del

framework de CloudSim permitirá modelar la Nube y replicar los experimentos en un entorno fácil de controlar.

1.4 Alcances y limitaciones

Se analizarán los resultados obtenidos durante las simulaciones para identificar si los resultados esperados concuerdan con los resultados obtenidos en simulación y verificar si el modelo minimiza las violaciones del SLA. Caso contrario, no se aceptará la hipótesis y se planteará recomendaciones para trabajos futuros.

El análisis de resultados estará enfocado en los sistemas de reputación seleccionados, más no en los mecanismos de mercado utilizados.

Las características de los proveedores y usuarios serán constantes durante la simulación. La capacidad del proveedor no aumentará (dinámicamente) conforme se incremente la demanda de servicios.

1.5 Contribuciones

En estudios a futuro, podría considerarse la experimentación en plataformas de Nube reales, que aunque costoso, sería un estudio que permitiría que los resultados obtenidos sean más precisos. En trabajos posteriores podrían diseñarse modelos

basados en otros mecanismos de mercado y tomar en consideración otros sistemas de reputación.

CAPÍTULO 2

2. Fundamentos Teóricos

2.1 Computación en la Nube

2.1.1 Definición

En la actualidad la computación en la Nube es un modelo que provee infraestructura y aplicaciones como un servicio que puede ser escalable de acuerdo a la demanda y que se encuentra basado en un paradigma de negocio de pago por uso [23].

Los servicios ofrecidos varían desde aplicaciones de ofimática en el caso de Google y Microsoft, hasta proveedores como E-bay y Amazon que utilizan la Nube para

venta y subasta de productos [35]. Dichas compañías (Amazon, Google e IBM) han invertido e innovado en la computación en la Nube llegando a ser expertos en el alojamiento en sus centros de datos, pero explotando la idea de la Nube han logrado una mayor escalabilidad de manera que sus centros de datos puedan ser usados por terceros [37]. De acuerdo con [26] la computación en la Nube ofrece las siguientes características:

- **Alta escalabilidad:** el entorno y la infraestructura permiten que los servicios se adapten a cambios requeridos por los usuarios.

- **Agilidad:** se pueden compartir los recursos entre las distintas tareas a realizarse, permitiendo una reducción y mejora en los tiempos de respuesta.

- **Alta disponibilidad y confiabilidad:** la disponibilidad de los servicios alojados en la Nube es alta con lo cual la probabilidad de que ocurra un fallo en la infraestructura es mínimo, lo que garantiza la confiabilidad de la Nube.

- **Multisharing:** debido a que el trabajo en la Nube se realiza de manera distribuida y compartiendo recursos, estos son accesibles para múltiples usuarios y aplicaciones, los que pueden desenvolverse de una manera más efectiva reduciendo costos a través del uso de infraestructura compartida.

- **Servicios de pago por uso:** los servicios que ofrece la Nube a sus usuarios

son facturados haciendo uso de un modelo de pago por uso. En el SLA el usuario y el proveedor acuerdan el pago de los servicios basado en la los requerimientos del usuario.

Además, en Nubes que ofrecen servicios de infraestructura o de plataforma con la finalidad de que los servicios provistos sean más accesibles para los usuarios, la Nube debería proveer APIs (Application Programming Interfaces) para facilitar el acceso a sus servicios.

2.1.2 Clasificación

Existen varios criterios para clasificar una Nube. Una de las clasificaciones es la que realiza NIST (National Institute of Standards and Technology) en la que se identifica a la Nube de acuerdo a tres modelos de servicios distintos que ésta provee: software como servicio, plataforma como servicio e infraestructura como servicio[23].

Software como un servicio (SaaS): es un modelo que permite que aplicaciones alojadas por el proveedor sean utilizadas directamente por los usuarios finales, sin que sea necesaria la instalación de las mismas en sus computadores, ofreciéndoles un servicio a muy bajo costo que permite reducir gastos de infraestructura para el consumidor. El usuario no administra ni controla la infra-

estructura en la que está alojada la aplicación. El soporte y mantenimiento de servidores, almacenamiento, sistema operativo y funcionalidad relacionados con la aplicación/servicio/software son realizados por el proveedor.

Plataforma como un Servicio (PaaS): es un modelo que permite al usuario acceder a un ambiente de desarrollo y despliegue de aplicaciones. El proveedor da soporte para el uso de varias herramientas y lenguajes de programación. El usuario no administra la infraestructura en la que se encuentra alojada la aplicación, pero tiene control sobre las aplicaciones que han implementado y sobre las configuraciones necesarias en la infraestructura para el funcionamiento y uso del servicio.

Infraestructura como un Servicio (IaaS): es un modelo que permite al usuario acceder a la infraestructura, al ambiente de desarrollo y realizar el despliegue de las aplicaciones. Los recursos de hardware son virtualizados, pero el consumidor tiene control sobre recursos como: procesamiento, almacenamiento, sistemas operativos y conectividad relacionados con la aplicación. Ver figura 2.1.

Existe otra clasificación para la Nube en [26] en la que se indica que esta puede ser realizada de acuerdo al entorno de despliegue para los servicios, con lo cual tendríamos los siguientes cuatro tipos de Nube: Nubes públicas, Nubes privadas , Nubes híbridas y Nubes externas.

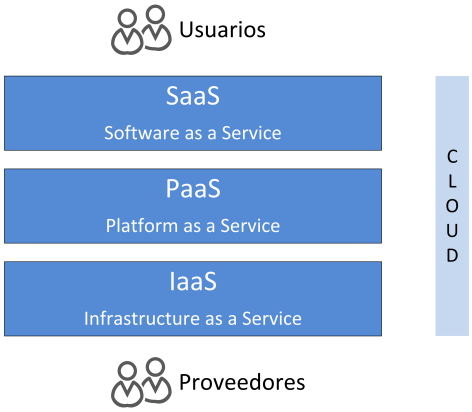


Figura 2.1: Servicios de la Nube

- **Nube pública:** utilizada para proveer servicios a organizaciones corporativas, personas naturales o jurídicas. Los servicios alojados en Nubes públicas suelen ser vendidas por un agente (Broker).

- **Nube privada o interna:** en una Nube privada o interna la infraestructura de la Nube es dedicada a una organización específica.

- **Nube híbrida:** este tipo de Nubes, es una combinación entre conceptos relacionados a una Nube privada y conceptos de una Nube pública.

- **Nube externa:** infraestructura que organizaciones proveen a otras organizaciones, no es una Nube pública. Como su nombre lo indica es una Nube utilizada por una organización pero se encuentra fuera de sus instalaciones.

Luego de haber analizado las clasificaciones principales de una Nube, a continuación en la figura 2.2 presentamos una referencia sobre la arquitectura de una Nube

definida por **National Institute of Standards and Technology (NIST)** que identifica los actores principales, sus actividades y sus funciones. El diagrama muestra una arquitectura genérica de alto nivel y tiene por objeto facilitar la comprensión de los requisitos, usos, características y estándares de la computación en la Nube.

En la figura 2.2 se introduce los principales elementos que forman parte del Modelo Conceptual de una Nube. Podemos identificar que en la Nube interactúan cinco actores principales: consumidor, proveedor, auditor, transportista y broker. A continuación describiremos cada uno de ellos:

- **Consumidor:** en el entorno de la Nube se denomina consumidor a la organización o persona que utiliza los servicios de la Nube. Con el fin de que los requerimientos técnicos y requerimientos no funcionales, niveles de calidad, sean cumplidos por parte del proveedor se establece un acuerdo de servicio SLA. En el SLA también se consideran las obligaciones y sanciones en caso de un incumplimiento a los requerimientos, lo que se denomina una violación de un SLA.

- **Proveedor:** el proveedor es aquella organización que maneja la infraestructura para que los distintos servicios: IaaS, PaaS y SaaS sean accesibles para los demás actores de la Nube. El proveedor define en el SLA las limitaciones del servicio y condiciones que debe cumplir el consumidor. Cabe aclarar que si bien es cierto el proveedor maneja la infraestructura de los servicios no

necesariamente es el dueño de la misma.

- **Auditor:** un auditor en la Nube es una entidad encargada de realizar un análisis independiente de los controles de servicios en la Nube con la finalidad de emitir una opinión sobre la misma. Estos análisis se conocen como auditorías, y se llevan a cabo para verificar la conformidad con las normas a través de la revisión de evidencia objetiva. Los servicios proporcionados por el proveedor que pueden ser evaluados son: controles de seguridad, impacto sobre la privacidad y el rendimiento.

- **Broker:** en ocasiones, la integración de los servicios puede volverse compleja para el consumidor, entonces podría darse el caso de que el consumidor decida contratar los servicios de un broker en lugar de contratar directamente al proveedor. Es decir que en la Nube se define a un broker como aquel que actúa como un negociador entre el consumidor y proveedor ofreciendo los servicios del proveedor con un valor agregado como reportes de rendimiento, mayor seguridad, etc. o simplemente como un intermediario.

- **Transportista:** el agente que actúa como intermediario y provee la conectividad para que los servicios de la Nube sean accesibles para los consumidores. Dicho acceso puede darse a través de dispositivos de red y telecomunicaciones.

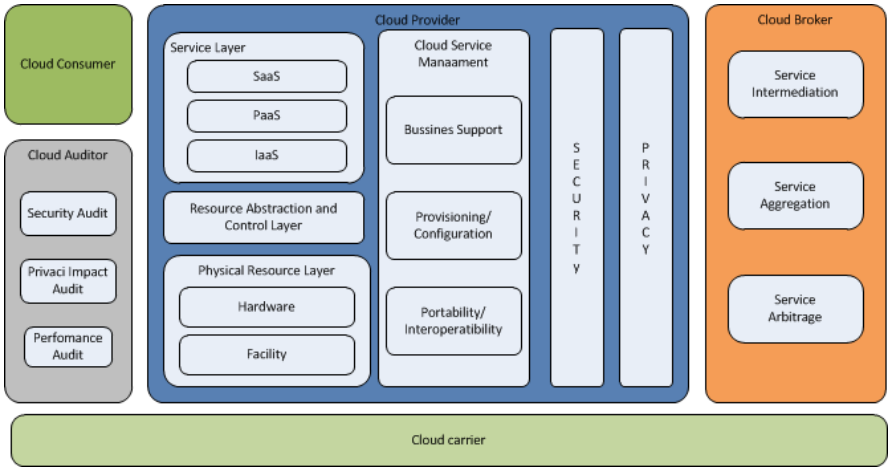


Figura 2.2: Modelo Conceptual
Fuente: National Institute of Standards and Technology [23]

2.2 Nubes basadas en mecanismos de Mercado

Según [8] el mercado puede ser definido como un conjunto de mecanismos mediante los cuales los compradores y vendedores realizan negociaciones para el intercambio de bienes o servicios. La cantidad de mercancía (bien o servicio) que los compradores están dispuestos a adquirir se conoce como demanda. Por parte del proveedor, la cantidad de mercancía que está dispuesto a vender se conoce como oferta.

Adicionalmente, una de las principales características del mercado es que los agentes: comprador y vendedor, actúan de manera independiente sin la supervisión de un controlador centralizado. Un sistema que utiliza el concepto de mercado o sus características puede ser definido como un sistema basado en mercado [43].

La computación en la Nube es un ejemplo representativo de un sistema basado en mercado. Las investigaciones realizadas se enfocan en combinar principios de economía para encontrar soluciones a problemas que ocurran en este entorno. También se busca mejorar la asignación de recursos mediante la definición de políticas conocidas como **Market-Based Resource Allocation(MBA)**, y establecer mecanismos de control conocidos como **Market-Based Control (MBC)**[8].

Por otra parte, mecanismos económicos como subastas y fijación de precios diferenciados han sido utilizados para establecer reglas en el proceso de asignación de recursos. En [11] se detalla un estudio de la aplicación de mecanismos de mercado en sistemas distribuidos. Para [2], [33] y [36] el uso de principios de mercado puede contribuir en mejorar la asignación de recursos en la Nube. En [12] y en [13] se emplean subastas para prevenir la violación de SLA definidas entre los usuarios y el proveedor de servicios.

Un sistema de mercado está compuesto por los siguientes elementos:

- **Operadores (Traders):** son los elementos encargados de cotizar los precios y realizar el proceso de regateo.

- **Marketplaces:** son los lugares donde se reúnen los componentes, compradores y vendedores para llegar a un acuerdo.

Los sistemas basados en mercado implementan agentes compradores que necesitan hacer un trabajo para obtener dinero y comprar recursos a los vendedores. También, implementan vendedores los cuales tienen recursos y compiten para venderlos. Este proceso de compra y venta está regido por dos reglas:

- Si la demanda excede la oferta, el precio podría elevarse
- Si la oferta excede la demanda, el precio podría disminuir

Estamos familiarizados con la idea de que el precio está en función de la demanda o la oferta [8]. La figura 2.3 muestra la relación entre oferta y la demanda. Además, de la forma en la que esto incide en el precio que los compradores están dispuestos a pagar y el precio que los vendedores están dispuestos a cobrar. Como podemos observar, a medida que la demanda excede a la oferta, el precio aumenta y si la oferta excede a la demanda entonces el precio disminuye.

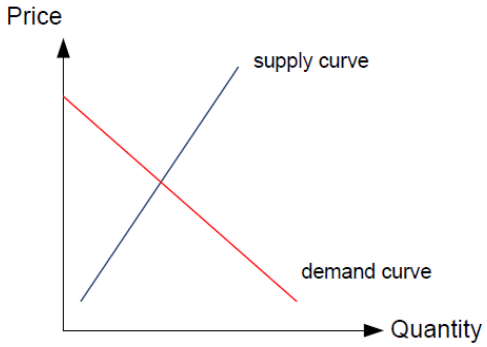


Figura 2.3: Oferta(supply curve) vs Demanda(demand curve)
Fuente: Minimal intelligence agents for bargaining behaviors in market based environments [8]

Buyya et al.[4] define una Nube como un sistema paralelo y distribuido compuesto de ordenadores interconectados y virtualizados considerados como recursos unificados basados en acuerdos de nivel de servicio. Esta definición brinda una perspectiva orientada hacia el mercado de la Nube y según lo revisado anteriormente con respecto a los sistemas basados en mercado, permite identificar los siguientes elementos:

* Usuarios
* Proveedores
* Brokers
* Auditores
* Transportistas

2.3 Sistemas de Reputación

2.3.1 Definición de Reputación

Con la finalidad de introducir los sistemas de reputación, primero analizaremos qué se entiende por reputación. Para el diccionario de la Real Academia de la Lengua

Española: La **reputación** es el “prestigio o estima que son tenidos a alguien o algo". De acuerdo con [29] dicha estima es necesariamente construída y actualizada a lo largo del tiempo con la ayuda de diferentes fuentes de información. Las interacciones directas y la información proveniente de testigos son las fuentes principales para construir una reputación.

Actualmente, la reputación es un objetivo de investigación multidisciplinario. Existen infinidad de definiciones para los términos reputación y confianza; para Mui L. “la reputación se refiere a la percepción que un agente tiene de las intenciones y normas de otro agente"[24]. Mientras que para Schlosser et al. la reputación “es una recopilación de información procesada sobre el rendimiento de un agente a través de las experiencias de terceros que han mantenido una transacción con él"[30].

Por otra parte, el término reputación viene ligado a un término que denominamos confianza, Mahoney G. asegura que “la confianza es una medida de la voluntad de proceder con una acción (decisión) que sitúa a las partes (entidades) en riesgo de daño y se basa en una evaluación de los riesgos, los beneficios y la reputación asociados con todas las partes involucradas en una situación dada"[22].

Los sistemas de reputación son descritos como: “un elemento importante para lograr la confianza dentro de las grandes comunidades distribuidas, especialmente cuando los agentes mutuamente desconocidos realizan transacciones ad-hoc,

donde a cada agente basándose en el conocimiento del comportamiento pasado le corresponde construir un criterio sobre su compañero de transacción"[30].

Una valoración en un sistema de reputación es una opinión individual acerca del resultado de una transacción. Para Resnick y Zeckhauser la finalidad de los sistemas de reputación es monitorear el comportamiento de un agente mediante la recopilación, agregación y distribución de tales retroalimentaciones o valoraciones [28].

Conceptualmente los sistemas de reputación poseen la siguiente arquitectura en la que intervienen los siguientes elementos y actores según Schlosser [30] (ver figura 2.4): El objetivo a ser valorado es conocido como ratee[1]. El colector recoge las valoraciones de los agentes, llamados raters[2]. Esta información es procesada y agregada dentro del sistema por el procesador. El algoritmo usado por el procesador para calcular y agregar la representación de la reputación de un agente es la métrica del sistema de reputación. Y finalmente es el emisor quien pone a disposición los resultados a otros agentes que los estén solicitando.

[1] No hay definición específica para este término en español
[2] Raters es el término que se usa para denominar a los agentes evaluadores

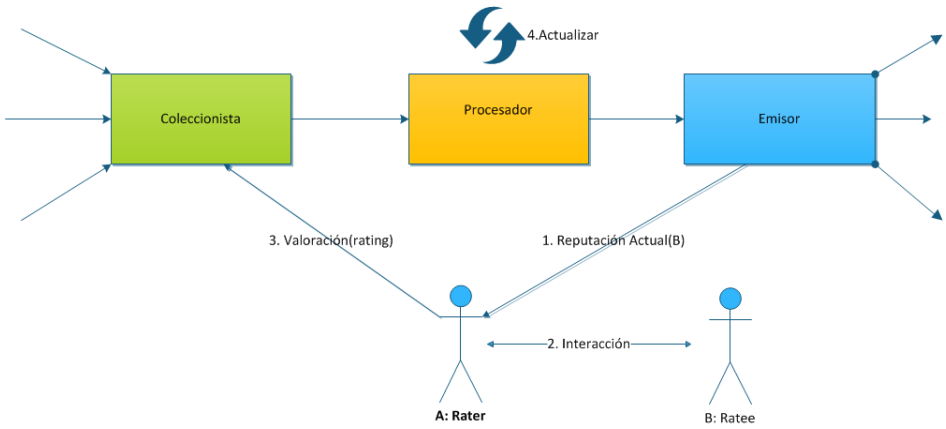


Figura 2.4: Arquitectura de un Sistema de Reputación
Fuente: Comparing and evaluating metrics for reputation systems by simulation[30]

2.3.2 Notación utilizada para métricas de sistemas de reputación

El estudio de la reputación de los sistemas requiere métricas cuantitativas que permitan la evaluación de éstos en un ambiente simulado. Con la finalidad de poder establecer comparaciones entre los diferentes sistemas de reputación se ha decidido adoptar la notación utilizada por Schlosser et al [30].

A continuación introduciremos la notación básica que usaremos a lo largo del presente proyecto: El conjunto de agentes será denotado por A, y el contexto de la transacción se encuentra representado de la siguiente manera: $\mathbf{C} \cdot \mathbf{T} = \{0, 1, ..., t_{now}\}$ E denota el conjunto de encuentros que ha ocurrido entre los agentes hasta ese momento. Un encuentro con información acerca de compañe-

ros en el contexto de una transacción, se puede describir de la siguiente manera:

$$E = \{(a, b, c) \in \mathbf{A} \times \mathbf{A} \times \mathbf{C} \mid a \neq b\} \tag{2.1}$$

En tanto que la calificación será denotado como $\rho(a, e)$ y representa la relación que existe entre un agente destino $a \in A$ y un encuentro $e \in E$ para establecer el conjunto de posibles valoraciones Q:

$$\rho(a, e) : \mathbf{A} \times \mathbf{E} \rightarrow Q \tag{2.2}$$

El conjunto de valoraciones Q puede tomar varias formas, por ejemplo: digamos que queremos referirnos al conjunto de valoraciones manejados por E-Bay, lo haríamos de la siguiente manera: $Q_{eBay} = \{-1, 0, 1\}$.

O podríamos decir que está representado por:

$$Q_i = [-1, 0, 1] \tag{2.3}$$

El subconjunto de todos las relaciones en las que un agente $a \in A$ ha completado una transacción y ha recibido una valoración es:

$$E_a := \{e \in E \mid (e = (a, ...) \vee e = (..., a....))\} \tag{2.4}$$

La lista ordenada con respecto al tiempo de las relaciones establecidas entre los agentes se define como $\overline{E_a}$.

El conjunto de todas las relaciones en la que un agente $a \in A$ y un agente $b \in A$ con un valor de clasificación válido para el agente a está representado por:

$$E_{a,b} := \{e \in E \mid (e = (a, b, ..) \vee e = (b, a..))\} \tag{2.5}$$

Un encuentro o relación entre a y b en un tiempo t está definido de la siguiente forma:

$$e^t_{a,b} \in E_{a,b} \tag{2.6}$$

El operador # indica el tamaño de un conjunto de la lista. La reputación de un agente $a \in A$ es un mapeo entre el agente a y el tiempo $t \in T$. El valor de la reputación más actual del agente a como se muestra a continuación:

$$r(a) := r(a, t_{now}). \tag{2.7}$$

La tabla (ver Tabla 2.1) resume la notación que permite entender la métrica específica para los diferentes sistemas de reputación según Schlosser et al.[30]

Símbolo	Descripción
$\mathbf{A}$	El conjunto de agentes
$\mathbf{C}$	El contexto de la transacción
$\mathbf{T}$	El conjunto de tiempo $(t_0, t_1, t_2, ..., t_{now})$
$\mathbf{E}$	El conjunto de encuentros o relaciones
$\mathbf{Q}$	El conjunto de valoraciones(ratings)
$\rho(a, b)$	La valoración de un mapeo entre un agente objetivo a y un encuentro e con el conjunto de posibles valoraciones Q
E_a	El subconjunto de todos los encuentros E en el cual el agente a ha completado la transacción y recibido una valoración.
$\overline{E_a}$	La lista ordenada en relación al tiempo de E_a.
$E_{a,b}$	El conjunto de encuentros entre el agente a y el agente b con una valoración válida.
$e^t_{a,b}$	Un encuentro entre a y b dado un tiempo t.
$\#$	El tamaño del conjunto de la lista.
$r(a)$	El valor más reciente de reputación del agente a.

Tabla 2.1: Resumen de notación de Métricas

2.3.3 Tipos de sistemas de reputación

Los sistemas de reputación dependen del contexto y ambiente en el que están siendo usados, debido a que la definición de reputación que se establece para el ambiente de un proveedor IaaS diverge de la que se establece para un proveedor de tipo SaaS. En cada uno de los contextos que se mencionaron, como ejemplo en su mayoría la reputación está establecida de acuerdo a las métricas que se han escogido con la finalidad de establecer la comparación entre dos agentes de acuerdo a su valor de reputación. Es por esto que se ha logrado establecer los siguientes tipos de sistemas en base al criterio de las métricas:

- **Sistemas Acumulativos:** aquellos sistemas en los que el valor de la reputación de un agente es el resultado de recoger y acumular todas las calificaciones realizadas (rating[3]). Esto quiere decir que su métrica refleja la fiabilidad de dicho agente basado únicamente en su buen comportamiento.

- **Sistemas Promedio:** se denominan de esta forma, a los sistemas en lo que la reputación es el valor promedio de las calificaciones que tiene un agente. La idea en la que se basa esta métrica es que el agente tiene un tipo de comportamiento en la mayoría de las veces.

- **Sistemas Desenfocados (Blurred[4]):** los sistemas de reputación calculan una suma promedio ponderada de todas las calificaciones obtenidas por un agente, en la cual los valores más recientes tienen mayor influencia en la reputación actual de dicho agente. Estos sistemas basan su reputación en la flexibilidad de la misma, lo que sostiene que el comportamiento de un agente puede variar a través del tiempo.

- **Sistemas basados en la última calificación:** Como su nombre lo indica, estos sistemas se basan de manera más estricta en calcular únicamente la última calificación (rating[5]) obtenida por el agente, son un caso extremo de

[3]Término usado para definir a la calificación que recibe un agente

[4]Término que se usa para denominar a los sistemas desenfocados o mejor conocidos como blurred

[5]Se denomina rating al valor de la calificación obtenida por un agente, también conocida como valoración

los sistemas Blurred.

- **Sistemas EigenTrust[6]:** estos sistemas hacen un cálculo combinando de forma iterativa la reputación local del agente con su reputación global. La reputación local es definida por un conjunto de agentes, por lo cual la misma puede variar en cada sistema local. En cambio los sistemas de reputación global su reputación es la misma en todo el sistema.

- **Sistemas Adaptativos:** el comportamiento de estos sistemas cambia dependiendo de la calificación que ha recibido el valor de reputación actual del agente. Por citar un ejemplo, si el valor actual de la reputación del agente es bajo entonces una calificación positiva causa un incremento considerable en la reputación; por otra parte, si el valor de la reputación actual ya es elevado, la misma calificación producirá una variación casi imperceptible. Lo mismo sucede con las calificaciones negativas

- **Sistemas de valores Betas:** sistemas que tratan de predecir el comportamiento futuro de un agente haciendo uso de estadísticas. Esta predicción hace consideraciones del comportamiento pasado de dicho agente y la probabilidad de que se comporte bien o mal en la siguiente solicitud que le sea enviada.

[6]No existe término en español.

Dicha clasificación puede variar de acuerdo al enfoque que se haya usado para organizarlos en [31]. Dicha clasificación viene basada en el criterio de las métricas, mientras que en [32] han adoptado una en base a las características que presentan dichos sistemas y los han dividido en dos categorías: de acuerdo a las fuentes de información que manejan, y a los tipos de visibilidad. Adicionalmente, los criterios para clasificar los sistemas de reputación pueden verse influenciados por las siguientes consideraciones:

1. Cantidad de esfuerzo requerida por los usuarios para generar reputación.

2. Acciones explicitas realizadas por los usuarios, como dar valoraciones y puntajes.

3. El comportamiento de los usuarios, como por ejemplo las tasas de retorno.

4. La facilidad de comprensión por parte del usuario.

5. Facilidad de implementación para los desarrolladores.

6. Grado de relevancia personal de las calificaciones de los usuarios. Se entiende por **relevancia personal** el grado en que las calificaciones tienen en cuenta los gustos y disgustos de los usuarios o el grado en que las recomendaciones se adaptan a cada usuario.

7. Los sistemas de reputación pueden ser agrupados de acuerdo a la naturaleza de la información que dan sobre el objeto de interés o de acuerdo a la forma que fue generada la calificación o valoración del agente.

Pero, para fines de nuestro proyecto hemos decidido usar la clasificación descrita en [31], ésta segmenta los sistemas de reputación en base al modelo de métricas que se usa para calcular el valor de la reputación.

2.3.4 Enfoque con Múltiples Agentes

Una simulación con múltiples agentes está basada, como su nombre lo indica en un modelo de múltiples agentes que combina este tipo de mecanismo con un ambiente simulado en un tiempo virtual. En un enfoque de simulación con múltiples agentes, las entidades activas son modeladas como agentes que viven en un ambiente que está compuesto por otros agentes, y objetos ambientales o físicos como recursos, que son modelados como entidades. Pero, la clave de este enfoque son las interacciones que ocurren entre los agentes que de manera autónoma toman decisiones en base al modelo de comportamiento que se les ha establecido.

Los sistemas multi-agentes o Multiple Agent Systems, por sus siglas en inglés conocidos como (MASs) tienen varias áreas de aplicación según [32]. Para tener una idea general de las aplicaciones de estos sistemas mencionaremos las principales:

1. Resuelven problemas que son altamente complicados para agentes centralizados debido a sus limitados recursos;

2. Permiten la interoperabilidad e interconexión entre múltiples sistemas existentes;

3. Resuelven problemas que naturalmente conciernen a la sociedad de agentes que interactúan autónomamente.

4. Proveen soluciones usando fuentes de información espacialmente distribuidas; y

5. Mejoran el rendimiento, como por ejemplo: eficiencia computacional, confiabilidad, manejabilidad, respuesta y flexibilidad.

Además para [32] los MASs poseen cuatro características:

1. No hay un control global del sistema

2. Datos descentralizados;

3. Computación asincrónica; y

4. Cada agente tiene un punto de vista limitado, por ejemplo: información incompleta o capacidades limitadas para resolver cierto problema.

2.4 SLA para arquitecturas en Nube

El incremento masivo de la popularidad de la computación en la Nube en los últimos años ha convertido a estos entornos en una fuente de servicios que se basan en contratos que aseguren la satisfacción del cliente. Debido a esto ha sido necesario de parte del proveedor establecer el uso de lo que se denomina un SLA.

Para [10] la provisión de servicios que se realiza en la Nube vienen estipuladas en el SLA, el cual no es más que una especie de contrato firmado entre el cliente y el proveedor del servicio, el cual incluye: los requerimientos no funcionales del servicio que son especificados como la calidad del servicio (QoS), las obligaciones que se consideran a través del SLA, los precios del servicio y además incluye las sanciones en el caso de que hayan violaciones del acuerdo previamente establecido. Adicionalmente, en [42] sostienen que el uso de un SLA está destinado a: “definir una base formal para el rendimiento y la disponibilidad de las garantías del proveedor. Los contratos SLA registran el nivel de servicio especificado por varios atributos, tales como: disponibilidad, capacidad de servicio, rendimiento, funcionamiento, facturación e incluso sanciones en caso de que se incurra en una violación del SLA". Los SLAs más precisos son aquellos que incluyen tanto especificaciones generales como las especificaciones técnicas, incluyendo partes del negocio, políticas de precio y propiedades de los recursos que se requieren para procesar

el servicios[41]. De acuerdo con Sun Microsystems, un buen SLA debe establecer los límites y expectativas de la provisión de servicios, pero además debe garantizar los siguientes beneficios[39]:

- Mejora de nivel de satisfacción del cliente: un SLA definido de manera clara y concisa aumenta el nivel de satisfacción del cliente, porque ayuda a los proveedores a centrarse en las necesidades de los clientes y garantiza que el esfuerzo se pone en la dirección correcta.

