Especificación de la Calidad en Software-as-a-Service: Definición de un Esquema de Calidad basado en el Estándar ISO/IEC 25010

María Julia Blas, Silvio Gonnet, Horacio Leone

INGAR, Instituto de Desarrollo y Diseño CONICET-UTN, Santa Fe, Argentina
{mariajuliablas, sgonnet, hleone}@santafe-conicet.gov.ar

Abstract. En la actualidad, no es sencillo encontrar herramientas que den soporte al análisis y evaluación de la calidad de servicios de software. En este trabajo se presenta un esquema que facilita el estudio de las propiedades de calidad asociadas a Software-as-a-Service, el cual integra en un único documento la información relacionada a la definición y entendimiento de las características de calidad (modelo de calidad) y la forma en la cual estas características deben ser relevadas (métricas de software). El esquema de calidad resultante contribuye no sólo al entendimiento de los aspectos de calidad asociados específicamente al ámbito de servicios de software, sino que además posibilita el análisis de la información requerida para la medición de la calidad en un contexto específico.

Keywords: Cloud computing, modelo de calidad, esquema de calidad, métricas de software, evaluación de la calidad.

1 Introducción

El estudio de la calidad de software es tan antiguo como el estudio del software en sí mismo [1]. De acuerdo con la IEEE, la calidad de un software es "el grado en el cual el software posee una combinación de atributos deseados" [2]. Sin embargo, la intrínseca complejidad de estos productos dificulta la determinación de la calidad asociada [3]. Este panorama empeora al intentar aplicar los conceptos y técnicas existentes sobre plataformas más novedosas basadas en modelos de ejecución modernos; como ser, por ejemplo, el esquema de computación en la nube (Cloud Computing, CC).

En las últimas décadas, los modelos de calidad de software (MCS) se han convertido en poderosos mecanismos de soporte para la gestión de la calidad [1]. Estos modelos presentan una taxonomía de atributos de calidad junto con un conjunto de relaciones que los vinculan, los cuales pueden ser utilizados como marco de referencia durante la elaboración de la especificación inicial, la evaluación del diseño y el testeo de un sistema de software [4]. Existen dos tipos de modelos: generales y específicos [5]. Mientras que los primeros son desarrollados con el objetivo de ser utilizados en cualquier tipo de software (es decir, sus atributos de calidad se eligen para ser aplicables sobre cualquier tipo de producto), los modelos específicos son diseñados para ser utilizados sobre una clase de software particular (por lo que sus atributos son elegidos

para cubrir características de calidad propias de la clase elegida). La selección del MCS a ser utilizado sobre un producto de software concreto es un verdadero reto [5]. Para poder realizar una verdadera comparación en un ámbito específico, es importante que el experto a cargo posea la destreza requerida para identificar el modelo apropiado (esto es, el modelo que posea la mayor cantidad de características de calidad pretendidas) [6]. En entornos de CC esta búsqueda se convierte en un verdadero problema ya que no existe un MCS consolidado para el análisis, medición y evaluación de la calidad; lo que lleva a una falta de entendimiento y claridad en el área.

Tomando como punto de partida esta problemática, en este trabajo se presenta un modelo de calidad genérico que puede aplicarse en cualquier tipo de modelo de distribución basado en Software-as-a-Service (SaaS). Este modelo se complementa con un conjunto de métricas que posibilitan el relevamiento de las diferentes propiedades de calidad identificadas. Las relaciones entre ambos elementos sientan las bases requeridas para especificar un esquema de calidad genérico aplicable a la evaluación de la calidad en entornos SaaS. La definición de este tipo de esquemas junto con su utilidad y aplicabilidad han sido detalladas formalmente en un trabajo previo [7].