- Incrementa la calidad del servicio: cada punto establecido en un SLA corresponde a un indicador clave de rendimiento que especifica el servicio al cliente dentro de una organización específica.

- Mejora la relación entre las dos partes: un SLA claro indica las recompensas y sanciones políticas de una prestación de servicios. El consumidor puede controlar el nivel del servicio que desea consumir de acuerdo a lo que se conoce como objetivos de nivel de servicio o **SLO**(Service Level Objectives por sus siglas en inglés) especificados en el SLA. Por otra parte, establecer dicho contrato ayuda a las partes a resolver los conflictos acerca del precio del servicio con mayor facilidad.

Para Leopoldi [20] un SLA define la capacidad de responder de un proveedor, el rendimiento objetivo de los requerimientos del consumidor , la meta de desempeño

de los consumidores, el alcance de disponibilidad garantizada, y la medición y mecanismos de información. Pero para tener una descripción más comprensiva de los SLA, analizaremos sus componentes que según Jin et. al.[16] son los siguientes (ver figura 2.5):

- **Propósito:** los objetivos a cumplir mediante el uso de un SLA.
- **Restricciones:** pasos o acciones que se deben tomar para garantizar necesario que el nivel solicitado de se prestan los servicios.
- **Período de validez:** tiempo de vigencia del SLA.
- **Alcance:** servicios que serán entregados a los consumidores, y servicios que no se tratarán en el SLA.
- **Partes:** organizaciones o individuos involucrados y sus funciones (por ejemplo: los proveedores y consumidor).
- **Objetivos de nivel de servicio:** niveles de servicios que ambas partes están de acuerdo. Algunos servicios utilizan indicadores de nivel, como disponibilidad, rendimiento y fiabilidad.
- **Sanciones:** si el servicio suministrado no alcanza el SLO o está por debajo de la medición del desempeño, ocurrirán algunas sanciones.

- **Servicios opcionales:** servicios que no son obligatorios, pero podrían ser necesarios.

- **Administración:** los procesos que se utilizan para garantizar la consecución de los SLO y la relación de responsabilidades de organización para el control de estos procesos.

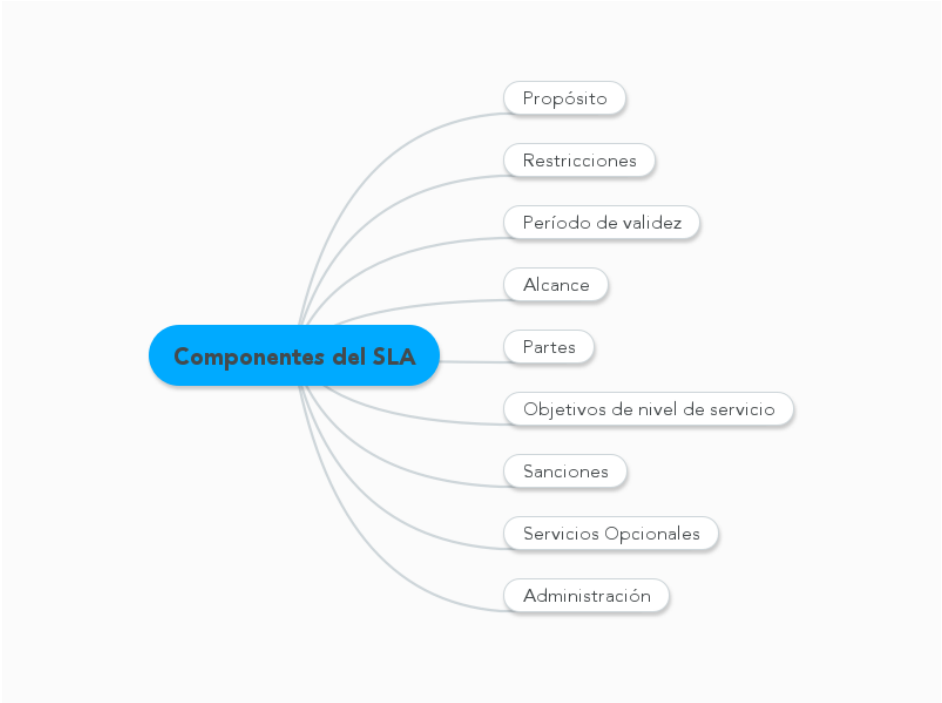


Figura 2.5: Componentes de un SLA
Fuente: Analysis on service level agreement of web services [16]

Por otra parte, al hablar de SLA es necesario sostiene Ron et. al. en [21] definir el ciclo de vida del mismo. Éste se compone de tres fases: la primera fase es la fase de creación, en la cual el consumidor encuentra un proveedor de servicio que se ajuste a sus requerimientos de servicio. Luego pasamos a la fase de operación en

la cual el consumidor tiene acceso sólo de lectura al SLA. En tercer lugar se lleva a cabo la fase de eliminación, en la cual el SLA es terminado y todas las configuraciones de información asociadas son eliminadas de los sistemas de servicio(ver figura 2.6).

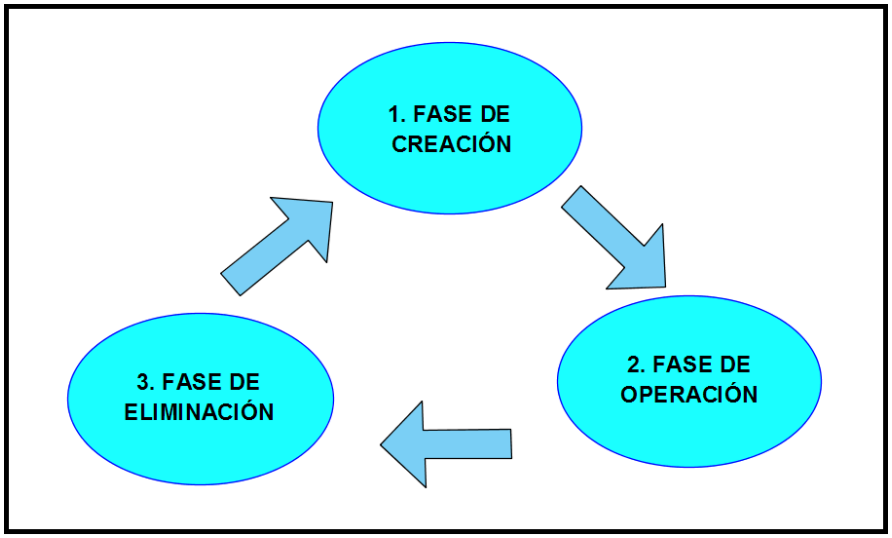


Figura 2.6: Ciclo de vida de un SLA
Fuente: Client classification policies for sla negotiation and allocation in shared cloud datacenters [21]

2.4.1 Tipos de SLAs

El uso de los recursos por usuarios externos puede afectar la calidad del servicio que reciben los usuarios internos si el SLA no establece previamente prioridades entre clientes en base al precio y al almacenamiento de recursos. En [21] se sugiere aplicar a los SLAs una clasificación basada en el cliente con la finalidad de mantener un calidad de servicio elevada. Dicha clasificación considera la información sobre los usuarios cuando les otorga acceso a los recursos y jerarquiza los

SLAs en base a dos criterios:

- QoS que los usuarios esperan adquirir: a mayor calidad de servicio, mayor precio. Ésta es la manera tradicional de clasificar los servicios.

- Afinidad entre el cliente y el proveedor: clientes de la misma compañía que el proveedor o de entidades que tienen una relación privilegiada con el proveedor pueden contratar los servicios a mejores precios, mejor calidad de servicio u obtener cualquier otro privilegio adicional.

2.4.2 Calidad de Servicio (QoS)

La calidad de servicio es un término extenso, ampliamente utilizado para describir la experiencia general que va a recibir un usuario o una aplicación a través de la red [14]. Una Nube se caracteriza por poseer múltiples proveedores, cada uno con su propio sistema de gestión de términos de servicio, plataformas operativas, y niveles de seguridad. Estos entornos también pueden ser dinámicos por la forma automática del enrutamiento de datos, aplicaciones y las necesidades de infraestructura basados en algún criterio de calidad de servicio (QoS), como disponibilidad, fiabilidad, latencia, precio, etc. Por lo tanto, la calidad de servicio queda totalmente definida en base a los diferentes requerimientos del usuario, cómo estos fueron cumplidos y en qué nivel se cumplieron dichas expectativas.

En [38] se plantean algunas suposiciones acerca de los parámetros de QoS, que son de aplicación general :

- Los requisitos de calidad de servicio son cuidadosamente identificados desde el punto de vista del usuario, para que un conjunto de indicadores y valores de referencia pueden ser definidas en base a estos requisitos.

- Los indicadores pueden ser medidos y monitoreados con respecto a estos valores de referencia para comprobar si se cumplen los requisitos.

- Los valores de referencia están bien definidos (posiblemente en las normas) y se incluyen en el contrato entre el proveedor y el cliente.

Para [38] estos supuestos son aplicables para los entornos Nube, además hacen énfasis en que una evaluación adecuada de la calidad de un determinado servicio requiere una serie de acciones:

- Análisis de los requerimientos específicos de calidad de servicio del usuario.

- Elección de los indicadores más apropiados.

- Definición del método más adecuado de medición y;

- Definición del valor de referencia adecuada para el indicador.

Adicionalmente, en nuestros días a los usuarios de este tipo de servicios en la Nube no sólo les preocupa la funcionalidad integrada en los servicios del software, sino que además están prestándole atención a los parámetros QoS que contratan y basan su concepto de calidad en requerimientos no funcionales como por ejemplo: fiabilidad, seguridad, rendimiento, entre otros. En [25] ponen a consideración una nueva clasificación de dichos requerimientos basada en la propuesta de Kotonya y Sommerville, ISO/IEC 9126 y Lamsweerde, que se acerca un poco más a las características y naturaleza de estos entornos en la Nube. Para nuestro proyecto nos enfocaremos en la clasificación de estos requerimientos no funcionales o NFRS (Non Functional Requeriments por sus siglas en inglés), los cuales para servicios orientados a la ingeniería de software se refieren al comportamiento general de una aplicación orientada a servicios que se desarrollan partir de la composición de servicios reutilizables. Por ejemplo, la calidad de servicio se atribuye al cumplimiento de normas y regulaciones que están relacionados de manera general con el comportamiento de toda la aplicación orientada a servicios. Dicho cumplimiento de QoS puede verse afectado de manera directa por la arquitectura del servicio y otras consideraciones del software, por lo cual cuando se ofrece el servicio en el SLA se recalca cuáles serán los requerimientos no funcionales que entrarán a ser considerados como QoS. En el **Capítulo 3** se detallarán dichos requerimientos, en la figura 2.7 se muestran algunos ejemplos.

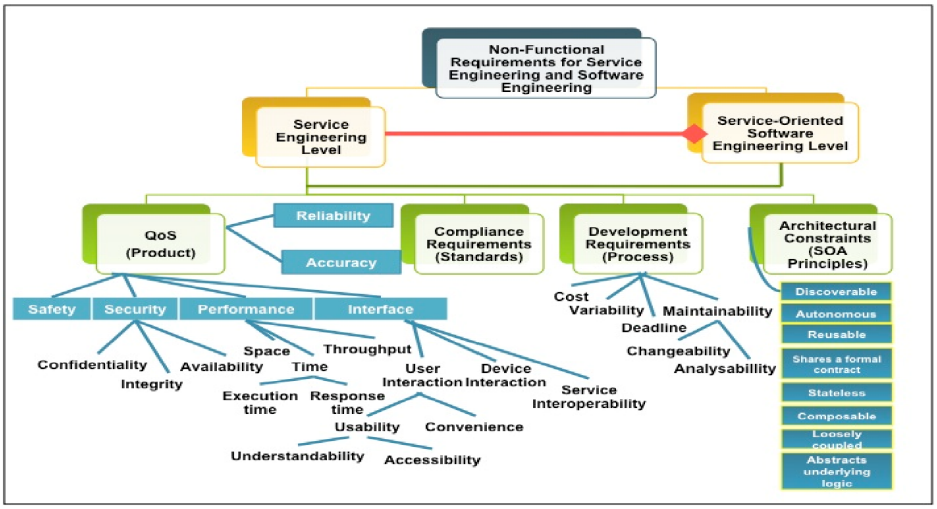


Figura 2.7: Requerimientos No Funcionales
Fuente: A new classification of non functional requirements for service-oriented software enineering [25]

2.5 Software de Simulación CloudSim

Según lo revisado anteriormente, la computación en la Nube proporciona infraestructura, plataforma y aplicaciones como servicios basados en un modelo de pago por uso. Estos servicios pueden ser de distinto tipo tales como redes sociales, alojamiento web, proveedores de contenido, entre otros. Al ser diferentes, los requisitos de hardware y las configuraciones para que el servicio funcione correctamente varían de un servicio a otro, lo que implica que si se requiere evaluar el rendimiento del servicio y la Nube bajo diferentes configuraciones del sistema y de usuarios es difícil de lograr y costoso [6].

El uso de plataformas reales tales como Amazon EC2 [1], Microsoft Azure [7],

Google App [15] para evaluar el rendimiento y la relación costo-beneficio de las aplicaciones sujetos a condiciones variables, como disponibilidad y carga de trabajo, están limitados a la rigidez de la infraestructura [5]. Por tanto, esto hace que los resultados de los experimentos sean difíciles de replicar, pues resulta costoso en términos de tiempo volver a configurar las variables que intervienen en la prueba. Y no todas las configuraciones pueden ser manejadas o controladas por los programadores de las aplicaciones; estos afectan de manera directa a la evaluación. Debido a esto, se busca realizar los experimentos en herramientas de simulación [3]. Estas herramientas brindan la posibilidad de realizar pruebas de rendimiento en un entorno fácil de controlar permitiendo de esta manera reproducir los resultados. Según [34] las pruebas basadas en herramientas de simulación permiten:

i. Probar y evaluar las aplicaciones en un entorno repetible y controlable.

ii. Ajustar el sistema a escenarios críticos como: cuellos de botella antes de ser implementado en Nubes reales

iii. Experimentar con diferentes cargas de trabajo para evaluar las políticas de asignación de recursos dentro de la Nube.

La herramienta de simulación CloudSim permite modelar y simular entornos Nubes controlables en los que desarrolladores e investigadores pueden evaluar el rendimiento de la aplicación de una manera fácil. CloudSim presenta las siguientes

características:

1. Soporte para el modelamiento y simulación de Nubes de larga escala incluyendo sus centro de datos.

2. Una plataforma autónoma para el modelamiento de Nubes, servicios, brokers y políticas de asignación.

3. Soporte para la simulación de conexión de red entre los elementos del sistema simulados.

2.5.1 Enfoque de Simulación

En la figura 2.8 se muestra las capas de la plataforma de CloudSim y los componentes de su arquitectura utilizados para la simulación. A continuación daremos una breve descripción de ellos.

La capa base, **SimJava**, es el motor de simulación donde se encuentran implementadas las funcionalidades requeridas en los niveles más altos, como colas y procesamiento de eventos, creación de los componentes del sistema (servicios, centros de datos, broker, máquinas virtuales), la comunicación entre los componentes y la gestión del reloj de la simulación.

En el siguiente nivel se encuentran las librerías que implementan **GridSim** [19]. Estas son un conjunto de herramientas que sirven de apoyo a los componentes de niveles superiores en la implementación de componentes fundamentales como: recursos, conjuntos de datos, servicios de información.

CloudSim es implementado en el siguiente nivel, a través de recursos de programación extiende las funcionalidades centrales expuestas por la capa de **GridSim**. Es así como **CloudSim** proporciona soporte para el modelado y simulación de centros de datos virtualizados basados en la Nube. **Cloudsim** posee además interfaces para el manejo de máquinas virtuales, memoria, almacenamiento y ancho de banda durante el periodo de simulación.

Las tareas fundamentales como el aprovisionamiento de hosts[7] para máquinas virtuales en función de las peticiones del usuario, administración de la ejecución de la aplicación y el monitoreo dinámico son gestionados por la capa previamente mencionada.

Con la finalidad de estudiar la eficacia de las distintas políticas de asignación de hosts, se requiere extender y modificar el núcleo de la funcionalidad utilizada para el aprovisionamiento de una **VM** (Virtual Machine[8]). Considerando que un host puede ser al mismo tiempo compartido entre un número de máquinas virtuales que

[7] Término plural de host
[8] Término en inglés utilizado para definir máquina virtual

ejecutan las aplicaciones en función de las especificaciones de QoS definidas por el usuario.

La capa de nivel superior es **UserCode**, la misma que expone funcionalidades de configuración para hosts, aplicaciones, VM, número de usuarios y sus tipos de aplicación, y políticas de planificación.

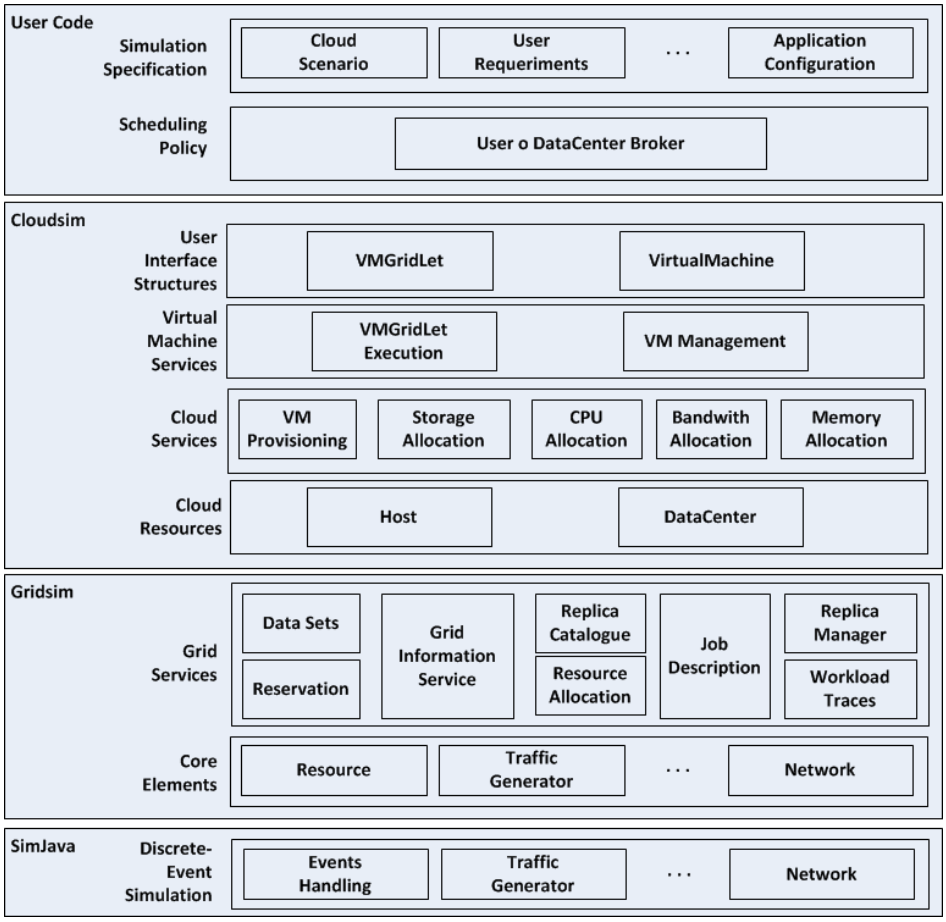


Figura 2.8: Plataforma de CloudSim
Fuente: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms [5]

2.5.2 Requerimientos

- **Modelando la Nube:** los servicios principales de infraestructura de hardware relacionados con las Nubes para el manejo de solicitudes de servicio son modelados en el simulador por un Datacenter[9]. Éstas solicitudes son procesadas por agentes en las máquinas virtuales. Dicho Datacenter está compuesto por un conjunto de hosts y es responsable de la administración de las VMs durante su ciclo de vida. Host[10] es un componente que representa una computadora física, un nodo, en una Nube. Al Host se le asignan recursos de procesamiento, memoria, almacenamiento, políticas de planificación para que pueda llevar a cabo la asignación de núcleos. Este componente implementa interfaces que soportan modelado y simulación de nodos de en un núcleo y varios núcleos.

  La asignación de máquinas virtuales específicas para un host es responsabilidad del componente **Virtual Machine Provisioner**. Dicho componente cuenta con métodos personalizados que ayudan en la implementación de nuevas políticas de asignación de máquinas virtuales. La política implementada por el Virtual Machine Provisioner es una política directa que asigna una máquina virtual al primer host en llegar usando First-Come-First-Serve

[9] Término en inglés utilizado para denominar un centro de datos
[10] Término en inglés usado para denominar un nodo que aloja un determinado servicio

(FCFS[11]). La política toma en cuenta cuantos núcleos de procesamiento deben ser asignados a cada VM.

Además, es posible asignar núcleos de CPU específicos para una VM específica o distribuir de forma dinámica la capacidad de un núcleo entre máquinas virtuales. Cada Host instancia un componente VM scheduler que implementa la política de espacio compartido (space-shared) o tiempo compartido (time-shared) para la asignación de núcleos a máquinas virtuales.

- **Modelando la asignación de VM:** cloudSim soporta VM scheduling en dos niveles: a nivel de host y a nivel de VM. En el primer nivel, es posible especificar la cantidad de poder de procesamiento total de cada núcleo que es asignado por cada host a la máquina virtual. En el siguiente nivel, la VM asigna cierta cantidad de poder de procesamiento a las unidades de trabajo individuales que están alojadas dentro de su motor de ejecución, denominados cloudlets[12].

- **Modelando el mercado en la Nube:** en el diseño de un simulador de Nube existen dos aspectos que se deben considerar como importantes: el modelado de los costos y las políticas de precio. Para realizar el proceso de modelado de la Nube cuatro propiedades relacionadas con el mercado son

[11] Siglas para denominar en inglés al algoritmo primero en llegar primero en ser atendido.
[12] Término sin traducción al español

asociadas al datacenter: costo por procesamiento, costo por unidad de memoria, costo por unidad de almacenamiento, y costo por unidad de ancho de banda utilizado. Costo por memoria y almacenamiento son los costos en los que incurre durante la creación de la máquina virtual. El costo por el ancho de banda es aquel en el que se incurre durante la transferencia de datos. En el caso de que las máquinas virtuales se hayan creado pero ninguna unidad de trabajo se ejecuta en ellas, sólo se incurrirán en costos de memoria y almacenamiento. Este comportamiento puede ser modificado por los usuarios.

2.5.3 Arquitectura

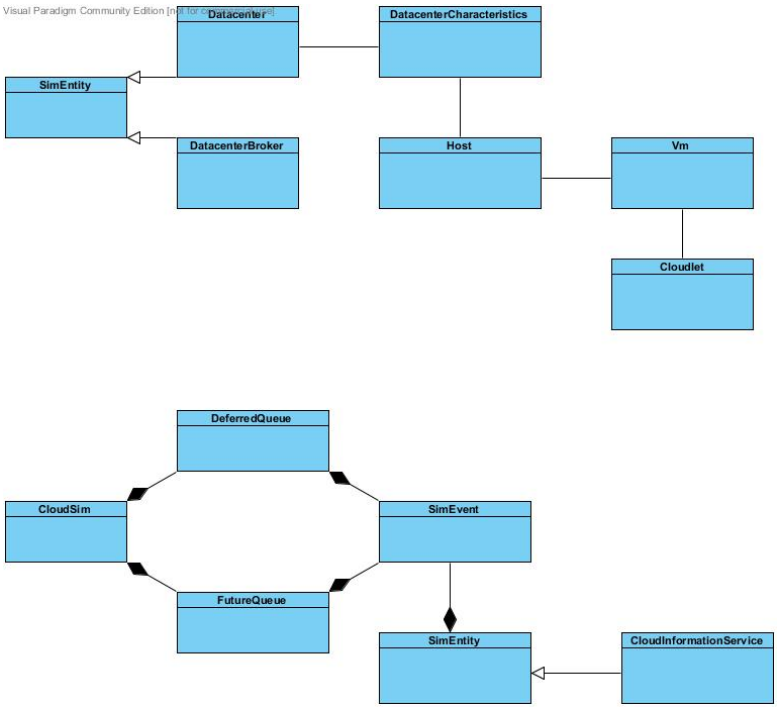


Figura 2.9: Arquitectura de CloudSim

CloudSim posee un gran conjunto de clases que hacen posible la simulación de Nubes y que permiten la ejecución de escenarios complejos en la plataforma. Las clases más relevantes se muestran en la figura 2.9. A continuación daremos una breve descripción de ellas.

I. **CloudSim:** es la clase principal, responsable de la gestión de colas de eventos y de controlar la ejecución secuencial de los eventos que intervienen en la simulación. Los eventos generados en tiempo de ejecución son almacenados y ordenados en base a un parámetro de tiempo, en una cola denominada cola de eventos futuros. Luego, los eventos programados en cada paso de la simulación son extraídos de esta cola y asignados a una cola de eventos diferidos. Un método de procesamiento de eventos es invocado para cada entidad. Además, un evento de la cola de eventos diferidos es seleccionado para las acciones que se requieran. Este manejo organizado de entidades y eventos que intervienen en la simulación, tiene las siguientes características:

   - Desactivación de entidades.

   - Permite el cambio de contexto de entidades entre diferentes estados. Haciendo posible llevar a cabo acciones como: pausar y reiniciar el proceso de simulación.

   - Creación de entidades en tiempo de ejecución.

- Interrumpir y reiniciar la simulación en tiempo de ejecución.

II. **DeferredQueue:** esta clase es la encargada de implementar la cola de eventos diferidos utilizada por CloudSim.

III. **FutureQueue:** esta clase implementa la cola de eventos futuros que es accedida por CloudSim.

IV. **CloudInformationService(CIS):** es una entidad que permite el registro e indexación de recursos. CIS soporta dos primitivas básicas:

   a) Publicar, lo cual permite a las entidades registrarse en CIS.

   b) Buscar, que permite que entidades como Cloud Coordinator y Brokers descubran el estado y la dirección de contacto del punto final de las otras entidades.

   Esta entidad es la encargada de notificar a otras entidades acerca del final de la simulación.

V. **SimEntity:** es una clase abstracta. Representa una entidad de simulación capaz de enviar mensajes a otras entidades o procesos y procesar los mensajes recibidos. Todas la entidades deben heredar de esta entidad y sobrescribir los siguientes tres métodos principales: `startEntity()`, `processEvent()` y

`shutdownEntity()`, que definen acciones relacionados a: Inicializar entidades, procesar eventos y destruir entidades, respectivamente.

VI. **DataCenter:** es la clase principal que modela el núcleo de la infraestructura de niveles de servicios de hardware y software. Encapsula un conjunto de hosts cuyas configuraciones relacionadas a memoria, núcleo, capacidad y almacenamiento pueden ser homogéneas o heterogéneas. Cada componente del DataCenter instancia un componente para el aprovisionamiento de recursos que implementa políticas para la asignación de dichos recursos.

VII. **DatacenterBroker:** esta clase modela un broker, el cual actúa como un intermediario entre los usuarios y el proveedor de servicios, dependiendo de los requerimientos de los usuarios (QoS). Investigadores y desarrolladores extienden de esta clase para experimentar con la aplicación y las políticas de asignación desarrolladas por ellos.

VIII. **VirtualMachine:** esta clase implementa una instancia de una VM. Del manejo de una VM es responsable el host. Un host puede instanciar simultáneamente múltiples VMs y asignar núcleos basados en una política predefinida de intercambio de procesador.

2.6 Mecanismos de mercado

A lo largo de los años, los seres humanos han ido estableciendo formas de intercambiar objetos o servicios entre ellos, llegando a establecer desde sus inicios nociones básicas de negociación, las cuales posteriormente formarían parte de lo que en nuestros días conocemos como mercado. En [40] se sostiene que las personas han usado los mercados para satisfacer sus necesidades y que además, dichos sistemas basados en mecanismos de mercado pueden ser: “cualquier sistema que utiliza el concepto o ciertas características que se encuentran en un mercado". Con la finalidad de entender más acerca de los mecanismos de mercado, analizaremos como surgieron los mismos a través de los diferentes cambios en los sistemas económicos.

2.6.1 Evolución de los mecanismos de mercado

La experimentación ha sido un campo importante en ciencias como biología, química y física. Antiguamente, la economía era considerada una disciplina no experimental. En la actualidad esto ha cambiado radicalmente, economistas e investigadores han realizado numerosos experimentos buscando probar hipótesis que permitan describir o explicar modelos económicos. Además, las pruebas se realizan en un ambiente controlado y se basan cada vez más en datos generados en

un laboratorio, en lugar de datos de campo. Los experimentos económicos están centrados en tres áreas según la clasificación de Chamberlin (1984): experimentos de mercado, experimento de juegos y experimentos enfocados en la elección individual, los cuales describiremos a continuación:

2.6.1.1 Tipos de experimentos

I. **Experimentos de mercado:** de acuerdo a lo revisado anteriormente, los experimentos de mercado son utilizados en investigaciones que permiten describir o establecer nuevos modelos y teorías que expliquen la actividad económica. Se enfocan en analizar las predicciones de las teoría de precios por ejemplo aquellos experimentos descritos por Chamberlin(1984) sobre la demanda inducida y la estructura del costo y la subasta doble por Vernon Smith[9].

II. **Experimentos de juego:** estos experimentos comenzaron a realizarse a finales de 1950 y principios de 1960 por sociólogos y psicólogos sociales quienes se mostraban escépticos a los resultados predichos por el famoso juego del “dilema del prisionero". Basado en dicho juego, los experimentos se enfocaron en estudiar entornos en los que la decisión de cada participante afectaba directamente al resto de participantes.

III. **Experimentos de elección individual:** debido al escepticismo entre los psi-

cólogos con respecto a la relevancia de la teoría de la utilidad propuesta por Von Neumann y Morgenstern (1944), surge este tipo de experimentos. Dicha teoría constituye la base para los modelos de asignación en los entornos. En este tipo de experimentos la incertidumbre que afecta a la toma de decisiones proviene de agentes externos y no de las decisiones de otros agentes.

Las dos ventajas principales de los experimentos de mercado son: el control y la replicación. El control es la capacidad de configurar el escenario de acuerdo a las condiciones requeridas para evaluar la teoría. La replicación es la capacidad de configurar el entorno con los mismos datos de experimentos anteriores para reproducir el experimento varias veces y que permita concluir, en base a los resultados obtenidos, si la hipótesis planteada es aceptada o no.

Posteriormente a las diferentes experimentaciones de la economía y con el desarrollo de la sociedad, se pudo poner en practica dichos experimentos a través de lo que ahora conocemos como mecanismo de mercado. A continuación ponemos a consideración el mecanismo de mercado más relevante y que se consideró para el proyecto: la Subasta.

2.6.2 Subastas

Se ha generado un gran interés en la utilización de mecanismos de mercado para la asignación de recursos en la Nube. Estos mecanismos buscan mejorar la eficiencia de las políticas de asignación utilizadas por los proveedores. Uno de los mecanismos más estudiados es el mecanismo denominado subasta. En base a lo revisado con anterioridad, el mercado es una institución de intercambio activo compuesto por vendedores y compradores. Los compradores compran cuando su disposición a pagar supera el precio y los vendedores venden cuando su disposición a aceptar cae por debajo de los precios, el objetivo del comprador es minimizar su gasto y el del vendedor es aumentar su ganancia. La subasta está definida como: “una de las muchas maneras que un vendedor puede utilizar para vender un objeto a los compradores potenciales con valores desconocidos” según [18]. Las subastas son usadas debido a que el vendedor desconoce el máximo valor que el comprador puede estar dispuesto a pagar. La incertidumbre con respecto a los valores que puedan llegarse a establecer en una negociación entre vendedores y compradores, es la característica principal de estos mecanismos. El precio del objeto que el comprador desea adquirir se convierte en una pugna entre los diferentes compradores, la misma que está basada en reglas definidas por el vendedor. De acuerdo con el mecanismo que el comprador utilice para fijar el precio que está dispuesto a pagar se tiene la siguiente clasificación:

a. **Subastas de valor privado:** cada comprador establece el valor del objeto según su criterio. Éste no posee conocimiento de los valores atribuidos por los otros compradores y el valor que asignen otros compradores no afecta al precio establecido.

b. **Subastas de valor común:** el valor de un objeto es interdependiente es decir, el valor que se asigne a un objeto depende del valor asignado por los demás compradores.

c. **Subastas de valores correlacionados:** es una combinación de las clasificaciones anteriores. El precio que se asigne al objeto depende del criterio/preferencia del comprador y del precio asignado por otros compradores.

Adicionalmente, según el mecanismo que el vendedor utilice para determinar al comprador ganador de la subasta, tendremos la siguiente clasificación:

i. **Subasta inglesa:** cada comprador hace una oferta[13]. El precio se incrementa en cada oferta hasta que no haya compradores interesados en realizar otra oferta. El objeto es vendido al comprador con la oferta más alta.

ii. **Subasta holandesa:** es la contraparte de la subasta inglesa. El vendedor establece un valor demasiado elevado para el producto. El precio se va disminuyendo de manera gradual hasta que un comprador resulte interesado.

[13]Quiere decir que el comprador establece un valor por el objeto

iii. **Subasta de oferta establecida al primer precio:** el funcionamiento es sencillo, todos los compradores hacen ofertas por el objeto sin conocimiento de las otras ofertas. Gana la subasta aquel que tenga la oferta con el precio más alto.

iv. **Subasta de oferta establecida al segundo precio:** todos los compradores realizan ofertas por el objeto. Gana la subasta aquel que haya realizado la oferta más alta, pero no paga el valor que ofertó sino el valor de la segunda subasta más alta.

v. **Subasta doble:** en este tipo de subasta los compradores realizan el pedido de lo que desean comprar y el vendedor las ofertas de lo que quiere vender de manera asíncrona y simultánea. En cualquier momento un vendedor puede aceptar el requerimiento del comprador, y el comprador puede aceptar la oferta de un vendedor[8]. Según los resultados obtenidos por Smith[17], la subasta doble genera un mercado competitivo en cuanto a las cantidades y precios. Aun en situaciones extremas de estructura(monopolio), y condiciones inusuales en la oferta y la demanda los resultados son más fiables en comparación con otro mecanismo.

vi. **Subasta de oferta publicada(Posted Offer[14]):** otro mecanismo utilizado comúnmente, es la subasta de oferta publicada(posted-offer), en la cual el proceso

[14] En el presente documento usaremos el término posted offer para referirnos a este tipo de subasta

de negociación consiste de dos pasos:

a. Los precios de los vendedores son recopilados y publicados.

b. Después de que todos los precios de los vendedores se publican, los compradores se toman de manera aleatoria de una cola de espera, y se les da la posibilidad de realizar las compras a los precios publicados, estableciendo lo que se conoce como un «lo tomas o lo dejas ».

vii. **Subasta Inversa(Reverse Auction)**[15]**:** en las subastas mencionadas anteriormente los compradores realizan la oferta y aquel que tenga la oferta más alta gana. En un subasta inversa sucede lo contrario, es decir el comprador anuncia la necesidad de un servicio. Los vendedores realizan las ofertas del servicio esperando ser elegidos por el comprador. Aquel vendedor que realice la oferta más baja gana. Este tipo de oferta hace que esta sea más competitiva, permitiendo a los vendedores disputar entre ellos para vender el servicio. Esto conlleva a una reducción del precio a pagar por parte del comprador. En [40] se destacan las siguientes características propias de la subasta inversa:

a. Reducción de precios.

b. Aumento de la competencia.

[15]En el presente documento usaremos el término reverse auction para referirnos a este tipo de subasta

c. Ahorro de tiempo.

d. Aumento de los proveedores.

CAPÍTULO 3

3. Diseño del mecanismo de mercado para la simulación

3.1 Descripción del modelo

El presente modelo tiene como objetivo describir la implementación de un sistema que se encarga de la asignación de recursos de proveedores de servicios a diferentes usuarios de una Nube basada en un mecanismo de mercado. Dicha asignación se realizó buscando cumplir un SLA definido entre el usuario y el proveedor. Con la finalidad de garantizar una asignación adecuada, este sistema se implementó utilizando un mecanismo de mercado que considere el costo del servicio y la reputación del proveedor como factores importantes al momento de escoger el proveedor al que debía ser asignado el recurso. Dado que nuestro objetivo era reducir las violaciones de SLAs por parte de los proveedores se decidió que la reputación del

proveedor dependería de los servicios que éste pudiera alojar de manera exitosa.

3.1.1 Elementos del modelo

En el modelo establecido, fueron definidos los siguientes componentes:

- SLA.
- Service.
- User.
- Broker.
- Provider.
- Market Mechanism.
- Reputation System.
- Trading Mechanism.

A los componentes se les asignó nombres en inglés. A continuación el gráfico (ver figura 3.1) muestra la arquitectura del modelo.

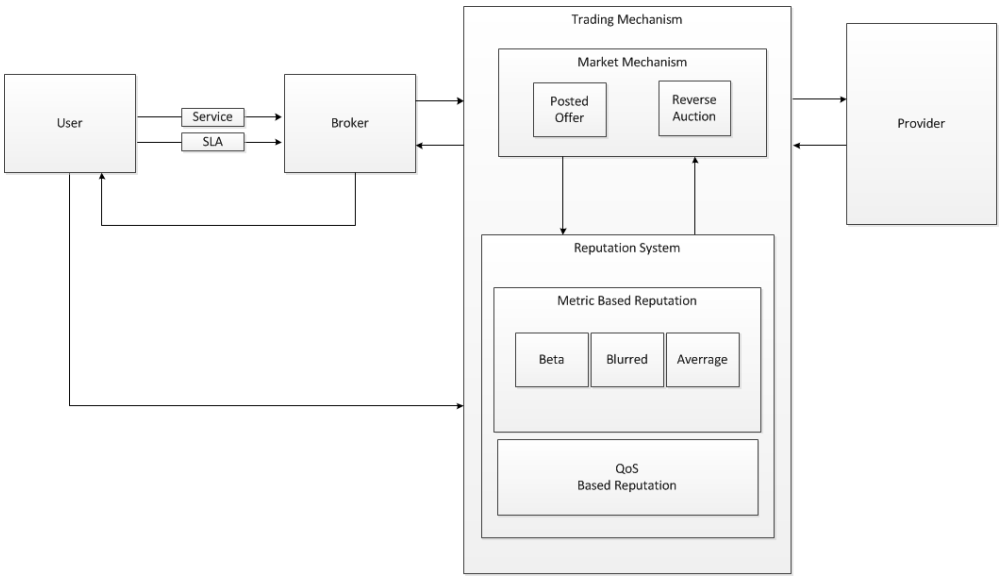


Figura 3.1: Componentes del modelo

3.1.1.1 SLA

Define la entidad que representa el SLA entre el usuario y el proveedor, en el cual se acuerdan los indicadores de calidad o parámetros QoSs[1]: $Availablity$[2], $Reliability$[3], $Perfomance$[4] y $Cost$[5]. En cada parámetro se establece el valor mínimo que debe cumplir el proveedor al alojar un servicio para que no sea considerado una violación. En la sección de **Anexos** (ver Anexos A) ilustraremos la plantilla del SLA usado para la simulación, es decir pondremos a su consideración todos los aspectos que serán considerados dentro de este acuerdo entre el proveedor y el usuario.

[1] Los nombres de los parámetros QoS serán usados en inglés a lo largo de este documento.
[2] Término en inglés para Disponibilidad
[3] Término en inglés para Confiabilidad
[4] Término en inglés para rendimiento
[5] Término en inglés para costo

3.1.1.2 Service

Esta entidad define el servicio a solicitar por parte del usuario. Establece la cantidad de Millones de Instrucciones Por Segundo (MIPS) a ejecutarse en el proveedor y la mínima reputación que debe tener un proveedor para que sea considerado un candidato para alojar el servicio.

3.1.1.3 User

Este componente define el servicio a solicitar y el SLA con los requerimientos no funcionales que deben cumplirse por parte del proveedor. Ésta entidad delega al broker para que maneje el proceso de solicitar el alojamiento del servicio en uno de los proveedores existentes en el mercado. El usuario califica al proveedor con un valor de rating de 1 para servicios alojados exitosamente y con -1 para no exitosos.

3.1.1.4 Broker

El componente Broker actúa de intermediario entre el usuario y el proveedor. Gestiona la elección del mejor proveedor y una vez ejecutada la tarea, notifica al usuario para que califique al proveedor. Adicionalmente, por cada tarea ejecutada de manera exitosa recibe una comisión.

3.1.1.5 Provider

Este componente es el propietario de los recursos y compite con otros proveedores por ejecutar las tareas solicitadas por los usuarios. Además, establece el costo mínimo y máximo que está dispuesto a cobrar por el alojamiento servicio. Lleva un registro de los servicios solicitados, el estado de dicho servicio es decir si fue alojado con exito o no, el rating asignado por el usuario para dicho servicio y el tiempo que tomó realizar la transacción.

3.1.1.6 Market Mechanism

Este componente representa el mecanismo de mercado a utilizar en el proceso de selección de proveedores. Su función es de seleccionar al proveedor más idóneo para alojar el servicio solicitado por el Broker en uno de los proveedores. Se seleccionó Posted Offer y Reverse Auction. En la sección **3.3** se ampliarán los detalles sobre mecanismos de mercado considerados en la presente implementación.

3.1.1.7 Reputation System

Es el componente encargado de evaluar la reputación de los proveedores. La reputación es calculada de dos maneras: basada en el historial de los servicios alojados y basada en el cumplimiento de los parámetros QoS definidos en el SLA. Hemos

denomidado a estos dos tipos de reputación como **Metric Based Reputation** y **QoS Based Reputation** respectivamente. En la sección **3.1.2** se describe con más detalle la implementación de este componente.

3.1.1.8 Trading Mechanism

Es la entidad que contiene los parámetros a ser utilizados por los demás componentes en el proceso de selección y alojamiento del servicio en la nube. Mantiene registro de la lista de proveedores, la lista de usuarios, la lista de brokers, la lista de servicios solicitados, el mecanismo de mercado y el sistema de reputación que se utilizaran en la simulación. Dado que el mecanismo de mercado y el sistema de reputación pueden tomar distintos valores esta entidad actúa como una especie de caja negra ya que los demás componentes realizan la consulta a la entidad sin preocuparse por los valores que tomen el mecanismo de mercado y el sistema de reputación.

3.1.2 Diseño del sistema de reputación

Siendo éste uno de los principales componentes de nuestro sistema, optamos por realizar el cálculo de la reputación con dos tipos de algoritmos: Metric Based Reputation y QoS Based Reputation. Buscamos de esta manera recabar evidencia que demuestre que el número de violaciones por proveedor se reduce si se toma en

cuenta la reputación en el proceso de selección de éste, independientemente del tipo de algoritmo utilizado para establecer su reputación. Además, nos permitirá analizar si existe una diferencia significativa en el número de violaciones por algoritmo.

A continuación, se detallan las características y diferencias de los algoritmos implementados:

I. **Metric Based Reputation:** denonimamos **Metric Based Reputation** a la reputación que es calculada con un algoritmo que considera el rating de los servicios alojados en el proveedor. En nuestro caso particular fueron seleccionadas dos métricas existentes: beta y blurred, pero además como aporte decidimos combinar estas métricas usando una fórmula de promedio, obteniendo así una nueva métrica a la que llamamos **average**.

   a.) **Beta Based Reputation:** se basa en una métrica que predice el comportamiento de un agente, en nuestro caso del proveedor, en su próxima transacción en base a su comportamiento anterior. Catalogando las acciones de dicho proveedor como acciones buenas r y acciones malas s, las cuales se combinan mediante el uso de las siguientes fórmulas:

   Primero, se debe establecer $r(a)$ que representa el valor de las acciones buenas realizadas por el proveedor:

$$r^a = \sum_{i=1}^{\#\overline{E_a}} \lambda^{\#(\overline{E_a})-\frac{i(1+\rho(a,\overline{E_a}[i]))}{2}} \tag{3.1}$$

Ecuación 3.1: **Tomado de** [32]

y las acciones malas $s(a)$ así; Con lo cual la reputación del proveedor

$$s^a = \sum_{i=1}^{\#\overline{E_a}} \lambda^{\#(\overline{E_a})-\frac{i(1-\rho(a,\overline{E_a}[i]))}{2}} \tag{3.2}$$

Ecuación 3.2: **Tomado de** [32]

basada en esta métrica denotado por $a \in A$ queda definida por la siguiente ecuación: donde $0 \leq \lambda \geq 1$

$$r(a)_{Beta} = \frac{r^a - s^a}{r^a + s^a + 2} \tag{3.3}$$

Ecuación 3.3: **Tomado de** [32]

b.) **Blurred Beta Reputation:** el uso de esta reputación permite cálculos que no dependen del tiempo, pero donde se consideran todos los trabajos realizados por el agente(proveedor) pero se les da una mayor ponderación al rating de los trabajos más recientes. Asumiendo que es **altamente probable** que dicho proveedor se comporte como lo hizo en sus transacciones más recientes. Basado en este supuesto, la reputación de un proveedor $a \in A$ viene dado por: donde los valores que el rating ρ puede tomar están definidos por el siguiente conjunto de valores:$\{-1, 0, 1\}$

$$r(a)_{Blurred} = \sum_{i=1}^{\#\overline{E_a}} \frac{\rho(a\overline{E_a}[i])}{\#(\overline{E_a}) - i + 1} \tag{3.4}$$

Ecuación 3.4: **Tomado de** [32]

c.) **Average Combinated Reputation:** esta métrica ha sido definida por nosotros mediante la combinación de las dos métricas anteriores, realizando un promedio entre las dos con la finalidad de mejorar la reputación tomando en cuenta tanto las transacciones más recientes del proveedor como a su vez todo su historial y así poder asignarle una reputación acorde a su comportamiento. Por lo tanto la reputación del proveedor ha quedado definida por $a \in A$:

$$r(a)_{Average} = \frac{r(a)_{Beta} + r(a)_{Blurred}}{2}, \tag{3.5}$$

Con la finalidad de ilustrar el proceso para establecer la reputación en base a las métricas, presentamos una tabla que considera valores para el rating $\{-1, 0, 1\}$ y que reemplazando dichos valores en las fórmulas establecidas para dichas métricas, se espera que el valor de la reputación tenga un valor positivo o negativo.

Estos valores pueden ser positivos o negativos de acuerdo con la fórmula en la que se han reemplezados, la tabla (3.1) resume un bosquejo de cómo sería

Rating Values		Blured	Beta		Average Combinated
			$\lambda = 0$	$\lambda = 1$	
-1	-1	-	-	+	-
-1	0	-	+	+	-
-1	1	-	+	+	-
0	-1	+	-	+	-
0	0	+	+	+	+
0	1	-	+	+	-
1	-1	+	-	+	-
1	0	+	+	+	+
1	1	+	+	+	+

Tabla 3.1: Evaluación de ratings puntuales en fórmulas de métricas de reputación

el comportamiento de las tres métricas si se usan tres valores puntuales de rating.

II. **QoS Based Reputation:** para establecer el cálculo de esta reputación, se consideró necesario definir los parámetros QoS, los cuales fueron acordados en el SLA y que para este proyecto son: $Availability$, $Reliability$, $Performance$ y $Cost$.

Una vez establecidos los parámetros a ser medidos y dado que dichos QoS están en unidades distintas, buscamos una forma de obtener una única reputación resultado de la combinación de dichos QoS en base a una método conocido como Logic Scoring of Preferences**(LSP)** descrito en [27]. Dicho método nos permite construir una sola reputación en base a otros parámetros sin importar las unidades en las que están medidos dichos parámetros. Éste

método funciona de la siguiente manera: Primero se debe calcular el parámetro L, definido mediante la siguiente ecuación: donde E está definida en base

$$L = (|\omega_1|E_r^1 + |\omega_2|E_r^2 + ... + |\omega_n|E_r^n)^{1/r}; 0 \leq E \geq 1, \sum_{n=1}^{n} |\omega_i| = 1 \quad (3.6)$$

Ecuación 3.6: **Fuente:** Research report: Reputation system simulation results[32]

a tres métricas distintas, las cuales consideran el tipo de dato del parámetro QoS a evaluar:

a. Tipo de dato numérico: si el parámetro QoS puede ser expresado con un tipo de dato numérico, entonces la función de evaluación E queda definida por: Donde v_{max} representa el valor máximo de todos los servicios y v_{min} el

$$E = \begin{cases} 1 - \left(\dfrac{v_{max} - v}{v_{max} - v_{min}}\right) & \text{;si} \omega \geq 0 \\ \left(\dfrac{v_{max} - v}{v_{max} - v_{min}}\right) & \text{;para el resto} \end{cases} \quad (3.7)$$

Ecuación 3.7: **Tomado de** [32]

valor mínimo. v es el valor del servicio a evaluar. Nótese que se considera el peso(ω) del parámetro QoS a evaluar. El significado del peso(ω) es el siguiente: si el peso es igual a 1, entonces el criterio es requisito duro(**HARD**), lo que significa que los servicios que no cumplan con este criterio deben ser desechados. Si el peso(ω) es menor que 1 y mayor que 0, entonces el criterio se considera un requisito suave(**SOFT**) y se ejecuta el servicio.

b. Tipo de dato lógico: Si el parámetro QoS es de tipo de dato lógico, entonces la función de evaluación E queda definida por:

$$E = \begin{cases} 1 & \text{;si el parámetro QoS es met} \\ 0 & \text{;para el resto} \end{cases} \tag{3.8}$$

Ecuación 3.8: **Tomado de** [32]

c. Otros tipos de datos: Si el parámetro QoS es de tipo de dato definido por el usuario, entonces la función de evaluación E queda definida por:

$$E = \frac{(e_1 + e_1 + \ldots\ldots\ldots\ldots\ldots + e_n)}{n} \quad \text{donde } e \text{ es el valor de cada elemento} \tag{3.9}$$

Ecuación 3.9: **Tomado de** [32]

donde ω representa el peso de cada parámetro, y r es el valor lógico adoptado del método LSP, se optó por usar r=1. A continuación presentamos la definición de los parámetros QoS en la que se basa el presente proyecto (ver Tabla 3.2):

Parámetro	Definición
$Availability$	Este requisito está relacionado con la capacidad del servicio de responder las peticiones del usuario cuando este lo requiera. $Availability = \frac{TotaldelTiempodelServicio - TiempoDeServicioCaido}{TotaldelTiempodelServicio}$
$Reliability$	Este requisito está relacionado con la cantidad de servicios realizados con éxitos y el total de servicios ejecutados. $Reliability = \frac{\#serviciosExitosos}{totaldeServicios}$
$Performance$	Este requisito está relacionado con el monto de los recursos y el tiempo de ejecución de los servicios. En la Nube tiene que poseer un alto rendimiento por lo que a menor tiempo del servicio menor cantidad de recursos asignados en la ejecución de una petición.
$Cost$	Este requisito está relacionado con la cantidad que el usuario está dispuesto a pagar con el fin de estar provisto de un servicio que es compatible con todos los requisitos no funcionales adicionales.

Tabla 3.2: Definición de QoS basada en [25]

3.2 Consideraciones sobre el modelo

Para el presente modelo, se tiene las siguientes observaciones sobre sus componentes:

- Existen suficientes recursos en la Nube para satisfacer los requerimientos de servicio de los usuarios.

- Cada Broker gestiona con los Proveedores el alojamiento y ejecución de un servicio a la vez.

- Cada Proveedor define el precio del servicio independientemente del precio asignado por los otros proveedores.

3.3 Mecanismos de mercado

Se seleccionó dos mecanismos con la finalidad de poder contrastar los resultados para determinar que independientemente del mecanismo utilizado el sistema de reputación influye de manera significativa en la reducción de violaciones por parte del proveedor.

Por otra parte, con esta implementación durante el proceso de selección se consulta la reputación de los proveedores y si es mayor o igual a la mínima exigida, entonces el proveedor es considerado un candidato para alojar el servicio. Cada proveedor candidato realiza una oferta indicando el precio que está dispuesto a cobrar por el servicio solicitado. El precio es ponderado con un factor de penalización, f, el cual es calculado de la siguiente manera:

i. Se definen valores de penalización para cada uno de parámetros establecidos en el SLA, asignando los siguientes valores:.

   a. $penalty\ availability : 0{,}20$

   b. $penalty\ reliability : 0{,}20$

c. $penalty\ performance : 0{,}05$

$Availability$ y $Reliability$ reciben una penalización mayor por ser parámetros de tipo ***HARD*** lo cual indica que para el usuario dichos parámetros tienen mayor relevancia y por ende para nuestro modelo se considero que tengan una mayor ponderación. El parámetro QoS $Cost$ no tiene penalización pues es un valor que permance constante durante la simulación. Estos valores constituyen lo que hemos denominado una **Matriz de Penalización**

$$p = [penalty\ availability,\ penalty\ reliability,\ penalty\ performance] \tag{3.10}$$

ii. Se calculan los valores actuales otorgados por el proveedor para availability, reliability y performance denotados por: availabilityProvider, reliabilityProvider y performanceProvider respectivamente.

iii. Sean availabilitySLA, reliabilitySLA y performanceSLA los valores para availability, reliability y performance respectivamente definidos en el SLA

iv. Posteriormente, se procede a calcular el vector de penalización :

$$vp = \begin{bmatrix} a \\ r \\ p \end{bmatrix} \tag{3.11}$$

Donde :

$$Si\ availabilityProvider > availabilitySLA\ entonces\ a = 0\ sino\ a = 1 \tag{3.12}$$

$$Si\ reliabilityProvider > reliabilitySLA\ entonces\ r = 0\ sino\ r = 1 \tag{3.13}$$

$$Si\ performanceProvider < performanceSLA\ entonces\ p = 0\ sino\ p = 0 \tag{3.14}$$

v. Siendo f el factor de penalización :

$$f = p * vp \tag{3.15}$$

CAPÍTULO 4

4. Diseño del simulador

Durante la etapa de diseño del simulador se acordó por conveniencia de los autores del presente trabajo adoptar el idioma inglés para definir todos los términos que se usarían en dicha etapa y cuando se implementará el código del simulador.

4.1 Diagrama de casos de uso

En la implementación del simulador se consideraron como actores a: modeller[1] y tester[2]. A continuación describiremos brevemente los casos de uso considerados y las interacciones entre los usuarios (modeller y tester) con la herramienta de

[1] Término en inglés usado para identificar a quién modela la simulación.
[2] Término en inglés usado para identificar a quién realiza pruebas en el simulador.

simulación que denominamos Cloud Simulation Tool[3].

I. **Nombre del caso de Uso**

Configure Scenario(Configurar escenario)

Actor Principal

Modeller

Descripción

El actor define los parámetros necesarios para la simulación.

Flujo Principal

a. El actor ingresa la información de Usuarios.

b. El actor ingresa la información de Proveedores.

c. El actor ingresa la información de Servicios.

d. El actor selecciona las métricas para la reputación.

e. El actor selecciona el mecanismo de mercado.

Flujo Alterno

Si los datos no corresponden no tienen valores válidos se muestran mensajes

Poscondiciones

[3] En español significa herramienta de simulación de Nube

Ninguna

II. **Nombre del caso de Uso**

Simulate Scenario(Simular escenario)

Actor Principal

Modeller, Tester

Precondiciones

El actor debería haber configurado el escenario con la información necesaria descrita en el caso anterior.

Objetivo

Realizar la simulación de una Nube basada en un mecanismo de mercado utilizando un sistema de reputación para el alojamiento de servicios.

Flujo Principal

a. El actor inicia la simulación pulsando el botón Run Simulation[4].

b. El actor continúa con las rondas de simulación pulsando el botón Continue Simulation[5].

Poscondiciones

Se generan logs y charts para visualizar el proceso de simulación y los resultados respectivamente.

[4]En español quiere decir Ejecutar Simulación
[5]En español quiere decir Continuar Simulación

III. **Nombre del caso de Uso**

Export Scenario(Exportar escenario)

Actor Principal

Modeller.

Precondiciones

Ninguna.

Objetivo

El usuario puede exportar en un archivo XML los datos utlizados para la simulación para su posterior uso.

Flujo Principal

a. El actor pulsa el botón File.

b. El actor selecciona la opción export.

c. El actor ingresa el nombre y la ubicación donde desea exportar el archivo.

Poscondiciones

Se guarda el escenario en la ubicación seleeccionada.

IV. **Nombre del caso de Uso**

Import Scenario(Importar escenario).

Actor Principal

Modeller, Tester

Precondiciones

Debe existir un archivo XML válido de un escenario para poder importarlo.

Objetivo

Comprobar los resultados obtenidos durante una simulación usando datos importados.

Flujo Principal

a. El actor pulsa el botón File.

b. El actor selecciona la opción Import.

c. El selecciona el archivo y pulsa el botón Abrir.

Poscondiciones

El escenario es cargado correctamente en el simulador.

En la figura 4.1 se exponen las relaciones de los diferentes casos de uso descritos previamente.