El resto del trabajo se encuentra estructurado de la siguiente manera. La sección 2 introduce el concepto de CC poniendo énfasis en la necesidad de evaluar la calidad a nivel de capa de aplicación. La sección 3 presenta la especificación del esquema de calidad propuesto, detallando la forma en que se incorporaron las propiedades SaaS al modelo de calidad de producto de software definido en el estándar ISO/IEC 25010 [8] y las métricas requeridas para el relevamiento de cada una de estas propiedades. Algunos posibles usos del esquema resultante se describen en la sección 4. Finalmente la sección 5 sintetiza las conclusiones y trabajos futuros relacionados.

2 Cloud Computing & Software-as-a-Service

El paradigma de CC se ha convertido rápidamente en una de las estrategias de solución tecnológica más populares e influyentes del mundo actual [9]. Esto se debe a que logra un aumento en la capacidad y productividad en tiempo de ejecución, sin requerir inversión en infraestructura, licencias de software, ni capacitación para empleados [10]. Este modelo computacional posibilita el acceso a internet bajo demanda de forma beneficiosa y generalizada, permitiendo compartir un conjunto de recursos de computación configurables. Tales recursos son asignados y adjudicados rápidamente con un bajo esfuerzo de administración o con muy poca interacción con el proveedor del servicio [11].

La mayoría de las arquitecturas cloud propuestas se basan en el modelo de capas [9, 10, 12]. Sin embargo, dada su variabilidad, se optó por diseñar una nueva arquitectura que sintetice las características fundamentales del paradigma. La arquitectura resultante se visualiza en la Figura 1. Como puede observarse, el esquema consta de cinco capas, cada una de las cuales ha sido diseñada con un propósito específico, a saber:

1. Hardware: Capa responsable de asignar los recursos de forma eficiente.
2. Software kernel: Capa responsable de administrar los recursos de hardware subyacentes haciendo uso de software específico.

3. Software infrastructure: Capa responsable de administrar los recursos de red a las capas superiores.
4. Software environment: Capa responsable de proveer una plataforma de desarrollo para aplicaciones web que se ejecutarán sobre los recursos cloud.
5. Application: Capa responsable de facilitar a los usuarios el acceso a las aplicaciones instaladas en el centro de datos de un proveedor cloud. En esta capa residen las aplicaciones web que han sido desarrolladas haciendo uso de la plataforma situada en la capa precedente.

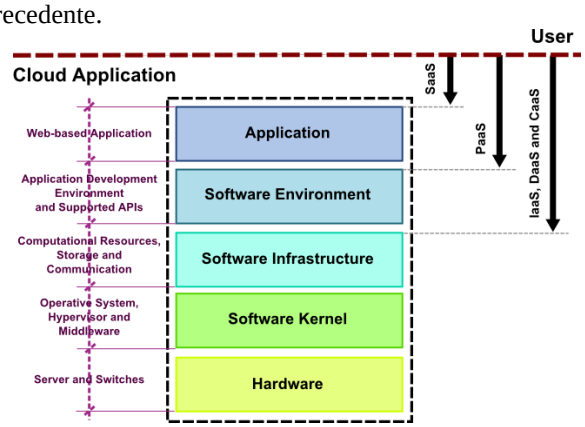

Fig. 1. Arquitectura cloud basada en capas (adaptado de [9],[10]).

Teniendo en cuenta que el concepto detrás de CC es bajar el cómputo a los proveedores de recursos remotos, el proveedor de servicios cloud es el responsable de ejecutar, administrar y actualizar los recursos de hardware de acuerdo a los requerimientos de los usuarios. Este modelo computacional permite que los usuarios utilicen los servicios bajo demanda, según su consumo, a través de Internet. Sin embargo, no ofrece un único servicio. A diferencia de otros esquemas, el paradigma de CC contempla la posibilidad de ofrecer a los usuarios un conjunto de servicios relacionados, dentro de los cuales se destacan: infrastructure as a service (IaaS), data storage as a service (DaaS), communication as a service (CaaS), platform as a service (PaaS) y software as a service (SaaS). Tal como se visualiza en la Figura 1, cada uno de estos servicios se obtiene por medio de la exposición a los usuarios del software y/o hardware incluido dentro de un subconjunto de las capas propuestas como parte de la arquitectura.