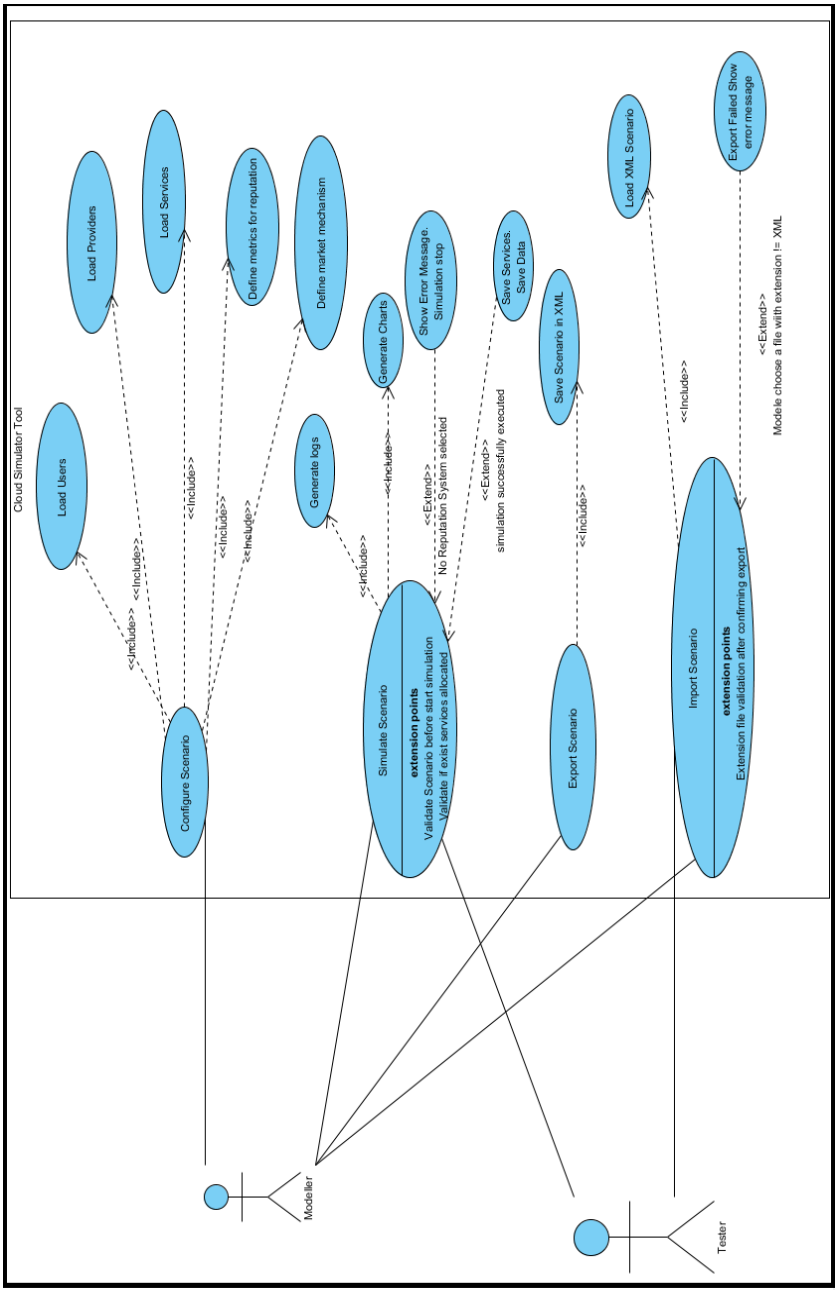


Figura 4.1: Casos de uso del simulador

4.2 Diagrama de clases

La arquitectura de nuestro simulador Cloud Simulation Tool extiende de CloudSim cuya arquitectura fue descrita en la sección 2.5 donde se describieron de forma breve las principales clases de dicho software. En esta sección pondremos a consideración las clases que fueron implementadas luego de extender de CloudSim. La figura 4.2 presenta el diagrama de clases que resume las clases principales y sus diferentes relaciones, las cuales fueron utilizadas para modelar la implementación del presente simulador. En las figuras 4.3 y 4.4 se detalla los atributos y procedimientos de dichas clases. A continuación se describe brevemente las clases principales.

- **ReputationSystem:** clase encargada de evaluar la reputación de los proveedores. El componente puede ser Metric Based Reputation o QoS Based Reputation.

- **Trading Mechanism:** esta clase permite administrar los parámetros a ser utilizados por los demás componentes en el proceso de selección y alojamiento del servicio en la nube

- **SLA:** clase que permite definir los indicadores de calidad o parámetros QoS comúnmente asociados a requerimientos no funcionales: availablity, reliabi-

lity, perfomance y cost.

- **Service:** esta clase define el servicio a solicitar por parte del usuario. Establece la cantidad de MIPS a ejecutarse en el proveedor y la mínima reputación que debe tener un proveedor.

- **User:** clase que define el servicio a solicitar y el SLA con los requerimientos no funcionales que deben cumplirse por parte del proveedor.

- **Broker:** esta clase actúa como intermediario entre el usuario y el proveedor. Gestiona la elección del mejor proveedor y una vez ejecutada la tarea, notifica al usuario para que califique al proveedor.

- **Provider:** clase que permite simular el comportamiento del propietario de los recursos.

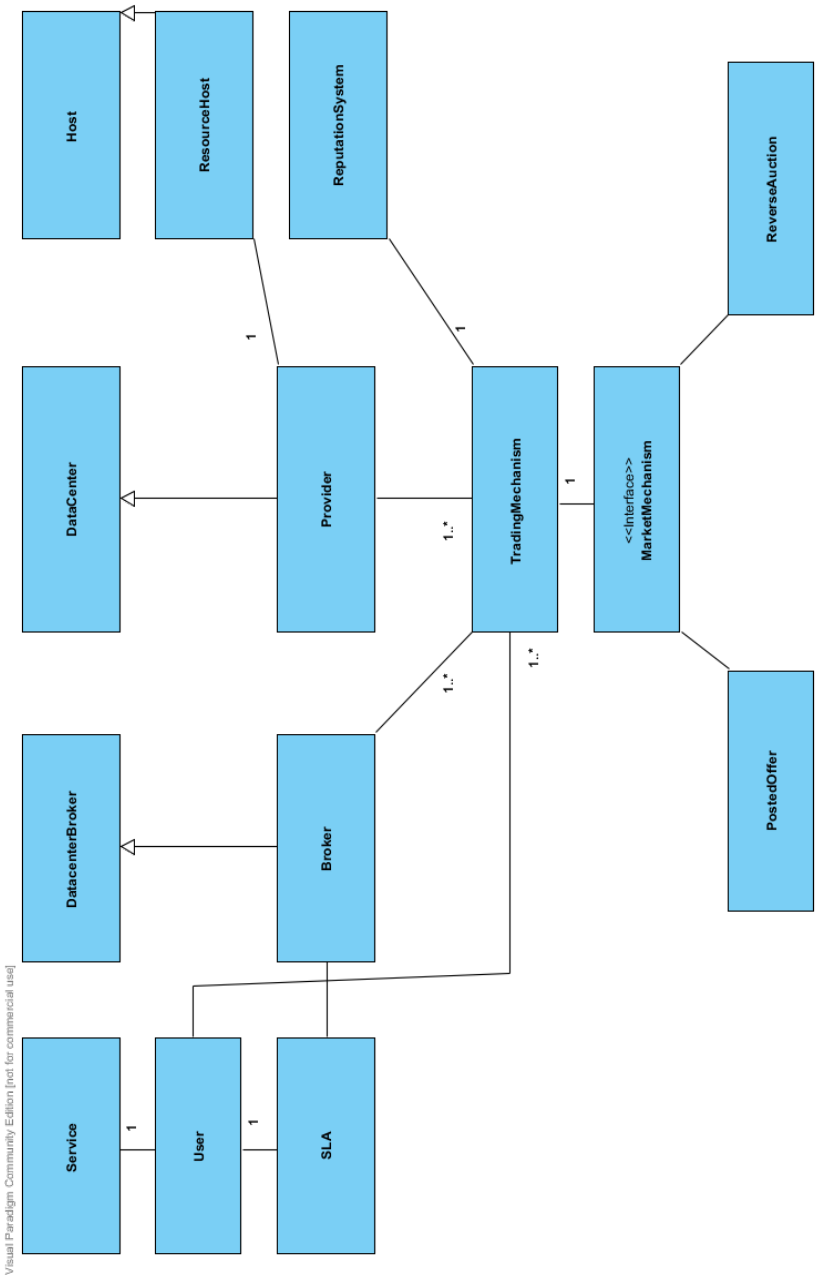


Figura 4.2: Diagrama de clases del simulador

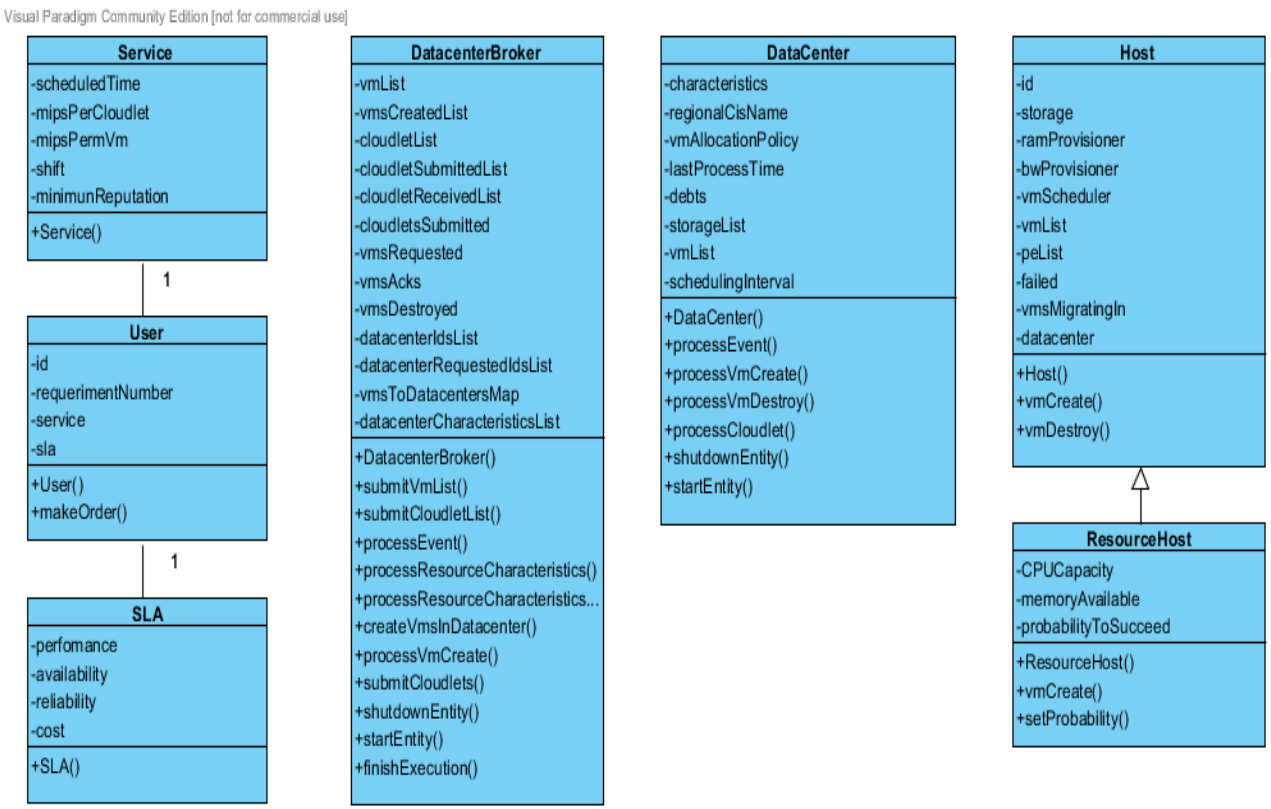


Figura 4.3: Diagrama de clases del simulador - 1

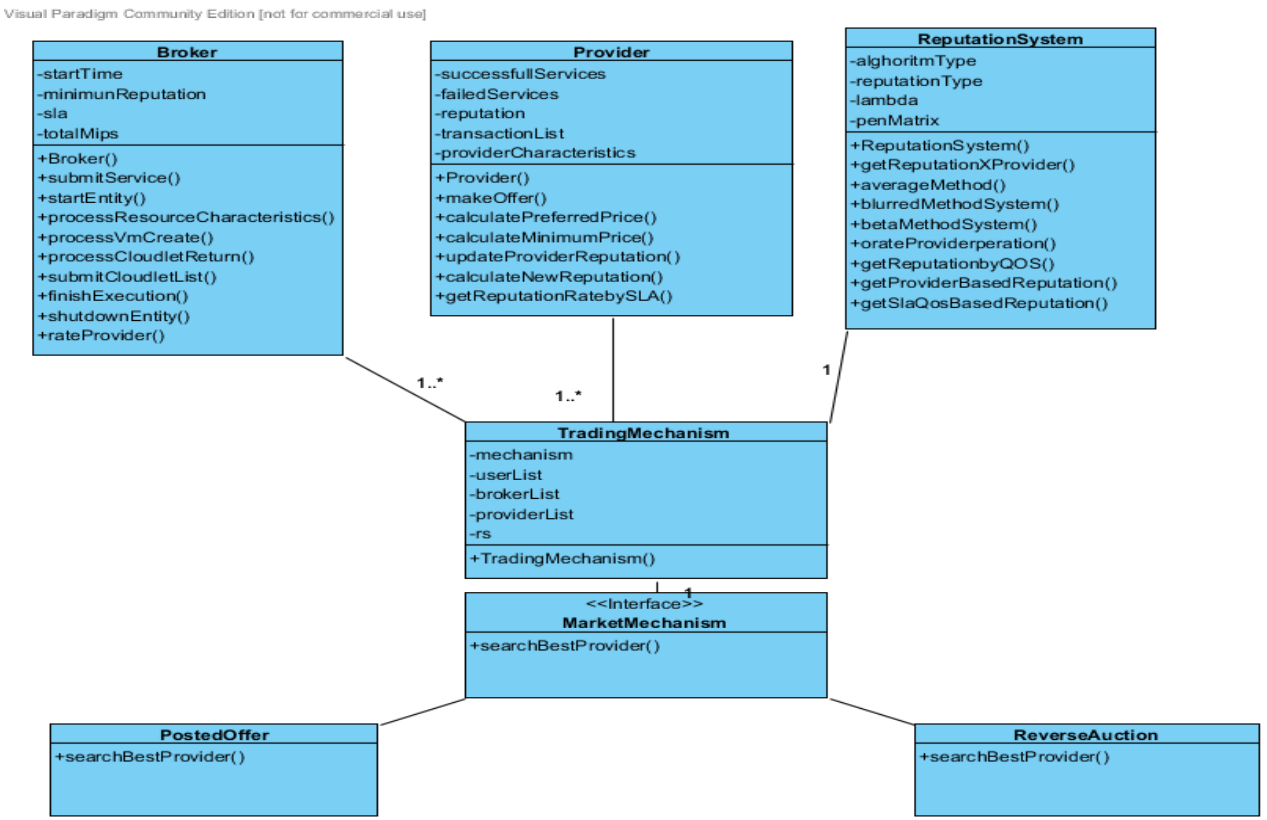


Figura 4.4: Diagrama de clases del simulador - 2

4.3 Diagrama de componentes

Los principales componentes que forman parte del diseño del presente simulador son mostrados en la figura 4.5 a continuación:

- **Simulator:** componente principal implementado por nosotros, el cual se comunica con los otros tres componentes y es el encargado de llevar a cabo todo el proceso de simulación.

- **XMLLoader:** este componente usa **DOM** (Document Object Model por sus siglas en inglés). Nos permite manejar una estructura para los archivos utilizados en la herramienta de simulación. Los escenarios que contienen la información de proveedores, usuarios y servicios a solicitar son guardados por medio de este componente en archivos XML. De igual manera los resultados de la simulación, los servicios por proveedor que fueron alojados exitosamente o no también son guardados utilizando este componente. El formato utilizado, XML, permite una fácil lectura del contenido y en el caso de los resultados, el posterior analisis en R para estos archivos no resultó complicado.

- **CloudSim:** según lo revisado anteriormente, es el componente que nos permite realizar la simulación.

- **ResultsGenerator:** componente encargado de manejar la generación de los resultados de la simulación, los mismos que se muestran en logs que registran los eventos ocurridos en la simulación. Pero, para generar reportes que permitan una mejor visualización de los mismos y un análisis más rápido se utilizó JFreeChart durante la implementación de este componente.

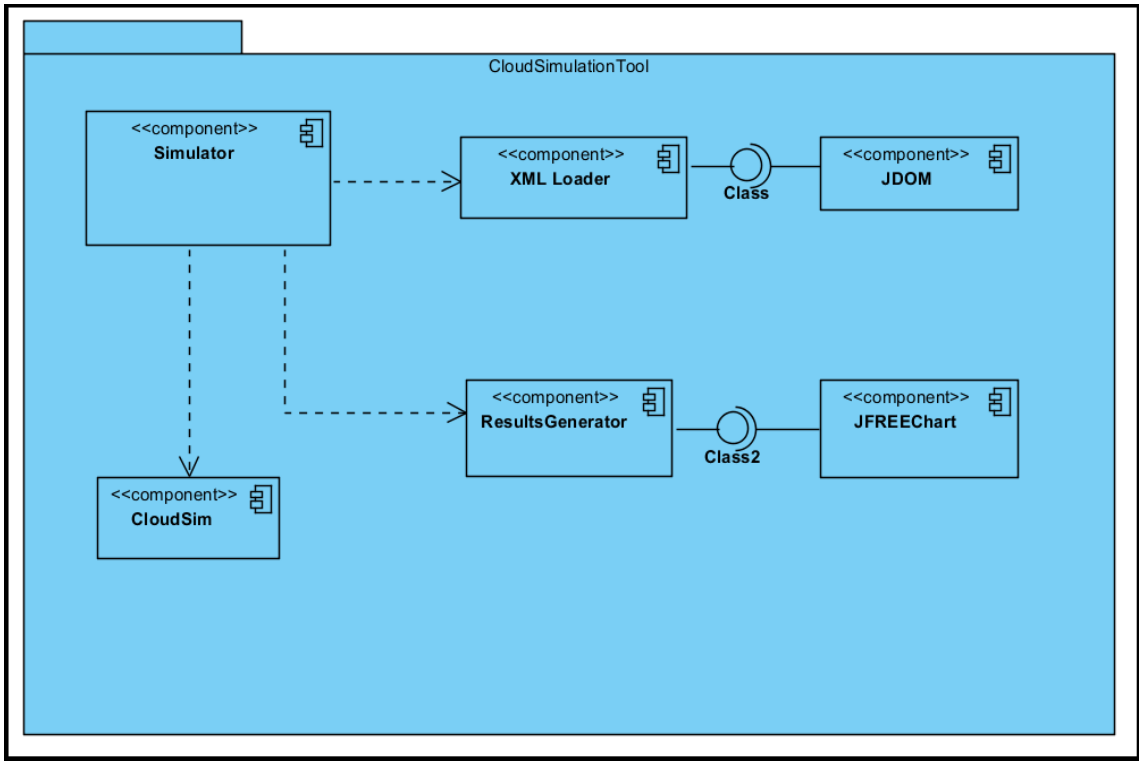


Figura 4.5: Diagrama de componentes del simulador

4.4 Diagrama de secuencias

Para mostrar la secuencia de interacciones de los mecanismos de mercado escogidos, se expone a continuación a través de los diagramas de secuencia corres-

pondientes a ambos mecanismos; el comportamiento de cada uno de ellos.

I Reverse Auction: las interacciones que suceden en un mercado que trabaja con ese mecanismo se ilustran en la figura 4.6.

II Posted Offer: se ilustran en la figura 4.7.

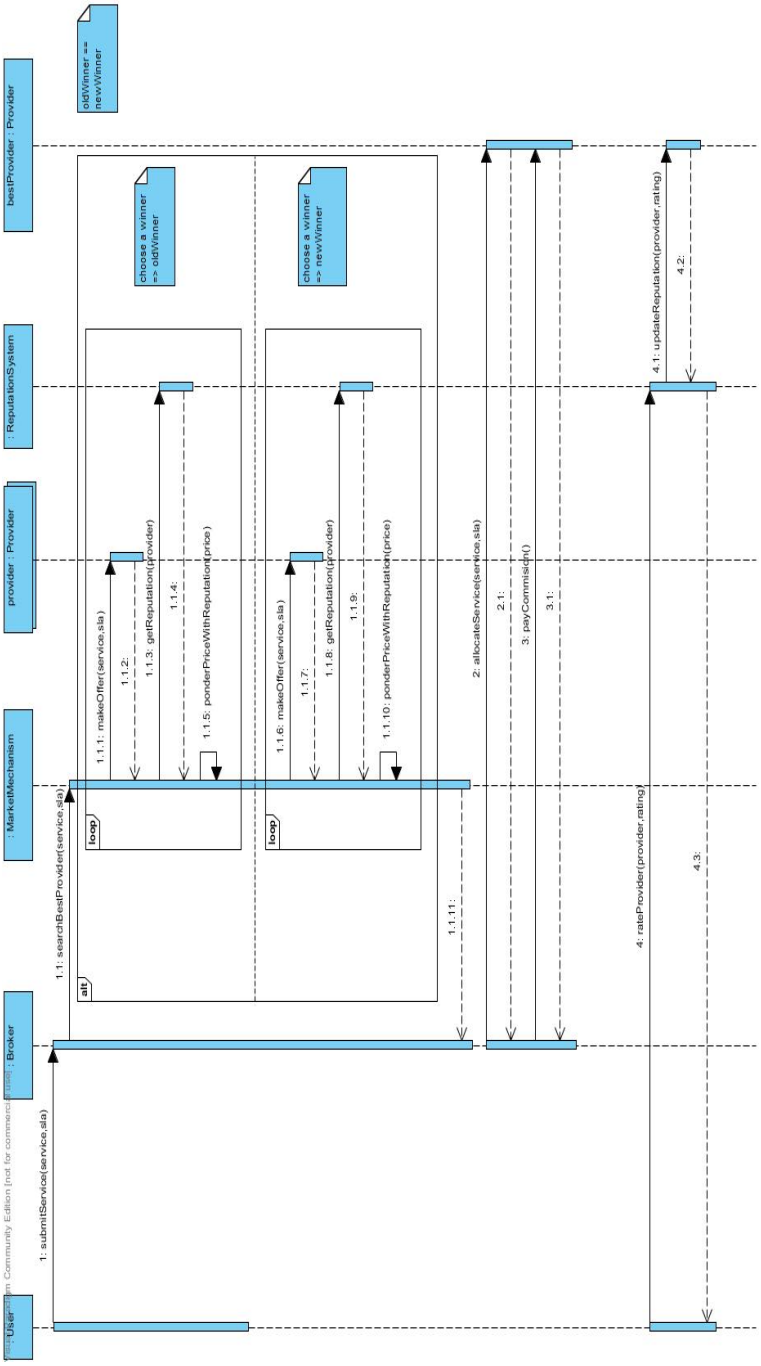


Figura 4.6: Diagrama de secuencia del mecanismo de mercado **Reverse Auction**

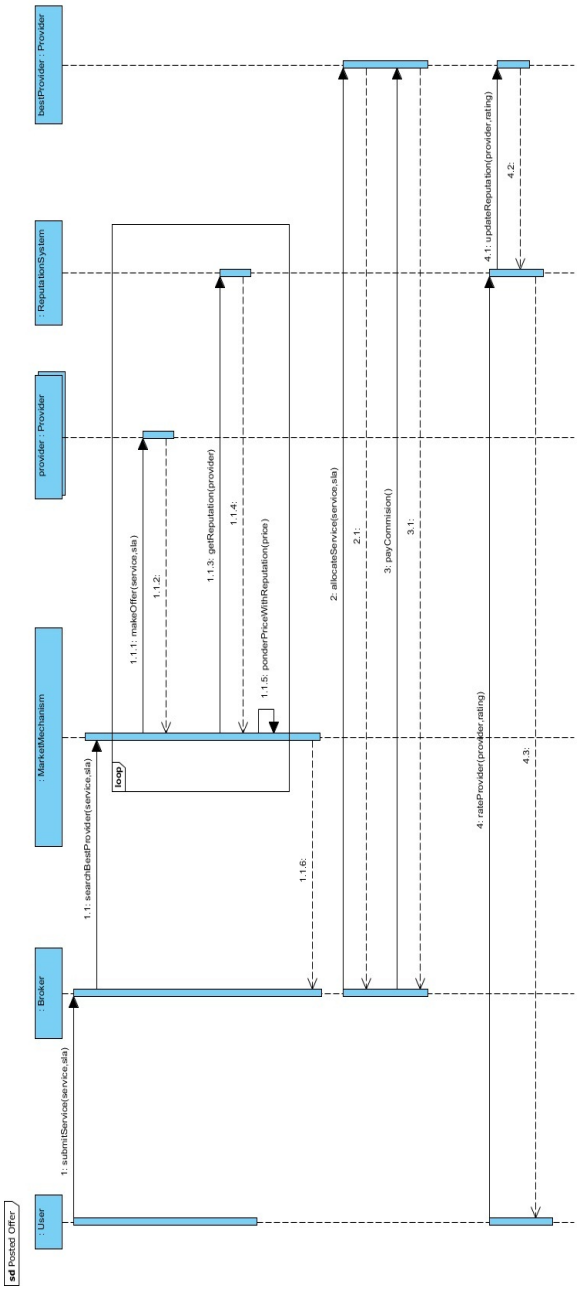


Figura 4.7: Diagrama de secuencia del mecanismo de mercado **Posted Offer**

4.5 Diagramas de despliegue

Con la finalidad de ilustrar el proceso de despliegue del simulador, se muestra en la figura 4.8 el correspondiente diagrama de despliegue:

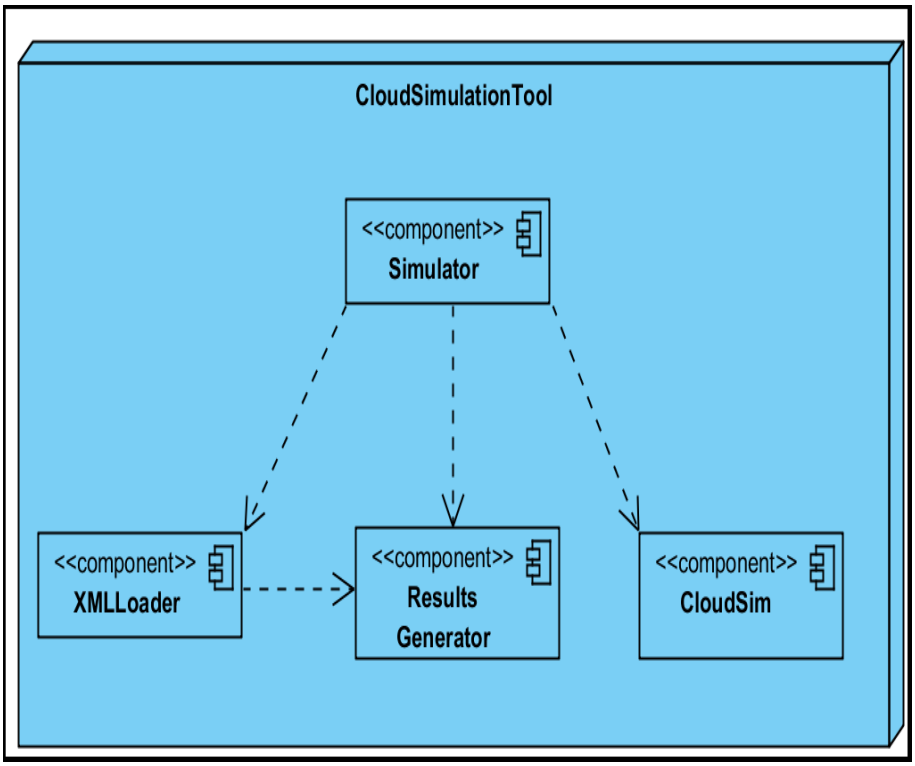


Figura 4.8: Diagrama de despliegue del simulador

Se puede observar que sólo existe un nodo principal y es el nodo de ejecución, el mismo que se encarga de invocar los componentes necesarios para llevar a cabo la simulación.

CAPÍTULO 5

5. Implementación y pruebas del simulador

5.1 Descripción del algoritmo

En la presente sección se describe el algoritmo utilizado por el simulador. Con la finalidad de que éste quede mejor ilustrado, se ha dividido en secciones, las cuales se muestran a continuación:

Code 5.1: Init Scenario

```
1    nRounds //number of rounds for the simulation
2    nUsers //number of Users
3    nServices //number of Services
4    nProviders //number of Providers
5    mechanism //market mechanism
6    reputationSystem //metric used to calculate provider reputation
```

Code 5.2: Round Simulate

```
7    For: i=1 to nRounds
8        CloudSim.initializeVariables() //init CloudSim package
9        userList // user list
10       providerList // provider list
11       serviceList // service list
12       tradingMechanism // negotiation mechanism
13       tradingMechanism.set(mechanism,reputationSystem)
```

Code 5.3: Load data

```
14       loadOrGenerateProviderData(nProviders)
15       loadOrGenerateUserData(nUsers,nServices)
```

Code 5.4: Make order

```
16       For: all users u in userList
17           broker = new Broker(u)
18           broker.submitService(u.getService(),u.getSLA())
19           tradingMechanism.addBrokerToList(broker)
20       EndFor
```

Code 5.5: Process Request

```
21       CloudSim.startAuction();
22       While: remain service requests
23           service = getActualRequest()
24           providerRequestedList <- provider requested list
25           bestProvider <- best Provider
26           preferredPriceToBuy = tradingMechanism.getEstimatedCostMIPS(service.
                 getTotalMips());
27           Do   :
```

Code 5.6: Search Best Provider

```
28              For: all providers p in providerList
29                  reputation = p.getReputation()
30                  totalMips = service.getTotalMips()
31                  minimumReputation = service.getMinimumReputation()
32                  sla = service.getSla()
33                  If:reputation is greater than  minimumReputation and p no exist
                        in providerRequestedList
34                      bid = p.makeOffer(totalMips,preferredPriceToBuy)
35                      increaseFactor = getFactor(sla,p)
36                      ponderedBid = ponderPriceWithReputation(bid,increaseFactor)
37                      If: ponderedBid is better than others
38                          bestProvider = p
39                      EndIf
40                  EndIf
41              EndFor
```

Code 5.7: Rate Provider

```
42              If  : bestProvider exist
43                  isAllocated = bestProvider.allocateService(service)//True if
                        service is alocated
44                  rating = user.rateService(isAllocated,service)
45                  reputationSystem.rateProvider(bestProvider,rating,service)
46                  providerRequestedList.add(bestProvider)
47              EndIf
48          While(isAllocated == FALSE And bestProvider==NULL And
                providerRequestedList.size() < providerList.size() )
49          If : bestProvider == NULL
50              print("No provider Available");
51          EndIf
52      EndWhile
```

Code 5.8: Save Status

```
53          saveServiceStatus(serviceList)
54          saveProviders(providerList)
55          printProviderStatus(providerList)
56      EndFor
```

- **Init Scenario:** inicialmente se define el número de rondas que llevará a cabo la simulación, el número de proveedores a crear en la Nube, el número de usuarios que realizarán las solicitudes de servicio y la cantidad de servicios que van a ser solicitados para alojarse en la Nube. Luego, procedemos a definir el mecanismo de mercado a utilizar en el proceso de negociación. El mecanismo puede ser: Posted Offer o Reverse Auction. A continuación, definimos la métrica que será utilizada para calcular la reputación del proveedor ya sea Beta, Blurred, Average o QoSBased.

- **Round Simulate:** una vez definidos estos valores, el proceso de selección de proveedores y alojamiento de servicios se realiza por rondas. Se inicializa el componente CloudSim. Además se definen listas para usuarios, proveedores y servicios de manera que se tenga acceso a la información y estado de estas entidades durante el transcurso de la simulación. Se define el mecanismo de negociación y el mecanismo de mercado.

- **Load Data:** se realiza la carga de información de usuarios y proveedores.

En el caso de la primera ronda de simulación, los datos son generados de manera aleatoria.

- **Make order:** a continuación los usuarios realizan la petición de alojamiento de servicios. Para cada petición se define un Broker. Como ya se mencionó anteriormente, un broker actúa de intermediario entre el usuario y el proveedor.

- **Process Request:** se busca el mejor proveedor para alojar el servicio. Se realiza el cálculo de la cantidad que el Broker está dispuesto a pagar por la cantidad de MIPS que requiere.

- **Search Best Provider:** cada proveedor que cumpla con la reputación mínima exigida por el broker realiza una oferta en base a la cantidad de MIPS solicitados y la cantidad dispuesta a pagar por el Broker. Se genera un factor de penalización que es calculado tomando en cuenta los servicios alojados anteriormente y el cumplimiento o no con los paramentos: availablity, reliability, perfomance y cost definidos en el SLA. La oferta realizada por el proveedor es ponderada con la penalización generada. Si existe un proveedor que cumpla con los requerimientos establecidos en el SLA, la reputación mínima exigida y la oferta del proveedor es mejor que la de los otros proveedores, el proveedor es seleccionado y el Broker espera que el servicio sea alojado.

- **Rate Provider:** si existe un proveedor que cumpla con los requerimientos establecidos en el SLA, la reputación mínima exigida y la oferta del proveedor es mejor que la de los otros proveedores, el proveedor es seleccionado y el Broker espera que el servicio sea alojado. Al terminar la ejecución del Job, el Broker notifica al usuario que el proceso ha finalizado. El usuario califica al proveedor con un valor de 1 si fue exitoso o de -1 si el servicio solicitado no fue exitoso. Luego de esto se actualiza la reputación del proveedor tomando en cuenta el último registro realizado.

- **Save Status:** luego de cada ronda la información de los proveedores es guardada y utilizada en la siguiente ronda. De esta manera cada proveedor guarda un historial de los servicios alojados durante la simulación.

5.2 Características del simulador

5.2.1 Configuración de los escenarios

Para realizar la simulación se requiere seleccionar el mecanismo de mercado junto con el sistema de reputación y configurar atributos de proveedores y servicios. Se puede seleccionar uno o más mecanismos de mercado y de la misma manera uno o más sistemas de reputación. (ver figura 5.1)

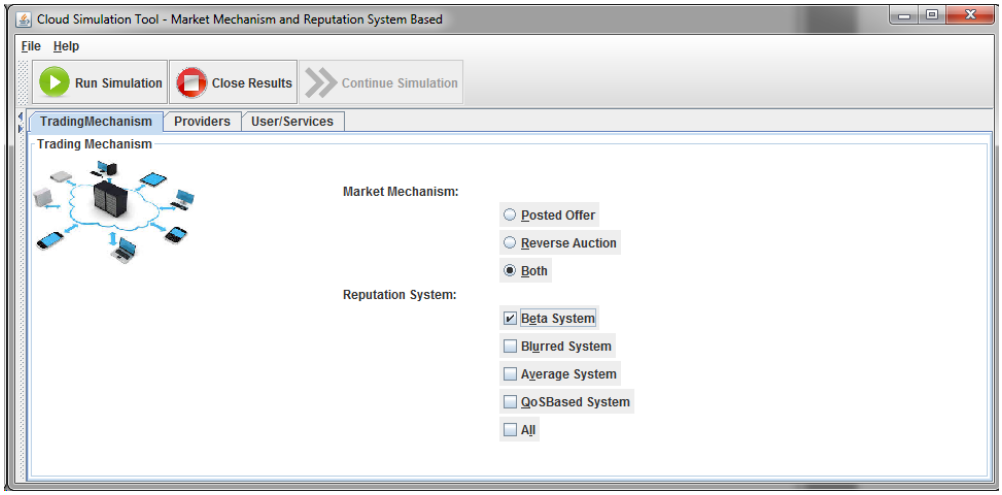


Figura 5.1: Configuración de Mecanismo de Mercado y Sistema de Reputación

Un proveedor representa el DataCenter en la Nube en el cual se alojarán los servicios. Los valores de availability, reliability y perfomance son atributos del proveedor que asegura se cumplirán al alojar un servicio. La configuración de estos valores en el simulador se muestra en la figura 5.2. Los atributos requeridos para ésta entidad son los siguientes:

- **Name:** identificador del proveedor

- **Initial Reputation:** la reputación inicial del proveedor. Inicialmente se define una reputacion de 50 para todos los proveedores, a fin de garantizar que en las primeras rondas todos los proveedores participen en el proceso de selección. El valor puede ser cambiado según el escenario que se quiera simular.

- **Preferred Price:** indica el valor por MIPS que el proveedor está dispuesto

a cobrar. Puede tomar un valor entre 0.3 y 1. Los valores de 0.3 y 1 son definidos para que todos los proveedores tengan el mismo rango de precios para ofertar, garantizando la competitividad y disminuyendo la posibilidad de que el proveedor oferte una cantidad muy baja que conlleve una perdida.

- **Max Price:** indica el máximo precio por unidad de MIPS que el proveedor cobraría. Puede tomar un valor entre 0.3 y 1. Los valores de 0.3 y 1 son definidos para que todos los proveedores tengan el mismo rango de precios para ofertar, garantizando la competitividad y disminuyendo la posibilidad de que el proveedor oferte una cantidad muy baja que conlleve una perdida.

- **Mips per Core:** representa la capacidad de procesamiento por núcleo con las que cuenta el proveedor para alojar los servicios.

- **Probability to Succeed:** establece la probabilidad de que el proveedor aloje un servicio de manera exitosa. Puede tomar el valor de 0 a 1.

- **Availability:** puede tomar un valor entre 0 y 1. Indica el porcentaje que el proveedor estará disponible cuando se realicen requerimientos de servicio.

- **Reliability:** puede tomar un valor entre 0 y 1. Define la relación entre el número de servicios alojados exitosamente y el total de servicios alojados. Es decir, a mayor valor, mayor será el número de servicios que fueron alojados

de manera exitosa.

- **Perfomance:** Está relacionado con el monto de los recursos y el tiempo de ejecución de los servicios. A menor tiempo del servicio menor cantidad de recursos serán asignados en la ejecución de una petición.

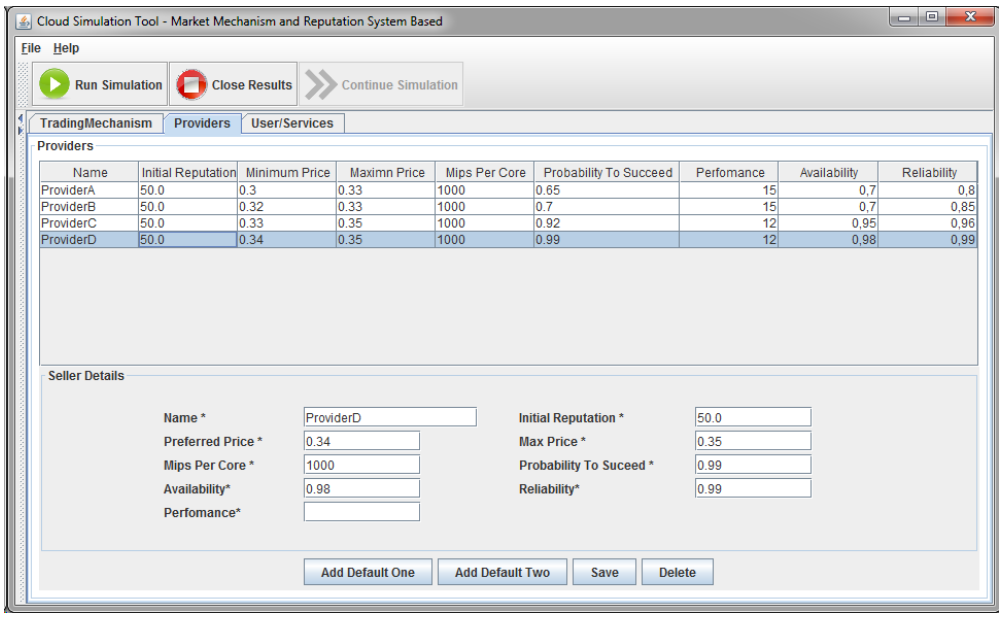


Figura 5.2: Configuración de Proveedores

Los valores y pesos de los parámetros definidos en el SLA para los servicios van de 0.0 a 1.0. Los pesos se definen de acuerdo a la prioridad o preferencia que se desee dar a cada parámetro. Los pesos y valores de los parámetros QoS son utilizados en el cálculo de QoS Based Reputation. La configuración de estos valores en el simulador se muestra en la figura 5.3. Los atributos de los servicios que deben ser definidos son los siguientes:

- **Id:** identifica al usuario

- **Number of Requirements:** define el número de servicios a solicitar por el usuario.

- **Scheduled Time:** indica el tiempo en el que los servicios serán solicitados en la nube.

- **Mips per Job:** la cantidad de MIPS que el usuario solicita al Proveedor sea alojado en la Nube.

- **Mips per VM:** la cantidad de MIPS que una máquina virtual del proveedor puede alojar.

- **Min. Reputation:** la reputación mínima que un proveedor debe tener para ser considerado como un candidato para alojar el servicio.

- **Preferred Cost:** el costo por unidad de MIPS que el usuario está preferentemente dispuesto a pagar para que el servicio solicitado sea alojado en la Nube.

- **Maximum Cost:** el costo máximo que está dispuesto a pagar el usuario por unidad de MIPS.

- **Cost Weight:** el peso que se asigna al precio acordado, *Maximum Cost*,

establecido como parámetro QoS en el SLA.

- **Availability value:** el valor de disponibilidad establecido como parámetro QoS en el SLA y que se espera sea cumplido por el proveedor.

- **Availability weight:** el peso asignado en el SLA a *Availability value*.

- **Reliability value:** el valor de confiabilidad establecido como parámetro QoS en el SLA y que se espera sea cumplido por el proveedor.

- **Reliability weight:** el peso asignado en el SLA a *Reliability value*.

- **Perfomance value:** el valor de rendimiento establecido como parámetro QoS en el SLA y que se espera sea cumplido por el proveedor.

- **Perfomance weight:** el peso asignado en el SLA a *Perfomance value*

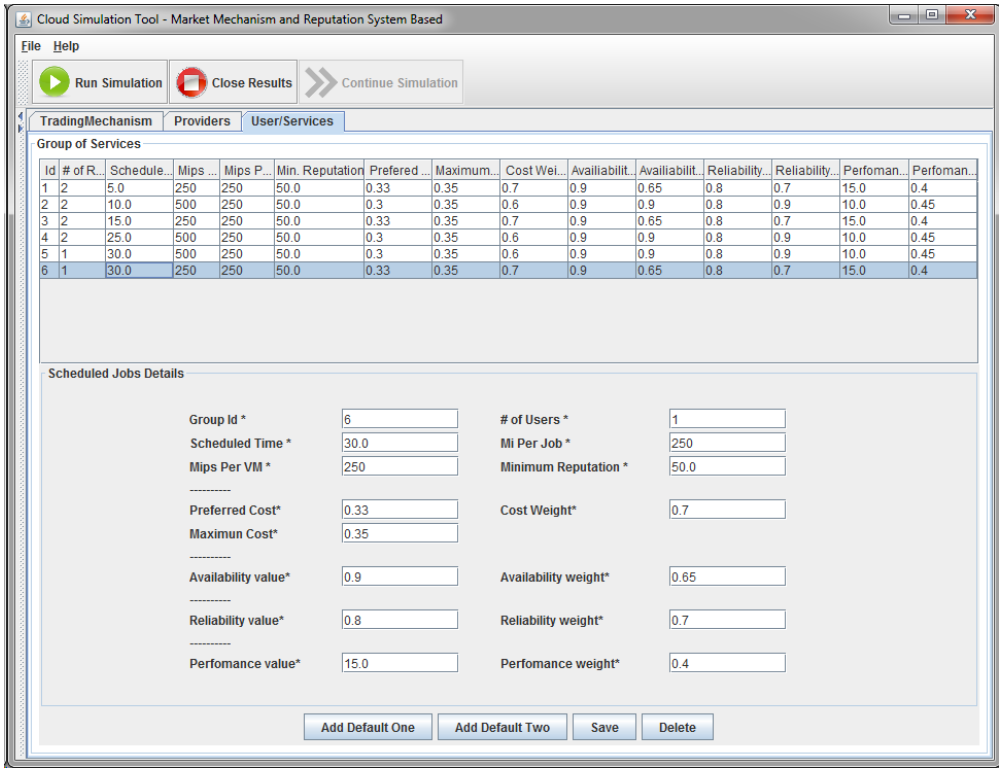


Figura 5.3: Configuración de Servicios

5.2.2 Simulación de escenarios

Luego de seleccionar los mecanismos de mercado, los sistemas de reputación, definir datos para proveedores y definir valores de los servicios, se procede a simular el alojamiento de servicios en los proveedores. Existe la opción de continuar la simulación, esto permite solicitar el alojamiento de servicios nuevamente pero con la información de los proveedores, reputación y lista de servicios solicitados actualizada en base a los resultados de la última simulación.

En la sección de **Anexos** (ver Anexo B) se encuentran los datos de servicios y

proveedores utilizados en la simulación.

5.2.3 Reportes de simulación

Para realizar un análisis de los resultados y observar el comportamiento de los distintos elementos durante la simulación se generó gráficos utilizando el componente JFree-Chart. Los gráficos que muestran los resultados son:

- Porcentaje de servicios alojados: el gráfico muestra el porcentaje de servicios alojados por proveedor luego de la simulación.

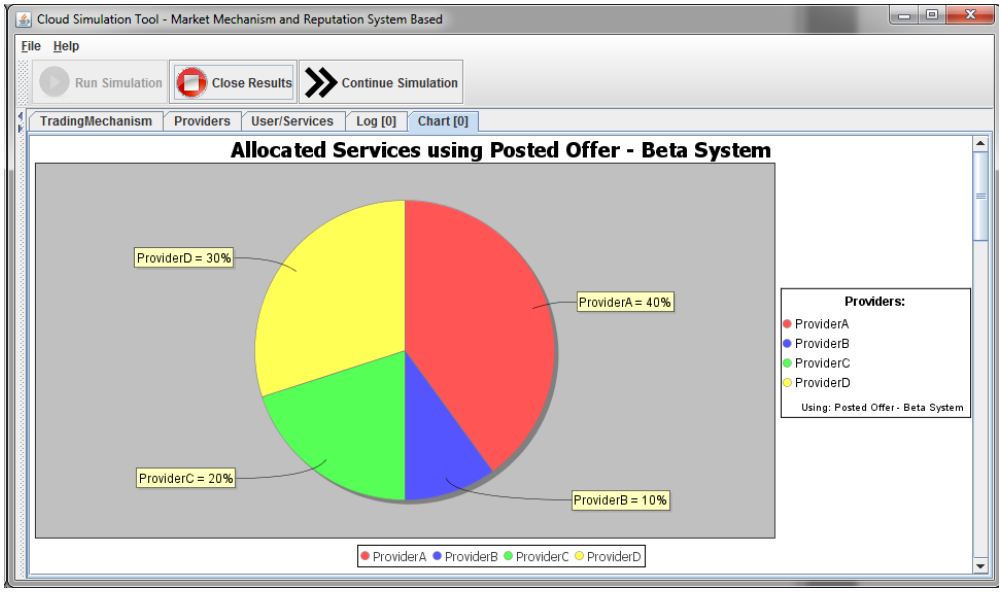


Figura 5.4: Porcentaje de servicios alojados

- Número de violaciones cometidas por los proveedores.

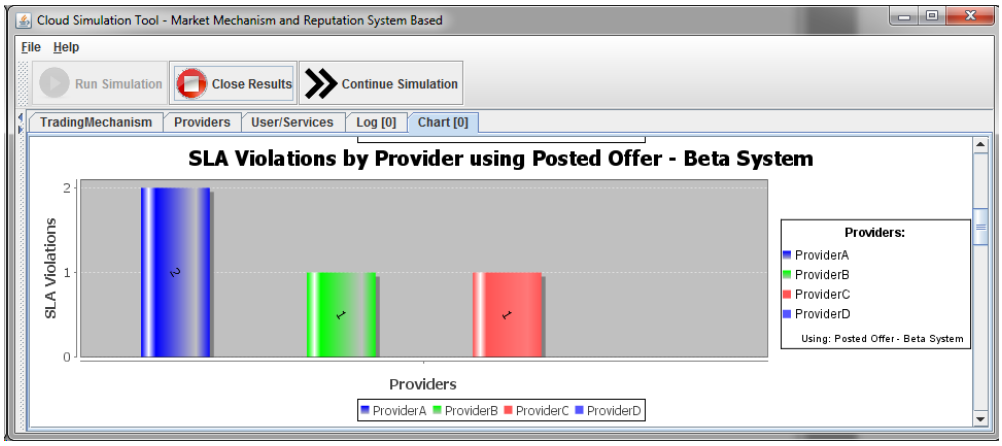


Figura 5.5: Violaciones cometidas por proveedor

- Reputación de proveedores

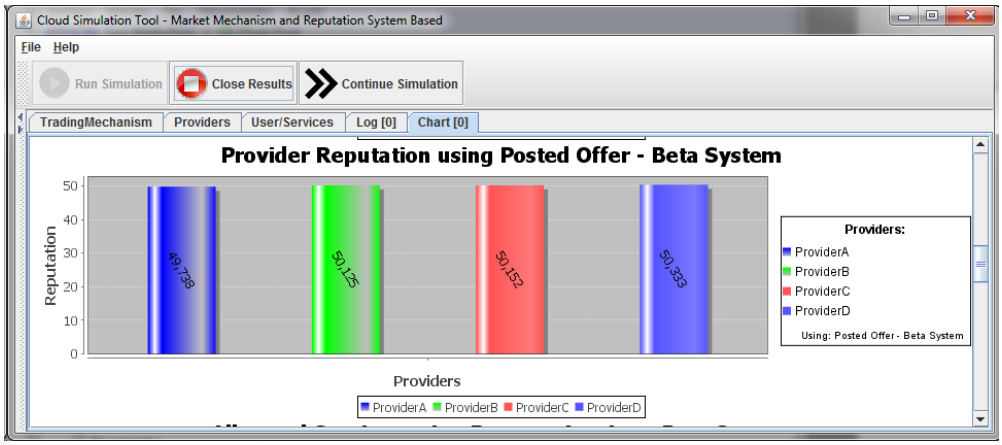


Figura 5.6: Reputación de proveedores

- Registro de eventos de la simulación

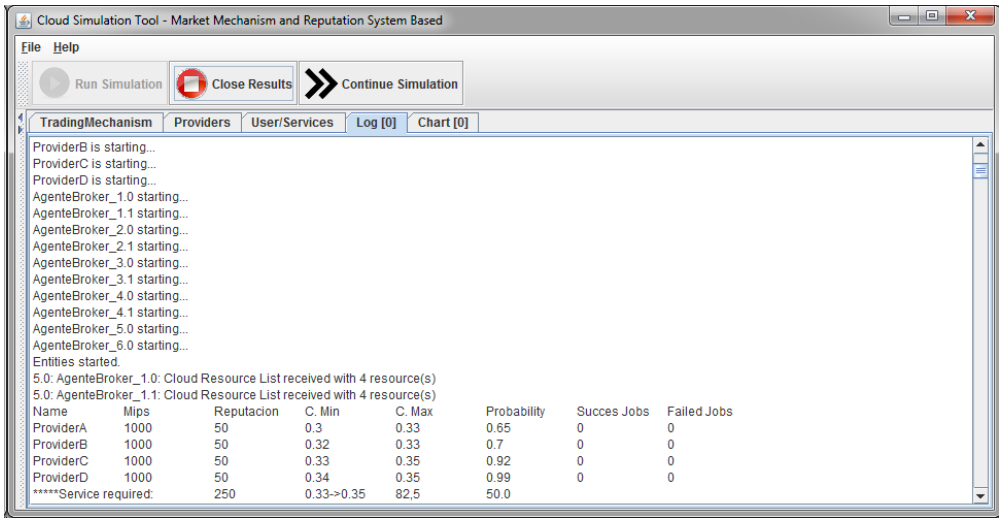


Figura 5.7: Registro de eventos ocurridos en la simulación

CAPÍTULO 6

6. Análisis de los Resultados

En el presente capítulo definiremos los resultados esperados durante el proceso de simulación. Posteriormente, detallaremos el escenario utilizado durante la misma. Luego, mostraremos los resultados obtenidos para finalmente realizar un análisis de éstos.

6.1 Resultados esperados

En la etapa de desarrollo del presente proyecto nos enfocaremos principalmente en mitigar las violaciones en los parámetros QoS de los SLAs, con lo cual esperaríamos obtener como resultado de la simulación una disminución en el número

de violaciones por proveedor independientemente del tipo de mecanismo de mercado utilizado durante la misma,bien sea este **Posted Offer o Reverse Auction**. Adicionalmente, se esperaría que aquellos proveedores con una reputación más alta reciban una mayor asignación de servicios que los proveedores que poseen una reputación más baja. De esta manera nos aseguraríamos el cumplimiento del objetivo principal de: disminuir de manera significativa las violaciones en los SLAs.

El proceso que se llevará a cabo en esta etapa es el siguiente:

I. Se deberá seleccionar un mecanismo de mercado y un sistema de reputación.

II. La simulación será realizada nuevamente, pero esta vez con los datos de los proveedores actualizados de acuerdo a la última ejecución realizada.

III. La reputación será modificada al finalizar cada ejecución.

IV. Se deberá repetir todo este proceso varias veces.

Como resultado de este proceso, se esperaría una disminución en el número de violaciones proporcional al aumento del número de simulaciones realizadas.

Finalmente, para realizar el procesamiento de los resultados, obtener el número de violaciones ocurridas por simulación y generar gráficos que permitan analizar los resultados se utilizará el programa estadístico **R**. Adicionalmente, se esperaría que

el gráfico de violaciones **vs.** sistema de reputación para el mecanismo de mercado seleccionado sea similar al siguiente:

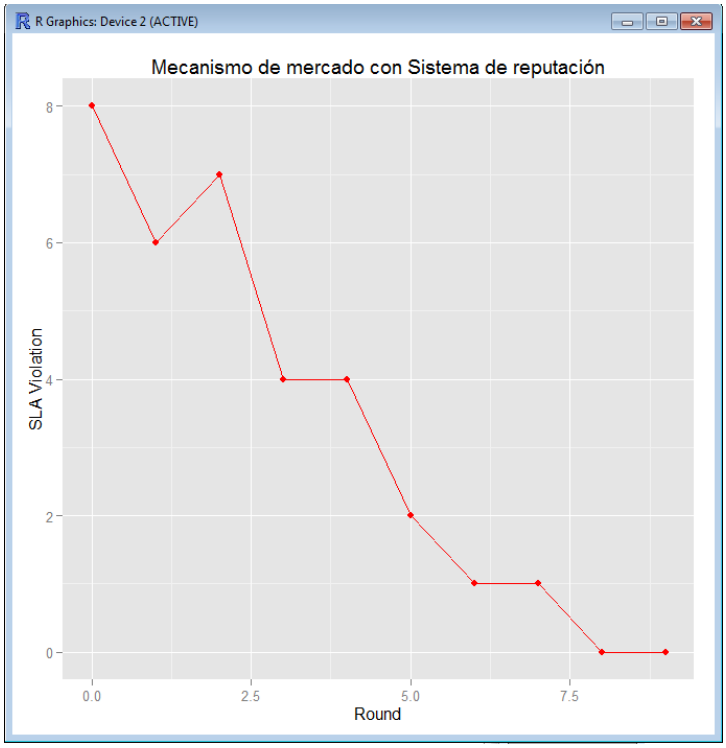


Figura 6.1: Violaciones para el mecanismo de mercado con Sistema de Reputación

6.2 Resultados obtenidos

Durante la simulación se establecieron las combinaciones de mecanismos de mercado y sistemas de reputación que se muestran en la tabla 6.1, la misma que nos permite observar una combinación entre los dos tipos de mecanismos de mercado y los cuatro sistemas de reputación escogidos en el presente proyecto.

Mecanismo de Mercado	Sistema de Reputación
Posted Offer	Beta System
Posted Offer	Blurred System
Posted Offer	Average System
Posted Offer	QoS Based System
Reverse Auction	Beta System
Reverse Auction	Blurred System
Reverse Auction	Average System
Reverse Auction	QoS Based System

Tabla 6.1: Configuración de escenarios

Para cada combinación establecida se definió el siguiente proceso:

I. Se debe definir el escenario para la combinación a simular. Para este paso se estableció a manera de prueba que los proveedores serían 4 proveedores y que el número de servicios que se alojarían entre los cuatro sería 10.

II. Se realizaron varias simulaciones. Cada simulación constituye lo que se denominó **ronda**, para la siguiente ronda se actualiza el estado de los proveedores. Por ende, la reputación y la lista de servicios alojados de cada proveedor se verán afectadas en cada simulación dependiendo si el servicio ha sido alojado de manera exitosa o si hubo un fallo en el proceso de alojarlo. Un total de 10 rondas fueron ejecutadas.

Hemos denominado **experimento** al acto de llevar a cabo el proceso descrito anteriormente, para esta etapa de prueba se ejecutaron 10 de estos experimentos.

La notación usada para un experimento fue la siguiente:

- exp_n: representa un experimento y n el número de experimento.
- $round_n$: representa una ronda y n el número de ronda en ejecución.
- n: es un valor entero que va del 1 al 10.

Fueron consideradas para los proveedores las siguientes características:

- $InitialReputation$: valor de reputación inicial.
- $MinimumPrice$: precio mínimo aceptado.
- $MaximumPrice$: precio máximo aceptado.
- $MipsperCore$: instrucciones por núcleo ejecutadas.
- $ProbabilitytoSucceed$: probabilidad de ejecutar el servicio con éxito.
- $Performance$: valor de $performance$.
- $Availiability$: valor de $availiability$.
- $Reliability$: valor de $reliability$.

La figura 6.2 muestra la configuración de cada uno de los proveedores realizada durante las pruebas.

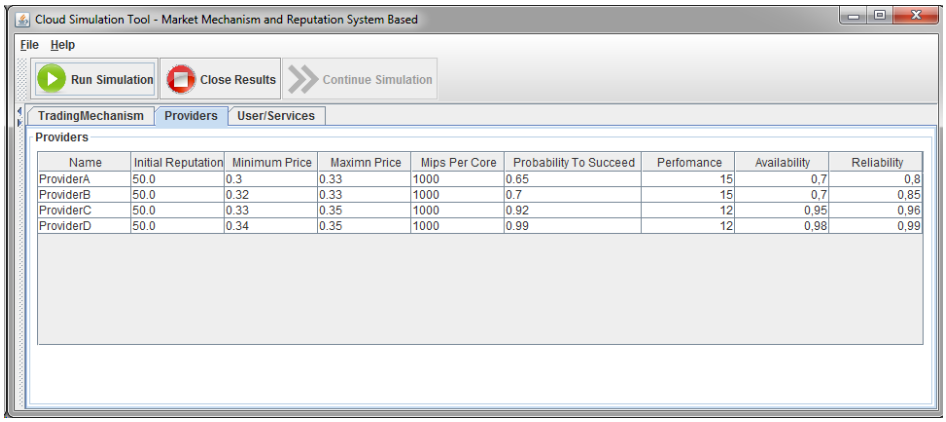


Figura 6.2: Configuración de proveedores para la simulación

Adicionalmente, los servicios establecidos para este proyecto fueron los siguientes (ver figura 6.3). Cada uno de ellos tiene la siguientes propiedades:

- Id: identificador del usuario al que pertenece el servicio.
- $Numberof Requeriments$: número de requerimientos.
- $ScheduledTime$: tiempo de solicitud.
- $MipsperJob$: número de MIPS solicitados.
- $MipsperVM$: número de MIPS por máquina virtual.
- $Min.Reputation$: reputación mínima que debe tener el proveedor.