2.1 Capa de Aplicación y Software-as-a-Service (SaaS)

La utilización del esquema SaaS como modelo de entrega de servicio se ha incrementado en los últimos años. No sólo se encuentra en ascenso la cantidad de servicios prestados y el número de usuarios, sino que también se evidencia un notorio crecimiento en la cantidad de proveedores disponibles para este tipo de plataformas [13].

Cuando un usuario accede a una aplicación cloud (alojada en la capa superior de la arquitectura propuesta), se encuentra indirectamente accediendo a un servicio de software. Este tipo de servicios posee múltiples beneficios para los usuarios, entre los que se incluyen [14]:

- *Disponibilidad y movilidad*: Los usuarios tienen la posibilidad de acceder a la funcionalidad de software en cualquier momento, desde cualquier ubicación, por medio de una conexión a Internet.
- *Bajos costos*: Los usuarios evitan el costo o inversión inicial de la funcionalidad de software, quedando además exentos de la realización de tareas de mantenimiento y/o actualizaciones del mismo.
- *Reducción de riesgos*: Los usuarios pueden utilizar la nube para probar ideas y conceptos vinculados a funcionalidades de software, de forma tal que les permita analizar el accionar de una solución previo a la realización de una mayor inversión.
- *Pago de acuerdo al uso*: Los usuarios pagan una cuota cuyo valor queda determinado de acuerdo al uso que realizan sobre la funcionalidad de software.

Para poder garantizar los beneficios enunciados con anterioridad, es evidente que un SaaS debe mantener un nivel de calidad relativamente más alto que el de un sistema de software tradicional. Aún más, debido a que las capas inferiores proveen de funcionalidad y recursos a la capa de aplicación, el relevamiento de la calidad de este tipo de servicios no trata únicamente con aspectos de calidad de aplicación. Teniendo en cuenta que el consumidor del servicio (usuario) experimenta de forma directa las características funcionales y no funcionales del mismo y, tomando en consideración el crecimiento de proveedores que se ha dado en los últimos tiempos, es evidente que el análisis y entendimiento de la calidad por parte de los desarrolladores puede llevar al éxito o fracaso de un servicio de software. En consecuencia, el estudio y relevamiento de las características de calidad propias de este tipo de servicios requiere de una rigurosa evaluación. El esquema de calidad desarrollado proporciona una especificación simple y concisa de las relaciones existentes entre los aspectos de calidad SaaS, dando como resultado un poderoso mecanismo de comunicación y análisis que facilita el entendimiento de la información relacionada con la calidad de servicios de software.

3 Esquema de Calidad: SaaS Quality Scheme

Al intentar examinar la calidad haciendo uso de una evaluación de atributos, se presenta un gran número de inconvenientes. Dentro de los principales problemas se incluyen: realización del control de calidad como última etapa del proyecto, entendimiento erróneo de la importancia de los atributos de calidad y utilización de un lenguaje inadecuado para expresar y especificar la calidad. Con el objetivo de superar estas dificultades, en [7] se definió una estrategia de documentación basada en ontologías que ayuda a especificar semánticamente las relaciones existentes entre diferentes conceptos relacionados con la calidad de software. Tal estrategia de documentación se denomina *esquema de calidad* (quality scheme, QS).

Dada la especificación de un producto de software, un QS queda definido como el conjunto de tripletas del tipo *{ atributo de software, métrica de software, subcaracterística de calidad }* que se utiliza para especificar los aspectos de calidad asociados al producto definido, dentro del cual: el *atributo de software* referencia a una parte de una entidad de software que requiere de una propiedad de calidad específica, la *métrica de software* es el mecanismo de medición que debe ser utilizado para relevar la

calidad del *atributo de software* y *la subcaracterística de calidad* es la propiedad de calidad que debe ser evaluada haciendo uso de la *métrica de software* sobre el *atributo de calidad*. La Figura 2 modela estas relaciones definiendo cada elemento de la tripleta como un concepto. Cada concepto pertenece a un modelo semántico, a saber:

- El *atributo de software* (concepto *attribute*) pertenece a la *ontología de software*. Este modelo representa genéricamente los componentes que forman parte de cualquier producto de software. Es válido tanto en ambientes tradicionales como en entornos más novedosos (como ser CC), ya que presenta un esquema de referencia apto de ser aplicado y/o especializado en diferentes contextos.
- La *métrica de software* (concepto *metric*) pertenece a la *ontología métrica*. Esta ontología define el concepto de métrica de software, brindando un marco de referencia adecuado para la correcta instanciación de métricas específicas.
- La *subcaracterística de calidad* (concepto *subcharacteristic*) pertenece a la *ontología de modelo de calidad de producto de software* basada en el estándar ISO/IEC 25010. Un modelo de calidad puede definirse como el conjunto de factores de calidad (junto con sus relaciones) que proporciona una base tanto para la especificación de requerimientos de calidad como para la evaluación de los componentes software [15]. El MCS propuesto en el estándar (modelado en la ontología) define una taxonomía estandarizada en términos de características y subcaracterísticas.

Aunque individualmente cada modelo representa un dominio específico, en conjunto permiten definir una estrategia de evaluación de calidad aplicable a un producto de software dado (esto es, un QS). En [7] se brinda información detallada sobre los modelos y sus relaciones.

Con el objetivo de especificar un QS aplicable a entornos de CC, en la siguiente sección se especifica la forma en la cual aspectos de calidad propios de este tipo de entornos fueron incorporados al modelo de calidad ISO/IEC 25010. Tal incorporación dio como resultado un nuevo modelo denominado “Modelo de Calidad SaaS”. Tomando en consideración que la existencia de atributos de calidad conlleva a la necesidad de especificar métricas que faciliten su determinación, junto con el modelo de calidad se define un conjunto de métricas aplicables para tal fin. Al finalizar la sección, se muestra como ambos elementos convergen en un QS para CC.

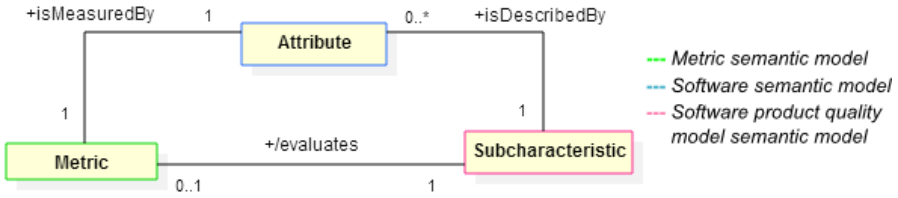

Fig. 2. Relaciones modeladas entre los elementos componentes de una tripleta QS.

3.1 Modelo de Calidad para Software-as-a-Service

Modelo de Calidad de Producto ISO/IEC 25010. El modelo de calidad propuesto en ISO/IEC 25010 clasifica la calidad de producto en base a un conjunto estructurado

de características, subcaracterísticas y atributos. Una característica representa una cualidad externa (propiedad experimentada por el usuario) que se obtiene por medio del balance de las subcaracterísticas que la componen. Una subcaracterística se manifiesta cuando el software es utilizado como parte de un sistema, pudiendo medirse de forma externa o interna. Cada subcaracterística se divide en atributos. Un atributo es una entidad que puede ser verificada o medida sobre el producto de software.