- $PreferredCost$: costo que el usuario está dispuesto a pagar.
- $MaximumCost$: costo máximo que el usuario está dispuesto a pagar.
- $CostWeight$: valor del peso que se asigna al precio.
- $Availiabilityvalue$: valor de $availiability$.
- $Availiabilityweight$: peso del parámetro $availiability$.
- $Reliabilityvalue$: valor de $reliability$.
- $Reliabilityweight$: peso del parámetro $reliability$.

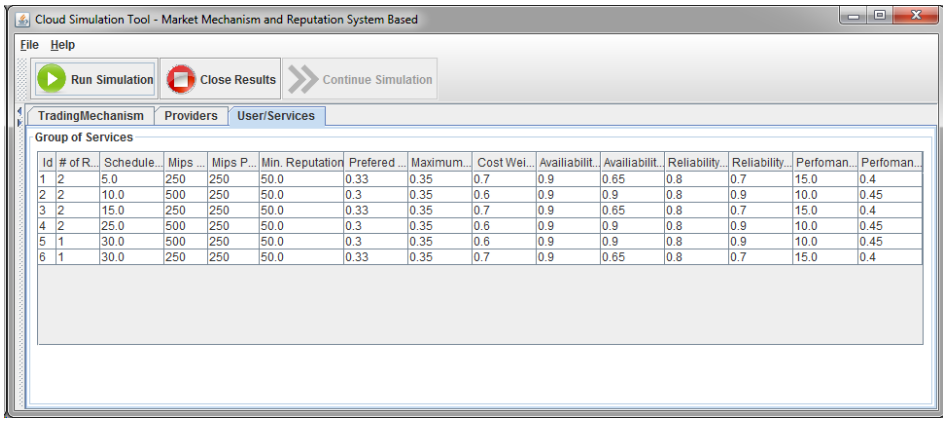


Figura 6.3: Configuración de servicios para la simulación

Con la finalidad de procesar los resultados obtenidos durante el proceso, se utilizó **R** para calcular el número de violaciones ocurridas durante cada una de las rondas en cada uno de los diez experimentos. Posteriormente se obtuvo el promedio de violaciones ocurridas durante dichas rondas. Este proceso se lo hizo con todas las

rondas ejecutadas en todos los experimentos realizados.

Para una mayor descripción de los servicios y los proveedores utilizados en la sección de **anexos** (ver anexos B) se muestran los datos utilizados. Todo el proceso de experimentación fue dividido en dos escenarios, ésta división se hizo considerando el tipo de reputación que estaba siendo utilizada durante la ejecución, en la siguiente sección detallaremos cada uno de ellos.

6.2.1 Escenario 1: Metric Based Reputation

El escenario 1 está compuesto por las simulaciones realizadas con una reputación calculada basada en las métricas usadas por los sistemas de reputación escogidos, que en este caso fueron: Beta System, Blurred System y Average System utilizando los mecanismos de mercado Posted Offer y Reverse Auction.

6.2.1.1 Posted Offer

En esta sección se ilustran los resultados obtenidos con respecto al número de violaciones por rondas ocurridas durante los diez experimentos realizados utilizando el mecanismo Posted Offer. El análisis de dichos resultados se hará en la sección posterior a ésta.

- Beta System

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	2	4	4	3	3	2	4	3	1	3	29
2	Round 2	0	0	1	1	0	2	0	0	5	1	10
3	Round 3	2	2	0	1	1	2	0	0	0	1	9
4	Round 4	1	0	1	0	1	0	0	0	0	0	3
5	Round 5	1	0	1	0	0	3	0	0	2	1	8
6	Round 6	0	0	0	1	0	0	0	2	1	0	4
7	Round 7	1	0	3	1	1	1	0	0	1	0	8
8	Round 8	0	0	1	0	1	1	0	0	1	0	4
9	Round 9	1	0	0	3	2	0	0	0	0	3	9
10	Round 10	0	0	0	1	1	4	0	0	0	0	6

Figura 6.4: Número de violaciones para Posted Offer con Beta System

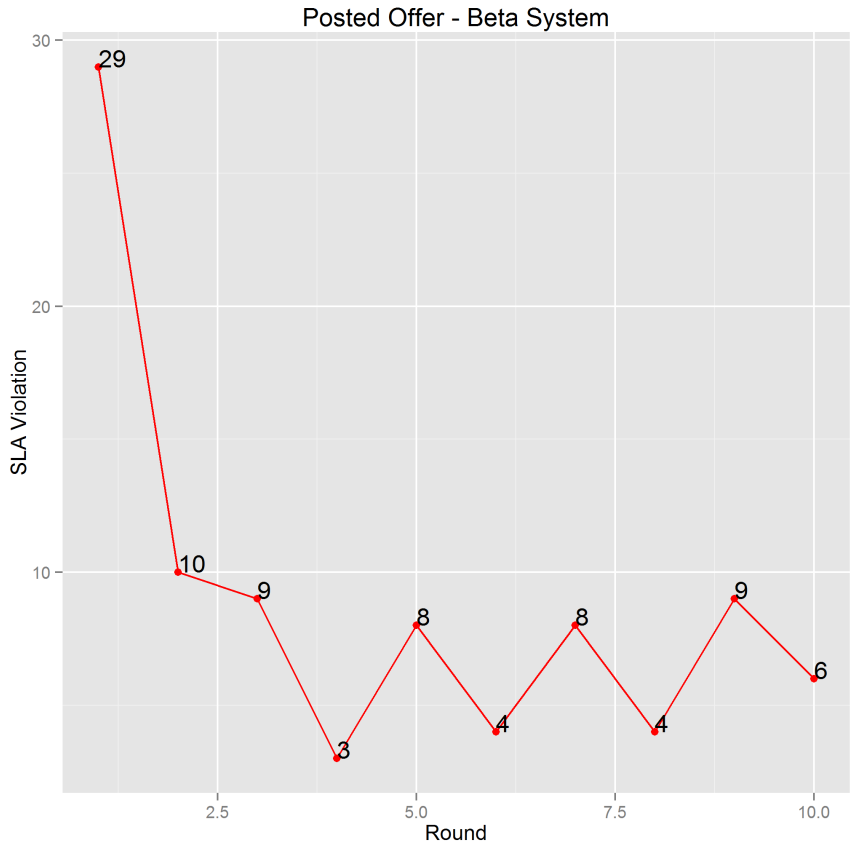


Figura 6.5: Curva de violaciones para Posted Offer con Beta System

- Blurred System

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	1	1	1	2	4	3	3	3	4	1	23
2	Round 2	3	2	1	4	0	1	1	0	0	3	15
3	Round 3	3	2	2	1	0	0	2	1	1	2	14
4	Round 4	0	3	4	2	0	0	1	0	2	0	12
5	Round 5	2	2	2	3	0	0	1	0	0	0	10
6	Round 6	0	0	1	0	0	0	2	0	0	0	3
7	Round 7	0	1	1	0	0	0	0	0	1	0	3
8	Round 8	0	0	0	0	0	0	1	0	0	0	1
9	Round 9	0	0	0	2	0	0	0	0	0	0	2
10	Round 10	0	1	0	0	0	1	1	4	0	0	7

Figura 6.6: Número de violaciones para Posted Offer con Blurred System

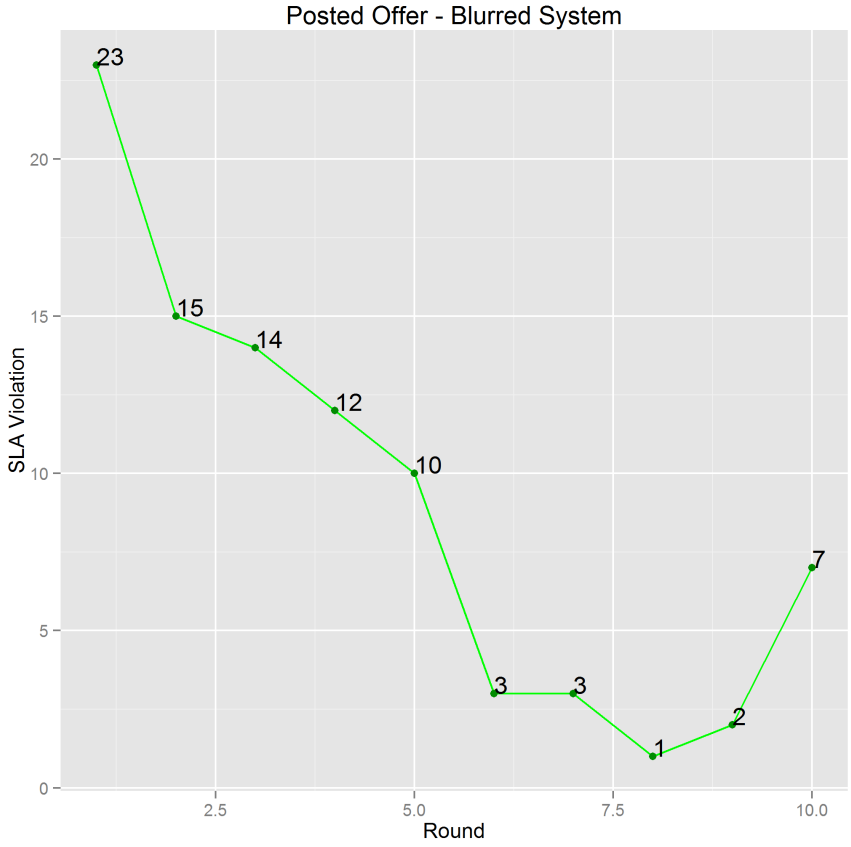


Figura 6.7: Curva de violaciones para Posted Offer con Blurred System

- Average System

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	3	4	2	3	4	3	4	3	3	4	33
2	Round 2	0	0	0	0	1	1	0	1	1	0	4
3	Round 3	0	0	0	0	0	0	0	1	0	0	1
4	Round 4	1	0	0	0	0	2	0	0	0	0	3
5	Round 5	0	2	1	2	0	1	0	2	3	0	11
6	Round 6	2	0	0	1	1	0	0	1	2	2	9
7	Round 7	0	0	0	0	2	1	0	1	0	0	4
8	Round 8	1	1	0	1	1	2	0	2	2	0	10
9	Round 9	1	0	0	2	2	1	0	0	0	0	6
10	Round 10	1	0	1	2	0	0	0	1	2	1	8

Figura 6.8: Número de violaciones para Posted Offer con Average System

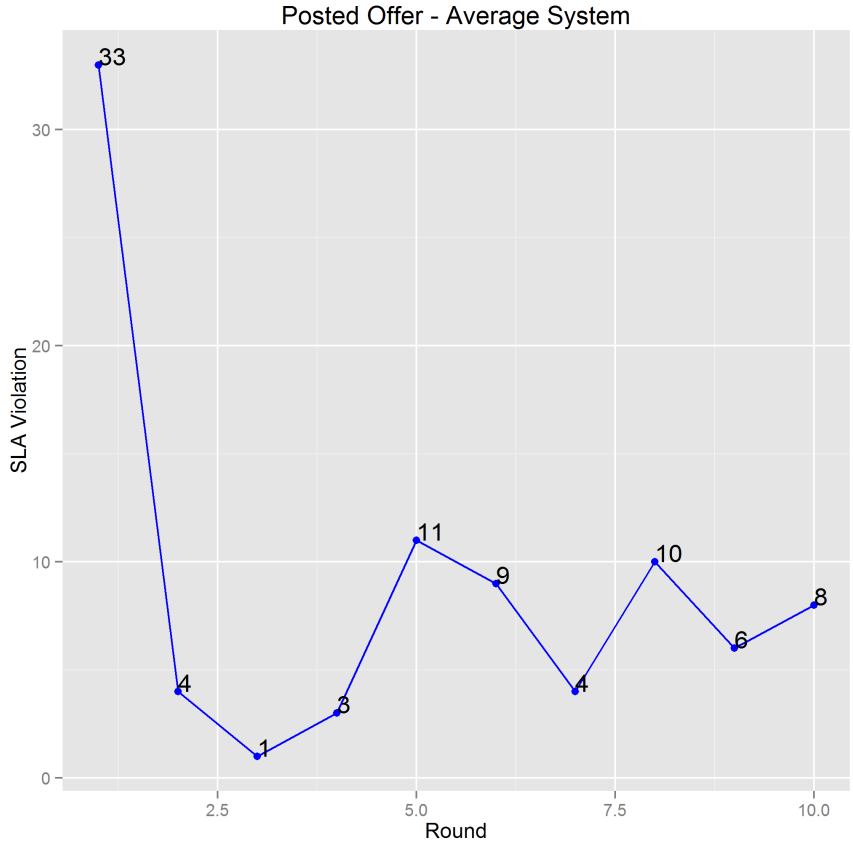


Figura 6.9: Curva de violaciones para Posted Offer con Average System

6.2.1.2 Reverse Auction

A continuación se muestran las violaciones por ronda ocurridas durante los diez experimentos realizados usando el mecanismo de mercado Reverse Auction.

- Beta System

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	4	4	4	2	2	3	4	2	0	2	27
2	Round 2	0	0	0	0	1	1	0	2	4	0	8
3	Round 3	0	0	1	0	1	1	4	1	0	0	8
4	Round 4	0	0	0	2	0	1	0	2	2	0	7
5	Round 5	0	0	0	1	1	1	0	0	2	0	5
6	Round 6	0	0	1	1	0	0	0	0	0	1	3
7	Round 7	0	0	0	3	2	0	0	0	0	1	6
8	Round 8	0	0	1	0	0	1	0	2	1	0	5
9	Round 9	0	0	0	1	0	1	0	0	0	0	2
10	Round 10	0	0	0	0	0	0	0	0	1	0	1

Figura 6.10: Número de violaciones para Reverse Auction con Beta System

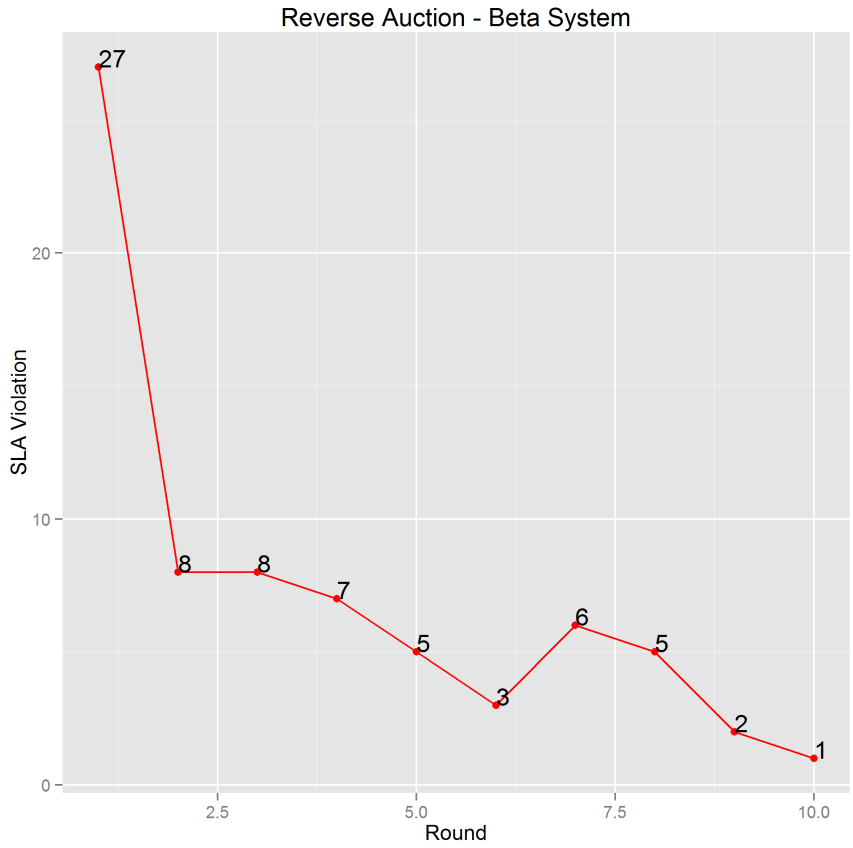


Figura 6.11: Curva de violaciones para Reverse Auction con Beta System

- Blurred System

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	3	6	3	3	2	2	3	3	3	1	29
2	Round 2	0	0	1	0	2	1	0	2	0	2	8
3	Round 3	2	0	1	0	2	0	1	1	2	2	11
4	Round 4	0	0	0	0	1	1	0	0	2	5	9
5	Round 5	0	0	0	0	1	1	2	0	0	1	5
6	Round 6	0	0	0	0	1	1	0	0	2	2	6
7	Round 7	0	0	2	0	3	1	0	0	1	0	7
8	Round 8	0	0	1	0	1	0	1	0	0	0	3
9	Round 9	0	0	2	0	0	1	3	0	1	1	8
10	Round 10	0	0	2	0	3	1	0	0	0	0	6

Figura 6.12: Número de violaciones para Reverse Auction con Blurred System

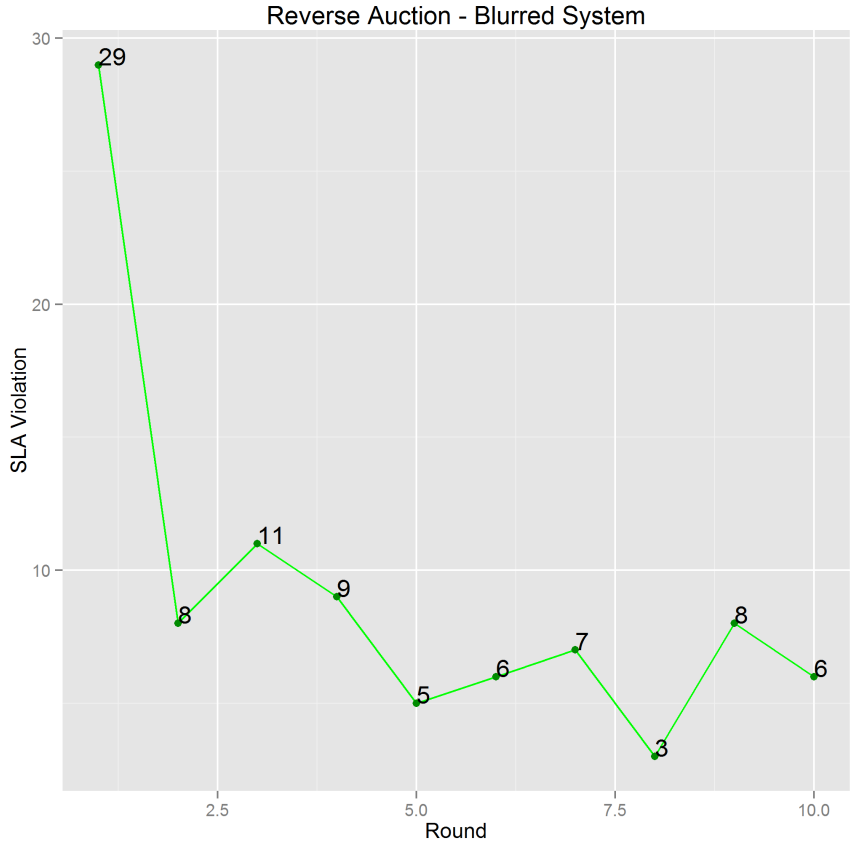


Figura 6.13: Curva de violaciones para Reverse Auction con Blurred System

- Average System

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	3	3	3	4	4	6	4	4	4	3	38
2	Round 2	0	0	0	0	2	1	0	1	0	2	6
3	Round 3	1	2	0	0	0	1	0	0	0	1	5
4	Round 4	1	0	2	0	0	1	1	0	0	0	5
5	Round 5	1	0	2	0	1	2	1	0	0	1	8
6	Round 6	0	0	1	0	0	1	0	0	0	2	4
7	Round 7	1	1	0	0	0	3	0	2	0	0	7
8	Round 8	0	0	2	2	0	0	1	1	0	0	6
9	Round 9	0	0	1	0	0	1	1	0	0	1	4
10	Round 10	2	1	0	0	0	1	0	1	0	1	6

Figura 6.14: Número de violaciones para Reverse Auction con Average System

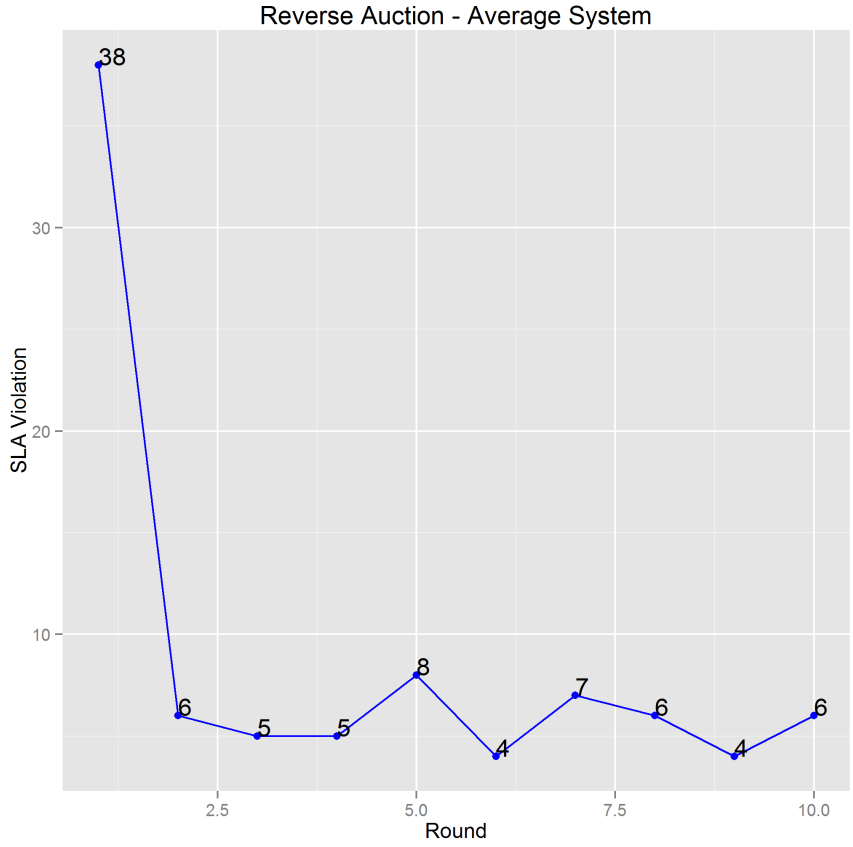


Figura 6.15: Curva de violaciones para Reverse Auction con Average System

6.2.2 Escenario 2: QoS Based Reputation

En este escenario se contempla la simulación con el sistema de reputación QoS Based Reputation utilizando los mecanismos de mercado Posted Offer y Reverse Auction.

6.2.2.1 Posted Offer

La figura 6.16 muestra las violaciones por ronda ocurridas durante los diez experimentos realizados con el mecanismo Posted Offer. En la figura 6.17 se ilustra una curva de las violaciones ocurridas durante la simulación usando dicho mecanismo.

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	3	3	4	3	4	2	2	2	3	3	29
2	Round 2	0	0	0	0	0	2	0	1	1	0	4
3	Round 3	0	1	1	0	0	0	1	1	0	0	4
4	Round 4	1	0	0	0	0	0	3	2	1	0	7
5	Round 5	1	1	0	0	0	0	1	2	0	0	5
6	Round 6	2	1	0	1	1	0	0	0	3	0	8
7	Round 7	1	0	0	2	0	0	2	0	2	0	7
8	Round 8	2	0	1	0	0	0	0	3	1	0	7
9	Round 9	1	1	2	1	2	0	1	1	1	0	10
10	Round 10	0	0	1	0	0	0	1	2	0	0	4

Figura 6.16: Número de violaciones para Posted Offer con QoS Based Reputation

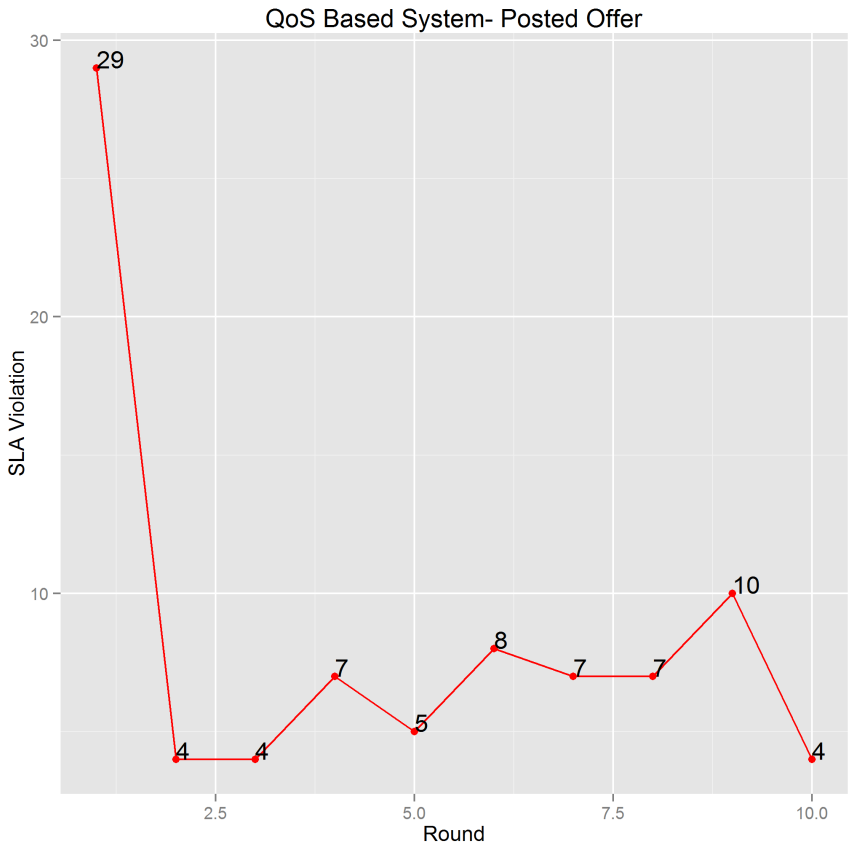


Figura 6.17: Curva de violaciones para Posted Offer con QoS Based Reputation

6.2.2.2 Reverse Auction

La figura 6.18 muestra las violaciones por ronda ocurridas durante los diez experimentos realizados con el mecanismo Reverse Auction. Adicionalmente, en la figura 6.19 se expone una curva de las violaciones ocurridas usando este mecanismo.

	row.names	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Total
1	Round 1	3	2	2	4	2	3	3	2	2	2	25
2	Round 2	1	2	3	0	0	0	0	0	0	3	9
3	Round 3	2	0	1	3	0	0	3	0	1	0	10
4	Round 4	0	0	0	1	2	0	1	1	0	2	7
5	Round 5	1	1	0	1	0	0	1	0	2	0	6
6	Round 6	1	2	0	0	0	0	1	0	0	0	4
7	Round 7	0	2	0	2	1	0	2	0	1	0	8
8	Round 8	0	4	0	1	1	0	1	1	1	0	9
9	Round 9	2	0	0	0	1	0	2	1	1	0	7
10	Round 10	0	0	0	0	2	0	0	1	0	0	3

Figura 6.18: Número de violaciones para Reverse Auction con QoS Based Reputation

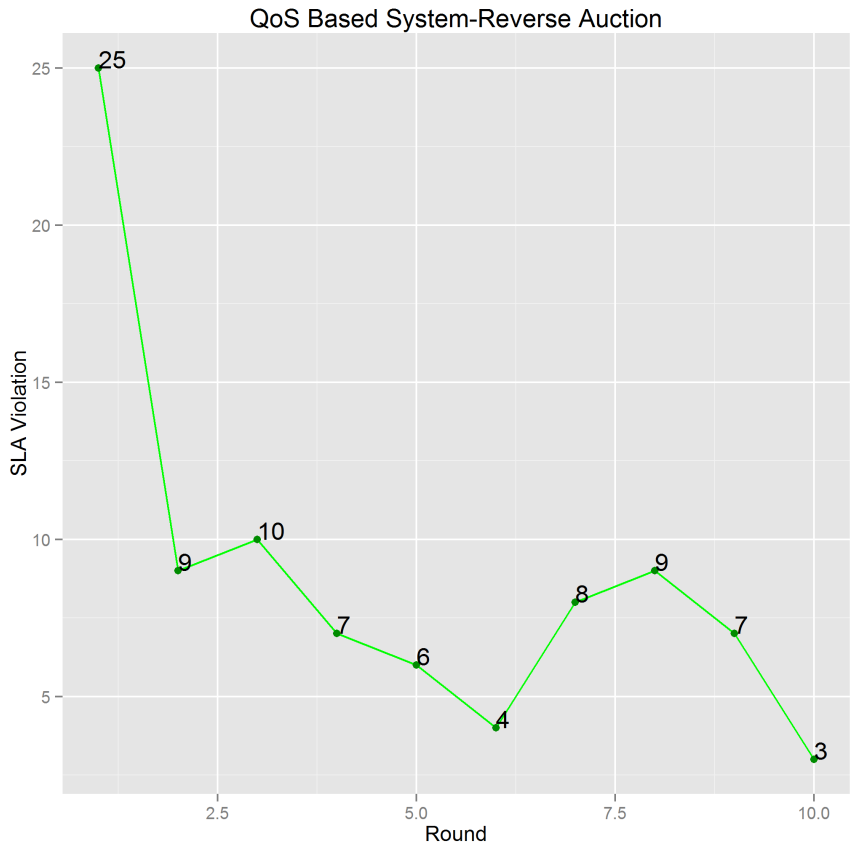


Figura 6.19: Curva de violaciones para Reverse Auction con QoS Based Reputation

6.3 Análisis y evaluación de los resultados

En la sección previa mostramos los resultados obtenidos durante la experimentación. Ahora, analizaremos los eventos ocurridos con respecto a las violaciones en los SLAS durante dicho proceso.

6.3.1 Metric Based Reputation

En las figuras 6.5, 6.7 y 6.9 podemos observar que el número de violaciones tiene una tendencia a disminuir conforme aumenta el número de rondas. En el caso de Posted Offer usando Beta System se tiene que en la $ronda_1$ hay un total de 29 violaciones mientras que en la $ronda_{10}$ hay 6, con lo cual podemos comprobar que hay una disminución considerable en el número de violaciones ocurridas. En cuanto a Blurred System usando el mismo mecanismo de mercado ocurre casi lo mismo: en la primera $ronda_1$ hay un total de 23 violaciones mientras que en la $ronda_{10}$ hay 7. Mientras que con Average System en $ronda_1$ hay un total de 33 violaciones y en la $ronda_{10}$ hay un total de 8.

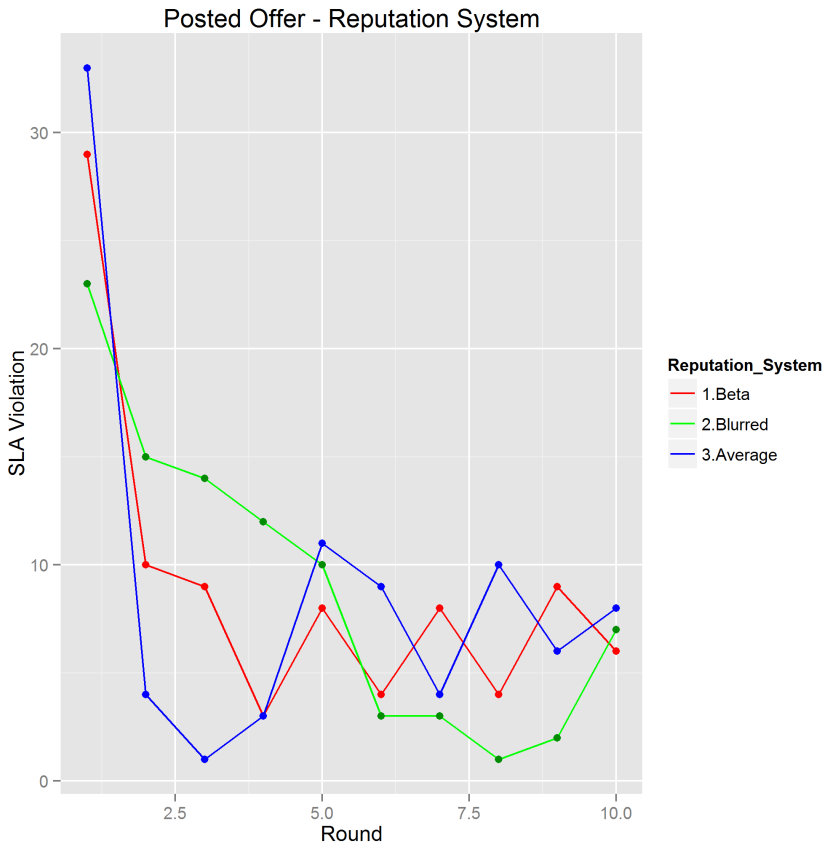


Figura 6.20: Violaciones para Posted Offer

Además, en la figura anterior (ver figura 6.20), se muestran las tres curvas de violaciones usando los diferentes sistemas de reputación con el mecanismo de mercado Posted Offer, donde se puede observar los picos en los valores antes mencionados. Se puede notar que, usando Average System en la $ronda_1$ se obtiene el mayor número de violaciones y que con Blurred System en la misma ronda se obtiene el menor número de violaciones de los tres sistemas de reputación usados. Por otra parte, en la $ronda_{10}$ haciendo uso de Beta System se obtiene la mayor disminución en el número de violaciones y con Average System la menor disminución.

De la misma manera, cuando el mecanismo de mercado utilizado fue Reverse Auction los resultados obtenidos fueron de acuerdo a las figuras 6.11, 6.13 y 6.15. En las cuales podemos considerar una disminución de las violaciones en los SLAs, obteniendo con Beta System en la $ronda_1$ un total de 27 violaciones y en la $ronda_{10}$ un total de 1 violación. En tanto que para Blurred System para la $ronda_1$ existieron un total de 29 violaciones y para la $ronda_{10}$ ocurrieron 6 violaciones en total. Finalmente, para Average System en la $ronda_1$ ocurrieron un total de 38 violaciones y en la $ronda_{10}$ existieron apenas en total 6. Las curvas de la figura 6.21 muestran los resultados usando Reverse Auction con las tres métricas establecidas. Sin duda alguna, dicha figura refleja una disminución considerable en el número de violaciones ocurridas, demostrando que durante la $ronda_1$ con Average System se obtiene el mayor número de violaciones mientras que con Beta System el menor. Del mismo modo, en la $ronda_{10}$ el menor número de violaciones ocurre usando Beta System y el mayor número usando las otras dos métricas.

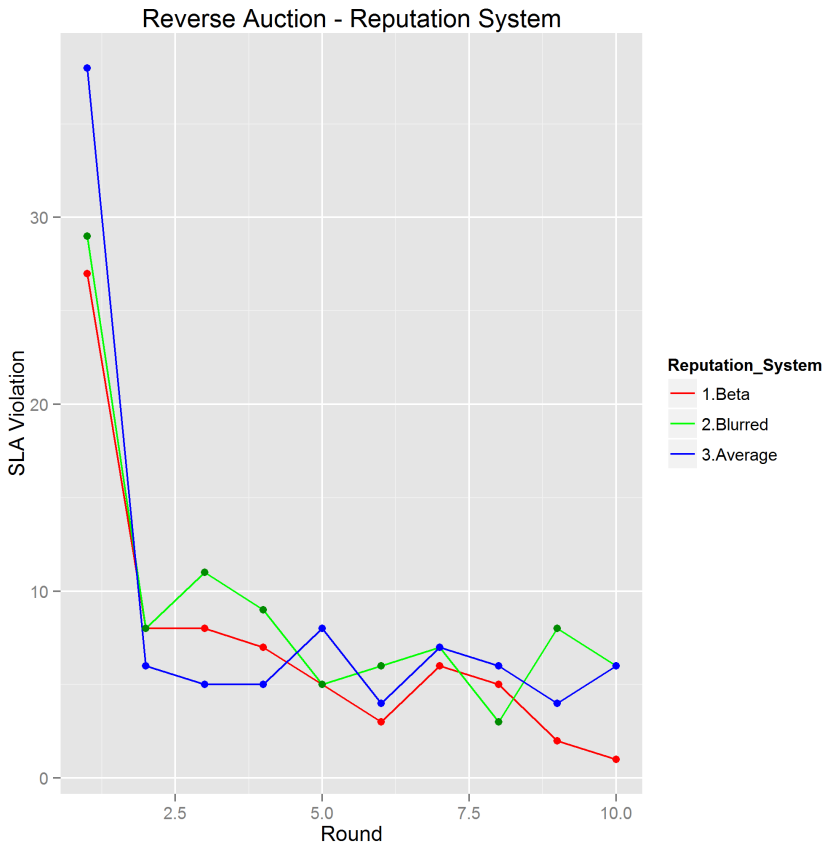


Figura 6.21: Violaciones para Reverse Auction

6.3.2 QoS Based Reputation

Durante la etapa de pruebas haciendo uso del presente sistema de reputación de acuerdo con las figuras 6.16 y 6.18 se observó que el mayor número de violaciones en la $ronda_1$ ocurre utilizando Posted Offer; en total existieron 29 violaciones en este escenario. Mientras que en la $ronda_{10}$ el menor número de violaciones que en este caso fue 3 ocurrió usando Reverse Auction. En la curva 6.22 se ilustra lo que se mencionó en el párrafo anterior, demostrando una disminución considerable

en el número de violaciones.

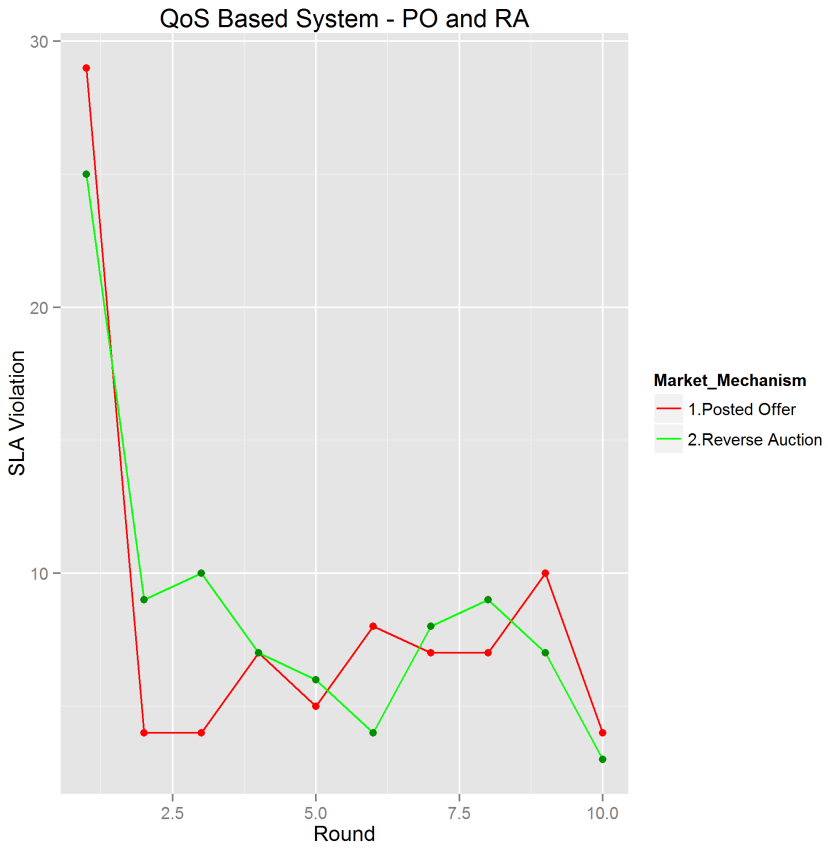


Figura 6.22: Violaciones para Posted Offer y Reverse Auction

En esta sección se puede observar que en las curvas inicialmente se tiene un gran número de violaciones, luego de ejecutar la primera asignación de servicios y haber modificado la reputación de los proveedores, el número de violaciones disminuye. En las siguientes simulaciones el número de violaciones aumentan y disminuyen, pero a nivel general podemos decir que tienden a disminuir.

En las curvas que corresponden al mecanismo de mercado Reverse Auction para los sistemas de reputación: Beta System, Blurred System, Average System y QoS

Based Reputation observamos que la pendiente de las curvas entre ronda y ronda en su mayoría es negativa, con lo cual la disminución de violaciones es más notoria. Cabe recalcar que, en este mecanismo el proceso de selección se realiza con varias ofertas, lo cual hace que el nivel de competencia entre proveedores ocurrido en este mecanismo sea mayor al nivel que existe en Posted Offer.

CAPÍTULO 7

7. Conclusiones

Durante el desarrollo del presente proyecto hemos buscado experimentar con un nuevo modelo para disminuir el número de violaciones a los SLAs que combine los conceptos provenientes de los sistemas basados en el mercado y los sistemas de reputación con la finalidad de mejorar la asignación de recursos en la Nube. Para alcanzar este objetivo hemos implementado el modelo y creado un ambiente controlado a través de una herramienta de simulación de uso abierto(CloudSim). El análisis de los resultados obtenidos a partir de las simulaciones realizadas en esta investigación demuestran que las violaciones a los SLAs disminuyen considerablemente de acuerdo al número de rondas ejecutadas independientemente del sistema de reputación y del mecanismo de mercado empleados para la simulación.

Adicionalmente, hemos podido contrastar dos mecanismos comerciales opuestos uno con el poder de mercado centrado en el vendedor como Posted Offer y otro en el comprador como Reverse Auction, en ambos casos hemos podido contraponer el uso de métricas de reputación con dos enfoques distintos: uno basado en el historial de transacciones del proveedor y el otro basado en los QoS acordados en el SLA. Por otra parte, el crecimiento de los entornos Nube ha incrementado el uso de las aplicaciones de dicho entorno, cabe recalcar que hemos podido comprobar que la reputación de un proveedor es un factor importante a considerar para escogerlo como candidato para usar su servicio, dicho valor permite segmentar o diferenciar a los buenos proveedores de los malos. Sin embargo, el número de violaciones a los SLAs se ha reducido considerablemente independientemente de lo antes mencionado.

A pesar de que nuestro estudio considera mecanismos de mercado con el poder de mercado centrado en el comprador y en el vendedor, una investigación futura podría implementar otro mecanismo de mercado como doble subasta, que trata de equilibrar el poder.

Otro de los aspectos que podrían enriquecer el diseño es la inclusión de nuevos criterios que involucren tiempos, por ejemplo: fecha límite para la finalización del trabajo. En cuanto a la fórmula para establecer la relación entre los distintos parámetros de calidad, ésta ha sido una primera propuesta utilizada para fines de

experimentación. Podría mejorarse teniendo en cuenta nuevos factores o utilizando diferentes fórmulas en función de otros parámetros claves.

Desde el punto de vista de la experimentación, se puede continuar trabajando para ampliar los conceptos de los agentes usuario y proveedor, para que la complejidad de los escenarios aumente, logrando así obtener situaciones más similares a la realidad del mercado, en el cual cuando un proveedor no tiene la demanda que espera o su desempeño no es totalmente eficiente realiza mejoras en los servicios a fin de captar nuevos usuarios, haciendo que los proveedores mejoren sus servicios dinámicamente. Por otro lado, con respecto al origen de los datos para la experimentación a futuro se recomienda el uso de datos estadiśticos de entornos reales de este tipo. Además, se podrían implementar mecanismos de comisiones para el agente broker, para posteriormente hacer estudios estadísticamente exhaustivos que determinen bajo qué mecanismo de mercado y sistema de reputación considerando nuevos parámetros y políticas ocurren mejores asignaciones de recursos. Por último, la herramienta podría extenderse a ser utilizado para la composición de servicios para Nubes Federadas.

APÉNDICE A

PLANTILLA DEL SLA

Como se indicó en la sección 3.1.1.1 (página 56) en este apéndice se encuentra el ejemplo de la plantilla del SLA utilizado en las simulaciones, ésta es un SLA entre proveedores de IT[1] y usuarios de la empresa cliente. El propósito de este acuerdo de nivel de servicioes es identificar los servicios básicos, y cualquier servicio opcional acordado, que será asumido por el proveedor de IT. Además del nombre del departamento encargado del sistema o de la aplicación. Este SLA abarca el período de fecha inicial a fecha final, durante el cual será revisado.

Service Level Agreement					
		Between IT department and user clients	**Date:MM/YY** _____		
		Contacts **IT Dept:**_____ **ABC user:**_____	**Effective Dates:** **From MM/YY:** _____ **to MM/YY:**_____		
		Approvals **IT Dept:**_____ **ABC user:**_____	**Effective Dates:** **From MM/YY** _____ **to MM/YY**_____		
Services	**QoS parameter**		**Goal**	**Actual**	**Difference**
	Availiability	8 am-5pm Mon-Sun	98 %	100 %	2 %
	Reliability	% de respuestas por intervalo de tiempo	90 %	95 %	5 %
	Performance	relación entre el monto de recursos y el tiempo de ejecución de los servicios.			
	Cost	valor que el usuario está dispuesto a pagar.			
		Firmas			
		IT DEPT.:_____	**ABC USER.:**_____		

Tabla A.1: Plantilla de SLA

[1] Information Technology

Criterios del SLA

- Valores de penalizacón para cada parámetro QoS:

  a. $penalty\ availability : 0; 20$

  b. $penalty\ reliability : 0; 20$

  c. $penalty\ performance : 0; 05$

- Los parámetros: $availability\ y\ reliability$ son criterios de tipo **HARD**.

- El parámetro $performance$ es un criterio de tipo **SOFT**.

- El parámetro $cost$ no tiene penalización, es un valor constante acordado previamente con el usuario y varía de acuerdo al servicio.

APÉNDICE B

DATOS DE PROVEEDORES Y SERVICIOS UTILIZADOS PARA LA SIMULACIÓN

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario>
<providers>
    <provider>
        <name>ProviderA</name>
        <initialReputation>50.0</initialReputation>
        <preferredPrice>0.3</preferredPrice>
        <maximunPrice>0.33</maximunPrice>
        <mipsPerCore>1000</mipsPerCore>
        <probabilityToSucceed>0.65</probabilityToSucceed>
        <perfomance>15.0</perfomance>
        <availability>0.7</availability>
        <reliability>0.8</reliability>
    </provider>
    <provider>
        <name>ProviderB</name>
        <initialReputation>50.0</initialReputation>
        <preferredPrice>0.32</preferredPrice>
        <maximunPrice>0.33</maximunPrice>
        <mipsPerCore>1000</mipsPerCore>
        <probabilityToSucceed>0.7</probabilityToSucceed>
        <perfomance>15.0</perfomance>
        <availability>0.7</availability>
        <reliability>0.85</reliability>
    </provider>
    <provider>
        <name>ProviderC</name>
        <initialReputation>50.0</initialReputation>
        <preferredPrice>0.33</preferredPrice>
        <maximunPrice>0.35</maximunPrice>
```

```
31            <mipsPerCore>1000</mipsPerCore>
32            <probabilityToSucceed>0.92</probabilityToSucceed>
33            <perfomance>12.0</perfomance>
34            <availability>0.95</availability>
35            <reliability>0.96</reliability>
36        </provider>
37        <provider>
38            <name>ProviderD</name>
39            <initialReputation>50.0</initialReputation>
40            <preferredPrice>0.34</preferredPrice>
41            <maximunPrice>0.35</maximunPrice>
42            <mipsPerCore>1000</mipsPerCore>
43            <probabilityToSucceed>0.99</probabilityToSucceed>
44            <perfomance>12.0</perfomance>
45            <availability>0.98</availability>
46            <reliability>0.99</reliability>
47        </provider>
48    </providers>
49    <trading />
50    <users>
51        <user>
52            <id>1</id>
53            <nRequirements>2</nRequirements>
54            <scheduledTime>5.0</scheduledTime>
55            <mipsPerCloudlet>250</mipsPerCloudlet>
56            <mipsPerVm>250</mipsPerVm>
57            <minimumReputation>50.0</minimumReputation>
58            <minProfit>0.33</minProfit>
59            <maximumProfit>0.35</maximumProfit>
60            <costWeight>0.7</costWeight>
61            <availability>0.9</availability>
62            <availabilityWeight>0.65</availabilityWeight>
63            <reliability>0.8</reliability>
64            <reliabilityWeight>0.7</reliabilityWeight>
65            <perfomance>15.0</perfomance>
66            <perfomanceWeight>0.4</perfomanceWeight>
67        </user>
68        <user>
69            <id>2</id>
70            <nRequirements>2</nRequirements>
```

```
71          <scheduledTime>10.0</scheduledTime>
72          <mipsPerCloudlet>500</mipsPerCloudlet>
73          <mipsPerVm>250</mipsPerVm>
74          <minimumReputation>50.0</minimumReputation>
75          <minProfit>0.3</minProfit>
76          <maximumProfit>0.35</maximumProfit>
77          <costWeight>0.6</costWeight>
78          <availability>0.9</availability>
79          <availabilityWeight>0.9</availabilityWeight>
80          <reliability>0.8</reliability>
81          <reliabilityWeight>0.9</reliabilityWeight>
82          <perfomance>10.0</perfomance>
83          <perfomanceWeight>0.45</perfomanceWeight>
84      </user>
85      <user>
86          <id>3</id>
87          <nRequirements>2</nRequirements>
88          <scheduledTime>15.0</scheduledTime>
89          <mipsPerCloudlet>250</mipsPerCloudlet>
90          <mipsPerVm>250</mipsPerVm>
91          <minimumReputation>50.0</minimumReputation>
92          <minProfit>0.33</minProfit>
93          <maximumProfit>0.35</maximumProfit>
94          <costWeight>0.7</costWeight>
95          <availability>0.9</availability>
96          <availabilityWeight>0.65</availabilityWeight>
97          <reliability>0.8</reliability>
98          <reliabilityWeight>0.7</reliabilityWeight>
99          <perfomance>15.0</perfomance>
100         <perfomanceWeight>0.4</perfomanceWeight>
101     </user>
102     <user>
103         <id>4</id>
104         <nRequirements>2</nRequirements>
105         <scheduledTime>25.0</scheduledTime>
106         <mipsPerCloudlet>500</mipsPerCloudlet>
107         <mipsPerVm>250</mipsPerVm>
108         <minimumReputation>50.0</minimumReputation>
109         <minProfit>0.3</minProfit>
110         <maximumProfit>0.35</maximumProfit>
```

```
111             <costWeight>0.6</costWeight>
112             <availability>0.9</availability>
113             <availabilityWeight>0.9</availabilityWeight>
114             <reliability>0.8</reliability>
115             <reliabilityWeight>0.9</reliabilityWeight>
116             <perfomance>10.0</perfomance>
117             <perfomanceWeight>0.45</perfomanceWeight>
118         </user>
119         <user>
120             <id>5</id>
121             <nRequirements>1</nRequirements>
122             <scheduledTime>30.0</scheduledTime>
123             <mipsPerCloudlet>500</mipsPerCloudlet>
124             <mipsPerVm>250</mipsPerVm>
125             <minimumReputation>50.0</minimumReputation>
126             <minProfit>0.3</minProfit>
127             <maximumProfit>0.35</maximumProfit>
128             <costWeight>0.6</costWeight>
129             <availability>0.9</availability>
130             <availabilityWeight>0.9</availabilityWeight>
131             <reliability>0.8</reliability>
132             <reliabilityWeight>0.9</reliabilityWeight>
133             <perfomance>10.0</perfomance>
134             <perfomanceWeight>0.45</perfomanceWeight>
135         </user>
136         <user>
137             <id>6</id>
138             <nRequirements>1</nRequirements>
139             <scheduledTime>30.0</scheduledTime>
140             <mipsPerCloudlet>250</mipsPerCloudlet>
141             <mipsPerVm>250</mipsPerVm>
142             <minimumReputation>50.0</minimumReputation>
143             <minProfit>0.33</minProfit>
144             <maximumProfit>0.35</maximumProfit>
145             <costWeight>0.7</costWeight>
146             <availability>0.9</availability>
147             <availabilityWeight>0.65</availabilityWeight>
148             <reliability>0.8</reliability>
149             <reliabilityWeight>0.7</reliabilityWeight>
150             <perfomance>15.0</perfomance>
```

```
151             <perfomanceWeight>0.4</perfomanceWeight>
152         </user>
153 </users>
154 </scenario>
```

APÉNDICE C

CÓDIGO R PARA GENERAR TABLAS DE RESULTADOS

Metric Based Reputation - Posted Offer

```
1  po<-function(x){
2      print("Processing Data: Provider Based Reputation")
3      print("Posted Offer")
4      require("XML")
5      require("plyr")
6      xmlfile = xmlParse(x)
7      jobs.data =ldply(xmlToList(x), data.frame)
8      #View(jobs.data)
9      jobs.data.temp <- jobs.data[c("nExp","nRq","reputationType","alghoritmType",
           "auctionType","status")]
10     #View(jobs.data.temp)
11     asdfg <- seq(0,1,by=0.1)
12
13     #Beta
14     jobs.data.0.0.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
            jobs.data.temp$alghoritmType==0 & jobs.data.temp$auctionType ==0, )
15     #View(jobs.data.0.0.0)
16     jobs.data.0.0.0.total <- ddply(jobs.data.0.0.0,.(nExp,nRq),summarise,failed
           = sum(status==0),total = length(status), percent = (failed*100) /total)
17     #View(jobs.data.0.0.0.total)
18     jobs.data.0.0.0.mean <- ddply(jobs.data.0.0.0.total,.(nRq),summarise,failed
           = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
19    #View(jobs.data.0.0.0.mean)
20     Beta = matrix(jobs.data.0.0.0.total$failed,ncol=jobs.data.0.0.0.mean$n,nrow=
           length(jobs.data.0.0.0.mean$nRq))
21     dimnames(Beta) <- list(rownames(Beta, do.NULL = FALSE, prefix = "Round "),
           colnames(Beta, do.NULL = FALSE, prefix = "Exp. "))
22    BetaTotal = matrix(jobs.data.0.0.0.mean$failed,ncol=1)
23     dimnames(BetaTotal) = list(c(1:length(jobs.data.0.0.0.mean$nRq)), c("Total")
           ) # column names
```

```
24  Beta <- cbind(Beta,BetaTotal)
25  View(Beta)
26
27  #Blurred
28  jobs.data.0.1.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==1 & jobs.data.temp$auctionType ==0, )
29  #View(jobs.data.0.1.0)
30  jobs.data.0.1.0.total <- ddply(jobs.data.0.1.0,.(nExp,nRq),summarise,failed
        = sum(status==0),total = length(status), percent = (failed*100) /total)
31  #View(jobs.data.0.1.0.total)
32  jobs.data.0.1.0.mean <- ddply(jobs.data.0.1.0.total,.(nRq),summarise,failed
        = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
33  #View(jobs.data.0.1.0.mean)
34  Blurred = matrix(jobs.data.0.1.0.total$failed,ncol=jobs.data.0.1.0.mean$n,
        nrow=length(jobs.data.0.1.0.mean$nRq))
35  dimnames(Blurred) <- list(rownames(Blurred, do.NULL = FALSE, prefix = "Round
        "),colnames(Blurred, do.NULL = FALSE, prefix = "Exp. "))
36  BlurredTotal = matrix(jobs.data.0.1.0.mean$failed,ncol=1)
37  dimnames(BlurredTotal) = list(c(1:length(jobs.data.0.1.0.mean$nRq)), c("
        Total")) # column names
38  Blurred <- cbind(Blurred,BlurredTotal)
39  View(Blurred)
40
41  #Average
42  jobs.data.0.2.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==2 & jobs.data.temp$auctionType ==0, )
43  #View(jobs.data.0.2.0)
44  jobs.data.0.2.0.total <- ddply(jobs.data.0.2.0,.(nExp,nRq),summarise,failed
        = sum(status==0),total = length(status), percent = (failed*100) /total)
45  #View(jobs.data.0.2.0.total)
46  jobs.data.0.2.0.mean <- ddply(jobs.data.0.2.0.total,.(nRq),summarise,failed
        = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
47  #View(jobs.data.0.2.0.mean)
48  Average = matrix(jobs.data.0.2.0.total$failed,ncol=jobs.data.0.2.0.mean$n,
        nrow=length(jobs.data.0.2.0.mean$nRq))
49  dimnames(Average) <- list(rownames(Average, do.NULL = FALSE, prefix = "Round
        "),colnames(Average, do.NULL = FALSE, prefix = "Exp. "))
50  AverageTotal = matrix(jobs.data.0.2.0.mean$failed,ncol=1)
51  dimnames(AverageTotal) = list(c(1:length(jobs.data.0.2.0.mean$nRq)), c("
        Total")) # column names
```

```
52     Average <- cbind(Average,AverageTotal)
53     View(Average)
54 }
```

Metric Based Reputation - Reverse Auction

```
1  ra<-function(x){
2     print("Processing Data: Provider Based Reputation")
3     print("Reverse Auction")
4     require("XML")
5     require("plyr")
6     xmlfile = xmlParse(x)
7     jobs.data =ldply(xmlToList(x), data.frame)
8     #View(jobs.data)
9     jobs.data.temp <- jobs.data[c("nExp","nRq","reputationType","alghoritmType",
          "auctionType","status")]
10    #View(jobs.data.temp)
11    asdfg <- seq(0,1,by=0.1)
12
13    #Beta
14    jobs.data.0.0.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
           jobs.data.temp$alghoritmType==0 & jobs.data.temp$auctionType ==1, )
15    #View(jobs.data.0.0.0)
16    jobs.data.0.0.0.total <- ddply(jobs.data.0.0.0,.(nExp,nRq),summarise,failed
          = sum(status==0),total = length(status), percent = (failed*100) /total)
17    #View(jobs.data.0.0.0.total)
18    jobs.data.0.0.0.mean <- ddply(jobs.data.0.0.0.total,.(nRq),summarise,failed
          = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
19   #View(jobs.data.0.0.0.mean)
20    Beta = matrix(jobs.data.0.0.0.total$failed,ncol=jobs.data.0.0.0.mean$n,nrow=
          length(jobs.data.0.0.0.mean$nRq))
21    dimnames(Beta) <- list(rownames(Beta, do.NULL = FALSE, prefix = "Round "),
          colnames(Beta, do.NULL = FALSE, prefix = "Exp. "))
22   BetaTotal = matrix(jobs.data.0.0.0.mean$failed,ncol=1)
23    dimnames(BetaTotal) = list(c(1:length(jobs.data.0.0.0.mean$nRq)), c("Total")
          ) # column names
24   Beta <- cbind(Beta,BetaTotal)
25    View(Beta)
26
27    #Blurred
```

```
28    jobs.data.0.1.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
          jobs.data.temp$alghoritmType==1 & jobs.data.temp$auctionType ==1, )
29    #View(jobs.data.0.1.0)
30    jobs.data.0.1.0.total <- ddply(jobs.data.0.1.0,.(nExp,nRq),summarise,failed
          = sum(status==0),total = length(status), percent = (failed*100) /total)
31    #View(jobs.data.0.1.0.total)
32    jobs.data.0.1.0.mean <- ddply(jobs.data.0.1.0.total,.(nRq),summarise,failed
          = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
33    #View(jobs.data.0.1.0.mean)
34    Blurred = matrix(jobs.data.0.1.0.total$failed,ncol=jobs.data.0.1.0.mean$n,
          nrow=length(jobs.data.0.1.0.mean$nRq))
35    dimnames(Blurred) <- list(rownames(Blurred, do.NULL = FALSE, prefix = "Round
          "),colnames(Blurred, do.NULL = FALSE, prefix = "Exp. "))
36    BlurredTotal = matrix(jobs.data.0.1.0.mean$failed,ncol=1)
37    dimnames(BlurredTotal) = list(c(1:length(jobs.data.0.1.0.mean$nRq)), c("
          Total")) # column names
38    Blurred <- cbind(Blurred,BlurredTotal)
39    View(Blurred)
40
41   #Average
42    jobs.data.0.2.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
          jobs.data.temp$alghoritmType==2 & jobs.data.temp$auctionType ==1, )
43    #View(jobs.data.0.2.0)
44    jobs.data.0.2.0.total <- ddply(jobs.data.0.2.0,.(nExp,nRq),summarise,failed
          = sum(status==0),total = length(status), percent = (failed*100) /total)
45    #View(jobs.data.0.2.0.total)
46    jobs.data.0.2.0.mean <- ddply(jobs.data.0.2.0.total,.(nRq),summarise,failed
          = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
47    #View(jobs.data.0.2.0.mean)
48    Average = matrix(jobs.data.0.2.0.total$failed,ncol=jobs.data.0.2.0.mean$n,
          nrow=length(jobs.data.0.2.0.mean$nRq))
49    dimnames(Average) <- list(rownames(Average, do.NULL = FALSE, prefix = "Round
          "),colnames(Average, do.NULL = FALSE, prefix = "Exp. "))
50    AverageTotal = matrix(jobs.data.0.2.0.mean$failed,ncol=1)
51    dimnames(AverageTotal) = list(c(1:length(jobs.data.0.2.0.mean$nRq)), c("
          Total")) # column names
52    Average <- cbind(Average,AverageTotal)
53    View(Average)
54 }
```

Qos Based Reputation

```
qo<-function(x){
    print("Processing Data: Qos Based Reputation")
    require("XML")
    require("plyr")
    xmlfile = xmlParse(x)
    jobs.data =ldply(xmlToList(x), data.frame)
    #View(jobs.data)
    jobs.data.temp <- jobs.data[c("nExp","nRq","reputationType","alghoritmType",
        "auctionType","status")]
    #View(jobs.data.temp)
    asdfg <- seq(0,1,by=0.1)

    #Beta
    jobs.data.1.3.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==1 &
         jobs.data.temp$alghoritmType==3 & jobs.data.temp$auctionType ==0, )
   #View(jobs.data.1.3.0)
    jobs.data.1.3.0.total <- ddply(jobs.data.1.3.0,.(nExp,nRq),summarise,failed
        = sum(status==0),total = length(status), percent = (failed*100) /total)
    View(jobs.data.1.3.0.total)
    jobs.data.1.3.0.mean <- ddply(jobs.data.1.3.0.total,.(nRq),summarise,failed
        = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
    View(jobs.data.1.3.0.mean)
    Beta = matrix(jobs.data.1.3.0.total$failed,ncol=jobs.data.1.3.0.mean$n,nrow=
        length(jobs.data.1.3.0.mean$nRq))
    dimnames(Beta) <- list(rownames(Beta, do.NULL = FALSE, prefix = "Round "),
        colnames(Beta, do.NULL = FALSE, prefix = "Exp. "))
   BetaTotal = matrix(jobs.data.1.3.0.mean$failed,ncol=1)
    dimnames(BetaTotal) = list(c(1:length(jobs.data.1.3.0.mean$nRq)), c("Total")
        ) # column names
   Beta <- cbind(Beta,BetaTotal)
    View(Beta)

    #Blurred
    jobs.data.1.3.1 <-subset( jobs.data.temp, jobs.data.temp$reputationType==1 &
         jobs.data.temp$alghoritmType==3 & jobs.data.temp$auctionType ==1, )
    #View(jobs.data.1.3.1)
    jobs.data.1.3.1.total <- ddply(jobs.data.1.3.1,.(nExp,nRq),summarise,failed
        = sum(status==0),total = length(status), percent = (failed*100) /total)
    View(jobs.data.1.3.1.total)
```

```
31    jobs.data.1.3.1.mean <- ddply(jobs.data.1.3.1.total,.(nRq),summarise,failed
          = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
32    View(jobs.data.1.3.1.mean)
33    Blurred = matrix(jobs.data.1.3.1.total$failed,ncol=jobs.data.1.3.1.mean$n,
          nrow=length(jobs.data.1.3.1.mean$nRq))
34    dimnames(Blurred) <- list(rownames(Blurred, do.NULL = FALSE, prefix = "Round
          "),colnames(Blurred, do.NULL = FALSE, prefix = "Exp. "))
35    BlurredTotal = matrix(jobs.data.1.3.1.mean$failed,ncol=1)
36    dimnames(BlurredTotal) = list(c(1:length(jobs.data.1.3.1.mean$nRq)), c("
          Total")) # column names
37    Blurred <- cbind(Blurred,BlurredTotal)
38    View(Blurred)
39 }
```

APÉNDICE D

CÓDIGO R PARA GENERAR GRÁFICAS DE RESULTADOS

Metric Based Reputation - Posted Offer

```
1  po<-function(x){
2      print("Processing Data: Provider Based Reputation")
3      print("Posted Offer")
4      require("XML")
5      require("plyr")
6      require("ggplot2")
7      require("reshape2")
8      xmlfile = xmlParse(x)
9      jobs.data =ldply(xmlToList(x), data.frame)
10     #View(jobs.data)
11     jobs.data.temp <- jobs.data[c("nExp","nRq","reputationType","alghoritmType",
           "auctionType","status")]
12     #View(jobs.data.temp)
13     asdfg <- seq(0,1,by=0.1)
14
15     #Beta
16     jobs.data.0.0.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
           jobs.data.temp$alghoritmType==0 & jobs.data.temp$auctionType ==0, )
17     #View(jobs.data.0.0.0)
18
19     jobs.data.0.0.0.total <- ddply(jobs.data.0.0.0,.(nRq,nExp),summarise,failed
           = sum(status==0),total = length(status), percent = (failed*100) /total)
20     #View(jobs.data.0.0.0.total)
21
22     jobs.data.0.0.0.mean <- ddply(jobs.data.0.0.0.total,.(nRq),summarise,failed
           = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
23     #View(jobs.data.0.0.0.mean)
24
```

```
25  jobs.data.0.0.0.average<-jobs.data.0.0.0.mean$failed
26  #View(jobs.data.0.0.0.average)
27
28  provider.based.vector.metrics.beta <- jobs.data.0.0.0.mean$nRq
29  #View(provider.based.vector.metrics.beta)
30
31
32  #Blurred
33  jobs.data.0.1.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==1 & jobs.data.temp$auctionType ==0, )
34  #View(jobs.data.0.1.0)
35
36  jobs.data.0.1.0.total <- ddply(jobs.data.0.1.0,.(nRq,nExp),summarise,failed
      = sum(status==0),total = length(status), percent = (failed*100) /total)
37  #View(jobs.data.0.1.0.total)
38
39  jobs.data.0.1.0.mean <- ddply(jobs.data.0.1.0.total,.(nRq),summarise,failed
      = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
40  #View(jobs.data.0.1.0.mean)
41
42  jobs.data.0.1.0.average<-jobs.data.0.1.0.mean$failed
43  #View(jobs.data.0.1.0.average)
44
45  provider.based.vector.metrics.blurred <- jobs.data.0.1.0.mean$nRq
46  #View(provider.based.vector.metrics.blurred)
47
48
49
50  #Average
51  jobs.data.0.2.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==2 & jobs.data.temp$auctionType ==0, )
52  #View(jobs.data.0.2.0)
53
54  jobs.data.0.2.0.total <- ddply(jobs.data.0.2.0,.(nRq,nExp),summarise,failed
      = sum(status==0),total = length(status), percent = (failed*100) /total)
55  #View(jobs.data.0.2.0.total)
56
57  jobs.data.0.2.0.mean <- ddply(jobs.data.0.2.0.total,.(nRq),summarise,failed
      = sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
58  #View(jobs.data.0.2.0.mean)
```

```
59
60  jobs.data.0.2.0.average<-jobs.data.0.2.0.mean$failed
61  #View(jobs.data.0.2.0.average)
62
63  provider.based.vector.metrics.average <- jobs.data.0.2.0.mean$nRq
64  #View(provider.based.vector.metrics.average)
65
66  test_data <- data.frame(
67      beta = as.numeric(jobs.data.0.0.0.average),
68      blurred = as.numeric(jobs.data.0.1.0.average),
69      average = as.numeric(jobs.data.0.2.0.average),
70      rounds = as.numeric(provider.based.vector.metrics.beta)
71  )
72  print(test_data)
73  p1<-ggplot(test_data,aes(x=rounds,colour=Reputation_System))+
74  geom_line(aes(y = beta, colour = "1.Beta"))+
75  geom_line(aes(y = blurred, colour = "2.Blurred"))+
76  geom_line(aes(y = average, colour = "3.Average"))+
77  xlab("Round") +
78  ylab("SLA Violation") +
79  ggtitle("Posted Offer - Reputation System")+
80  scale_color_manual(values=c("#FF0000","#00FF00","#0000FF","#000000"))+
81  geom_point(data=test_data, mapping=aes(x=rounds, y=beta),color = "red")+
82  geom_point(data=test_data, mapping=aes(x=rounds, y=blurred),color = "green4"
        )+
83  geom_point(data=test_data, mapping=aes(x=rounds, y=average),color = "blue")
84
85  #X11()
86  p2<-ggplot(test_data, aes(x=rounds))+
87  geom_line(aes(y = beta),color="red")+
88  xlab("Round") +
89  ylab("SLA Violation") +
90  ggtitle("Posted Offer - Beta System")+
91  scale_color_manual(values=c("#FF0000","#FF0000","#FF0000"))+
92  geom_point(data=test_data, mapping=aes(x=rounds, y=beta),color = "red")
93
94
95  #X11()
96  p3<-ggplot(test_data, aes(x=rounds))+
97  geom_line(aes(y = blurred),color="green")+
```

```
98      xlab("Round") +
99      ylab("SLA Violation") +
100     ggtitle("Posted Offer - Blurred System")+
101     scale_color_manual(values=c("#00FF00","#00FF00"))+
102     geom_point(data=test_data, mapping=aes(x=rounds, y=blurred),color = "green4"
            )
103
104     #X11()
105     p4<-ggplot(test_data, aes(x=rounds))+
106     geom_line(aes(y = average),color="blue")+
107     xlab("Round") +
108     ylab("SLA Violation") +
109     ggtitle("Posted Offer - Average System")+
110     geom_point(data=test_data, mapping=aes(x=rounds, y=average),color = "blue")+
111     scale_color_manual(values=c("#0000FF","#0000FF"))
112
113     ggsave(filename="betaPO.png", plot=p2)
114     ggsave(filename="blurredPO.png", plot=p3)
115     ggsave(filename="averagePO.png", plot=p4)
116     ggsave(filename="allPO.png", plot=p1)
117     multiplot(p2, p3, p4, p1, cols=2)
118 }
```

Metric Based Reputation - Reverse Auction

```
1  ra<-function(x){
2    print("Processing Data: Provider Based Reputation")
3    print("Reverse Auction")
4    require("XML")
5    require("plyr")
6    require("ggplot2")
7    require("reshape2")
8    xmlfile = xmlParse(x)
9    jobs.data =ldply(xmlToList(x), data.frame)
10   #View(jobs.data)
11   jobs.data.temp <- jobs.data[c("nExp","nRq","reputationType","alghoritmType","
         auctionType","status")]
12   #View(jobs.data.temp)
13   asdfg <- seq(0,1,by=0.1)
14
```

```
15  #Beta
16  jobs.data.0.0.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==0 & jobs.data.temp$auctionType ==1, )
17  #View(jobs.data.0.0.0)
18
19  jobs.data.0.0.0.total <- ddply(jobs.data.0.0.0,.(nRq,nExp),summarise,failed =
        sum(status==0),total = length(status), percent = (failed*100) /total)
20  #View(jobs.data.0.0.0.total)
21
22  jobs.data.0.0.0.mean <- ddply(jobs.data.0.0.0.total,.(nRq),summarise,failed =
        sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
23  #View(jobs.data.0.0.0.mean)
24
25  jobs.data.0.0.0.average<-jobs.data.0.0.0.mean$failed
26  #View(jobs.data.0.0.0.average)
27
28  provider.based.vector.metrics.beta <- jobs.data.0.0.0.mean$nRq
29  #View(provider.based.vector.metrics.beta)
30
31
32  #Blurred
33  jobs.data.0.1.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==1 & jobs.data.temp$auctionType ==1, )
34  #View(jobs.data.0.1.0)
35
36  jobs.data.0.1.0.total <- ddply(jobs.data.0.1.0,.(nRq,nExp),summarise,failed =
        sum(status==0),total = length(status), percent = (failed*100) /total)
37  #View(jobs.data.0.1.0.total)
38
39  jobs.data.0.1.0.mean <- ddply(jobs.data.0.1.0.total,.(nRq),summarise,failed =
        sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
40  #View(jobs.data.0.1.0.mean)
41
42  jobs.data.0.1.0.average<-jobs.data.0.1.0.mean$failed
43  #View(jobs.data.0.1.0.average)
44
45  provider.based.vector.metrics.blurred <- jobs.data.0.1.0.mean$nRq
46  #View(provider.based.vector.metrics.blurred)
47
48
```

```
49
50  #Average
51  jobs.data.0.2.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==0 &
        jobs.data.temp$alghoritmType==2 & jobs.data.temp$auctionType ==1, )
52  #View(jobs.data.0.2.0)
53
54  jobs.data.0.2.0.total <- ddply(jobs.data.0.2.0,.(nRq,nExp),summarise,failed =
        sum(status==0),total = length(status), percent = (failed*100) /total)
55  #View(jobs.data.0.2.0.total)
56
57  jobs.data.0.2.0.mean <- ddply(jobs.data.0.2.0.total,.(nRq),summarise,failed =
        sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
58  #View(jobs.data.0.2.0.mean)
59
60  jobs.data.0.2.0.average<-jobs.data.0.2.0.mean$failed
61  #View(jobs.data.0.2.0.average)
62
63  provider.based.vector.metrics.average <- jobs.data.0.2.0.mean$nRq
64  #View(provider.based.vector.metrics.average)
65
66  test_data <- data.frame(
67    beta = as.numeric(jobs.data.0.0.0.average),
68    blurred = as.numeric(jobs.data.0.1.0.average),
69    average = as.numeric(jobs.data.0.2.0.average),
70    rounds = as.numeric(provider.based.vector.metrics.beta)
71  )
72  print(test_data)
73  p1<-ggplot(test_data,aes(x=rounds,colour=Reputation_System))+
74    geom_line(aes(y = beta, colour = "1.Beta"))+
75    geom_line(aes(y = blurred, colour = "2.Blurred"))+
76    geom_line(aes(y = average, colour = "3.Average"))+
77    xlab("Round") +
78    ylab("SLA Violation") +
79    ggtitle("Reverse Auction - Reputation System")+
80    scale_color_manual(values=c("#FF0000","#00FF00","#0000FF","#000000"))+
81    geom_point(data=test_data, mapping=aes(x=rounds, y=beta),color = "red")+
82    geom_point(data=test_data, mapping=aes(x=rounds, y=blurred),color = "green4
          ")+
83    geom_point(data=test_data, mapping=aes(x=rounds, y=average),color = "blue")
84
```

```
85    #X11()
86    p2<-ggplot(test_data, aes(x=rounds))+
87      geom_line(aes(y = beta),color="red")+
88      xlab("Round") +
89      ylab("SLA Violation") +
90      ggtitle("Reverse Auction - Beta System")+
91      scale_color_manual(values=c("#FF0000","#FF0000","#FF0000"))+
92      geom_point(data=test_data, mapping=aes(x=rounds, y=beta),color = "red")
93
94
95    #X11()
96    p3<-ggplot(test_data, aes(x=rounds))+
97      geom_line(aes(y = blurred),color="green")+
98      xlab("Round") +
99      ylab("SLA Violation") +
100     ggtitle("Reverse Auction - Blurred System")+
101     scale_color_manual(values=c("#00FF00","#00FF00"))+
102     geom_point(data=test_data, mapping=aes(x=rounds, y=blurred),color = "green4
            ")
103
104   #X11()
105   p4<-ggplot(test_data, aes(x=rounds))+
106     geom_line(aes(y = average),color="blue")+
107     xlab("Round") +
108     ylab("SLA Violation") +
109     ggtitle("Reverse Auction - Average System")+
110     geom_point(data=test_data, mapping=aes(x=rounds, y=average),color = "blue")
            +
111     scale_color_manual(values=c("#0000FF","#0000FF"))
112
113   ggsave(filename="betaRA.png", plot=p2)
114   ggsave(filename="blurredRA.png", plot=p3)
115   ggsave(filename="averageRA.png", plot=p4)
116   ggsave(filename="allRA.png", plot=p1)
117   multiplot(p2, p3, p4, p1, cols=2)
118 }
```

Qos Based Reputation

```
1 qos<-function(x){
```

```
2  print("Processing Data: Provider Based Reputation")
3  print("QoS Based System")
4  require("XML")
5  require("plyr")
6  require("ggplot2")
7  require("reshape2")
8  xmlfile = xmlParse(x)
9  jobs.data =ldply(xmlToList(x), data.frame)
10 #View(jobs.data)
11 jobs.data.temp <- jobs.data[c("nExp","nRq","reputationType","alghoritmType","
       auctionType","status")]
12 #View(jobs.data.temp)
13 asdfg <- seq(0,1,by=0.1)
14
15 #PO
16 jobs.data.1.3.0 <-subset( jobs.data.temp, jobs.data.temp$reputationType==1 &
       jobs.data.temp$alghoritmType==3 & jobs.data.temp$auctionType ==0, )
17 #View(jobs.data.1.3.0)
18 jobs.data.1.3.0.total <- ddply(jobs.data.1.3.0,.(nRq,nExp),summarise,failed =
       sum(status==0),total = length(status), percent = (failed*100) /total)
19 #View(jobs.data.1.3.0.total)
20 jobs.data.1.3.0.mean <- ddply(jobs.data.1.3.0.total,.(nRq),summarise,failed =
       sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
21 #View(jobs.data.1.3.0.mean)
22 jobs.data.1.3.0.average<-jobs.data.1.3.0.mean$mean
23 #View(jobs.data.1.3.0.average)
24 provider.based.vector.metrics.beta <- jobs.data.1.3.0.mean$nRq
25 #View(provider.based.vector.metrics.beta)
26
27
28 #RA
29 jobs.data.1.3.1 <-subset( jobs.data.temp, jobs.data.temp$reputationType==1 &
       jobs.data.temp$alghoritmType==3 & jobs.data.temp$auctionType ==1, )
30 #View(jobs.data.1.3.1)
31 jobs.data.1.3.1.total <- ddply(jobs.data.1.3.1,.(nRq,nExp),summarise,failed =
       sum(status==0),total = length(status), percent = (failed*100) /total)
32 #View(jobs.data.1.3.1.total)
33 jobs.data.1.3.1.mean <- ddply(jobs.data.1.3.1.total,.(nRq),summarise,failed =
       sum(failed),mean = mean(percent),total = sum(total), n = length(nExp))
34 #View(jobs.data.1.3.1.mean)
```

```
jobs.data.1.3.1.average<-jobs.data.1.3.1.mean$mean
#View(jobs.data.1.3.1.average)
provider.based.vector.metrics.beta <- jobs.data.1.3.1.mean$nRq
#View(provider.based.vector.metrics.beta)


test_data <- data.frame(
  po = as.numeric(jobs.data.1.3.0.average),
  ra = as.numeric(jobs.data.1.3.1.average),
  rounds = as.numeric(provider.based.vector.metrics.beta)
)
print(test_data)
p1<-ggplot(test_data,aes(x=rounds,colour=Market_Mechanism))+
  geom_line(aes(y = po, colour = "1.Posted Offer"))+
  geom_line(aes(y = ra, colour = "2.Reverse Auction"))+
  xlab("Round") +
  ylab("SLA Violation") +
  ggtitle("QoS Based System – PO and RA")+
  scale_color_manual(values=c("#FF0000","#00FF00","#0000FF","#000000"))+
  geom_point(data=test_data, mapping=aes(x=rounds, y=po),color = "red")+
  geom_point(data=test_data, mapping=aes(x=rounds, y=ra),color = "green4")

p2<-ggplot(test_data, aes(x=rounds))+
  geom_line(aes(y = po),color="red")+
  xlab("Round")+
  ylab("SLA Violation") +
  ggtitle("QoS Based System– Posted Offer")+
  scale_color_manual(values=c("#FF0000","#FF0000","#FF0000"))+
  geom_point(data=test_data, mapping=aes(x=rounds, y=po),color = "red")


#X11()
p3<-ggplot(test_data, aes(x=rounds))+
  geom_line(aes(y = ra),color="green")+
  xlab("Round") +
  ylab("SLA Violation") +
  ggtitle("QoS Based System–Reverse Auction")+
  scale_color_manual(values=c("#00FF00","#00FF00"))+
  geom_point(data=test_data, mapping=aes(x=rounds, y=ra),color = "green4")

```

```
75    ggsave(filename="poQOS.png", plot=p2)
76    ggsave(filename="raQOS.png", plot=p3)
77    ggsave(filename="allQOS.png", plot=p1)
78    multiplot(p2, p1, p3, cols=2)
79  }
```

BIBLIOGRAFÍA

[1] Amazon. Amazon Web Services. http://aws.amazon.com/es/ec2/, 2014. Consultado el: 2014-02-01.

[2] Ardagna, D., Panicucci, B. y Passacantando, M. A game theoretic formulation of the service provisioning problem in cloud systems. En *Proceedings of the 20th International Conference on World Wide Web*, páginas 177–186. ACM, 2011. ISBN 978-1-4503-0632-4.

[3] Buyya, R. y Murshed, M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. En *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE (CCPE*, páginas 1175–1220. 2002.

[4] Buyya, R., Pandey, S. y Vecchiola, C. Cloudbus toolkit for market-oriented cloud computing. En *Cloud Computing*, páginas 24–44. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10664-4.

[5] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F. y Buyya, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice*

and Experience, páginas 23–50, 2011.

[6] Calheiros, R. N., Ranjan, R., Rose, C. A. F. D. y Buyya, R. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *CoRR*, 2009.

[7] Chappell, D. Introducing the windows azure platform. 2009. Consultado el: 2014-01-01.

[8] Cliff, D. y Bruton, J. Minimal–intelligence agents for bargaining behaviors in market–based environments. Informe técnico, University of Sussex–Brighton, 1997.

[9] Davis, D. D. y Holt, C. A. Experimental economics: Methods, problems, and promise. *Estudios Economicos*, páginas 179–212, 1993.

[10] Emeakaroha, V. C., Calheiros, R. N., Netto, M. A., Brandic, I. y De Rose, C. A. Desvi: An architecture for detecting sla violations in cloud computing infrastructures. En *Proceedings of the 2nd International ICST Conference on Cloud Computing (CloudComp10)*. 2010.

[11] Faniyi, F. y Bahsoon, R. Self-managing sla compliance in cloud architectures: A market-based approach. En *Proceedings of the 3rd International ACM SIGSOFT Symposium on Architecting Critical Systems*, páginas 61–70. ACM,

2012. ISBN 978-1-4503-1347-6.

[12] Faniyi, F. y Bahsoon, R. Self-managing sla compliance in cloud architectures: A market-based approach. En *Proceedings of the 3rd International ACM SIGSOFT Symposium on Architecting Critical Systems*, páginas 61–70. ACM, 2012. ISBN 978-1-4503-1347-6.

[13] Faniyi, F., Bahsoon, R. y Theodoropoulos, G. A dynamic data-driven simulation approach for preventing service level agreement violations in cloud federation. *Procedia Computer Science*, páginas 1167–1176, 2012.

[14] Ganghishetti, P. y Wanka, R. Quality of service design in clouds. 2011. Consultado el: 2013-12-20.

[15] Google. Google App Engine. https://cloud.google.com/appengine, 2014. Consultado el: 2014-02-01.

[16] Jin, L.-j., Machiraju, V. y Sahai, A. Analysis on service level agreement of web services. *HP June*, 2002.

[17] Ketcham, J., Smith, V. L. y Williamns, A. W. A comparison of posted-offer and double-auction pricing institutions. *The Review of Economic Studies*, páginas 595–614, 1984.

[18] Krishna, V. *Auction Theory*. Academic press, 2009.

[19] Legrand, A., Marchal, L. y Supérieuredelyon, E. N. Scheduling distributed applications: The simgrid simulation framework. En *In Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGri'03*, páginas 138–145. 2003.

[20] Leopoldi, R. It services management a description of service level agreements. *RL Consulting*, 2002.

[21] Macías, M. y Guitart, J. Client classification policies for sla negotiation and allocation in shared cloud datacenters. En *Economics of Grids, Clouds, Systems, and Services*, páginas 90–104. Springer, 2012.

[22] Mahoney, G. Trust, distributed systems, and the sybil attack. *PANDA Seminar Talk*, 2002.

[23] Mell, P. y Grance, T. The nist definition of cloud computing. *Technical report, Information Technology Laboratory*, 2009.

[24] Mui, L. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. Tesis Doctoral, Massachusetts Institute of Technology, 2003. Homepage has year as 2003, copyright is 2002, go figure.

[25] ODEH, Y. y ODEH, M. A new classification of non-functional requirements for service-oriented software enineering. 2011.

[26] Rao, S., Rao, N. y Kumari, K. Cloud computing: An overview. *Journal of Theoretical and Applied Information Technology*, páginas 71–76, 2009.

[27] Reiff-Marganiec, S., Yu, H. Q. y Tilly, M. Service selection based on non-functional properties. En *Service-Oriented Computing-ICSOC 2007 Workshops*, páginas 128–138. Springer, 2009.

[28] Resnick, P., Kuwabara, K., Zeckhauser, R. y Friedman, E. Reputation systems. *Commun. ACM*, páginas 45–48, 2000.

[29] Sabater, J. y Sierra, C. Social regret, a reputation model based on social relations. *SIGecom Exch.*, páginas 44–56, 2001.

[30] Schlosser, A., Voss, M. y Brückner, L. Comparing and evaluating metrics for reputation systems by simulation. 2004.

[31] Schlosser, A., Voss, M. y Brückner, L. On the simulation of global reputation systems. *Journal of Artificial Societies and Social Simulation*, página 4, 2005.

[32] Shen, Y. *Research report: Reputation system simulation results* . Proyecto Fin de Carrera, University of Helsinki,Helsinki Institute for Information Technology

(HIIT), Helsinki–Finland, 2009.

[33] Shi, W. y Hong, B. Resource allocation with a budget constraint for computing independent tasks in the cloud. *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, páginas 319–328, 2010.

[34] Song, H. J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K. y Chien, A. The microgrid: a scientific tool for modeling computational grids. En *Supercomputing, ACM/IEEE 2000 Conference*, páginas 53–53. IEEE, 2000.

[35] Suleiman, B., Sakr, S., Jeffery, R. y Liu, A. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *Journal of Internet Services and Applications*, páginas 173–193, 2012.

[36] Sun, D., Chang, G., Wang, C., Xiong, Y. y Wang, X. Efficient nash equilibrium based cloud resource allocation by using a continuous double auction. En *Computer Design and Applications (ICCDA), 2010 International Conference on*, páginas V1–94–V1–99. 2010.

[37] Weiss, A. Computing in the clouds. *Magazine netWorker*, páginas 16–25, 2007.

[38] Wu, L. y Buyya, R. Service level agreement (sla) in utility computing systems. *arXiv preprint arXiv:1010.2881*, 2010.

[39] Wustenhoff, E. y BluePrints, S. Service level agreement in the data center. 2002. Consultado el: 2013-12-20.

[40] Wyld, D. C. Reverse auctioning: Saving money and increasing transparency. *Using Technology Series*, página 7, 2011.

[41] Yeo, C. S. y Buyya, R. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software: Practice and Experience*, páginas 1381–1419, 2006.

[42] Yfoulis, C. A. y Gounaris, A. Honoring slas on cloud computing services: a control perspective. 2009.

[43] Zhou, Q. y Martin, P. Minimal intelligence agents for bargaining behaviors in market based environments. *ACSW Frontiers 04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, páginas 175–180, 2004.