El modelo identifica ocho características que cubren las propiedades tanto estáticas como dinámicas del software, a saber: *funcionalidad* (*functional suitability*), *rendimiento* (*performance efficiency*), *compatibilidad* (*compatibility*), *usabilidad* (*usability*), *confiabilidad* (*reliability*), *seguridad* (*security*), *mantenibilidad* (*maintainability*) y *portabilidad* (*portability*). Para cada una de estas características, el estándar detalla un conjunto de subcaracterísticas asociadas. Por ejemplo, para la característica *rendimiento* el modelo especifica tres subcaracterísticas: *capacidad* (*capacity*), *comportamiento temporal* (*time behaviour*) y *utilización de recursos* (*resource utilization*). Sin embargo, no ocurre lo mismo en el caso de los atributos asociados a las subcaracterísticas. Debido a que la aplicabilidad de los atributos varía de acuerdo al tipo de producto de software a analizar, el estándar no da cuenta de los atributos asociados a las subcaracterísticas definidas. La incorporación de los atributos aplicables a una determinada clase de producto debe realizarse por fuera del modelo. Para obtener mayor información acerca de las características/subcaracterísticas del modelo de calidad de producto de software del estándar ISO/IEC 25010, se sugiere remitirse a [8].

Tabla 1. Recopilación de las principales propiedades de calidad a evaluar en entornos SaaS.

PROPIEDAD	DESCRIPCIÓN	REF.
Reusabilidad - Reusability -	Capacidad de los componentes de un servicio de ser reutilizados en la construcción de otras aplicaciones del mismo tipo.	[17, 20]
Disponibilidad - Availability -	Si el SaaS no se encuentra disponible, los consumidores no podrán hacer uso de sus funcionalidades.	[16-18]
Escalabilidad - Scalability -	Habilidad de aumentar/disminuir de forma ágil y rápida los recursos asignados al entorno según el nivel de demanda.	[16, 20,21]
Personalización de servicios - Service customizability -	Capacidad del SaaS de ser modificados de acuerdo a requerimientos específicos de los consumidores.	[17, 20]
Uniformidad funcional - Functional feature commonality -	Nivel de homogeneidad de las características funcionales del servicio de software.	[17]
Uniformidad no funcional - Non-funct. feature commonality -	Nivel de homogeneidad de las características no funcionales del servicio.	[17]
Uso de infraestructura - Infrastructure utilization -	Cantidad de recursos de infraestructura reservados por el servicio que han sido efectivamente utilizados.	[21]
Tiempo de invocación - Invocation time -	Tiempo que tarda el sistema SaaS en ejecutar la invocación de un servicio por el cual el usuario está pagando.	[17, 18]
Estabilidad del servicio - Service stability -	Grado en el cual el servicio de software funciona sin que se den fallas y/o errores.	[17]
Precisión del servicio - Service accuracy -	Exactitud que posee el servicio de software al dar respuesta a una solicitud de usuario específica.	[18]
Cobertura de recursos - Resource coverage -	Promedio de recursos asignados en relación a la cantidad de recursos solicitados.	[21]
Robustez del servicio - Robustness of service -	Probabilidad de que el servicio de software pueda ser utilizado por los consumidores en un instante de tiempo dado.	[21]

Relevamiento de la Calidad en SaaS. Diversos autores han enfatizado la importancia de relevar la calidad en SaaS [16–21]. Aunque sus líneas de investigación difieren, sus trabajos han dado lugar a múltiples propiedades relevantes en CC (algunas de ellas relacionadas con modelos de calidad específicos). La Tabla 1 resume los aspectos claves que deben ser considerados al analizar la calidad en ambientes SaaS.

Modelo de Calidad para SaaS. Debido a que el modelo propuesto en el estándar ISO/IEC 25010 identifica un conjunto de factores de calidad abstractos que pueden ser utilizados en diferentes productos de software pero que, al mismo tiempo, pueden refinarse para su aplicación en un producto o entorno de software particular; es posible clasificarlo como un modelo de calidad mixto [22].

En este contexto, haciendo uso de las propiedades SaaS detalladas en la sección precedente y siguiendo la propuesta de Kläs, M. et al. [23], se especificó un nuevo modelo de calidad que se ajusta a las características particulares de los entornos SaaS. El modelo resultante se esquematiza en la Figura 3. Este modelo mantiene las características y subcaracterísticas del estándar y las combina con nuevas subcaracterísticas y atributos derivados de las propiedades SaaS identificadas. Esto da como resultado una taxonomía de calidad más específica, refinada y completa que la original.

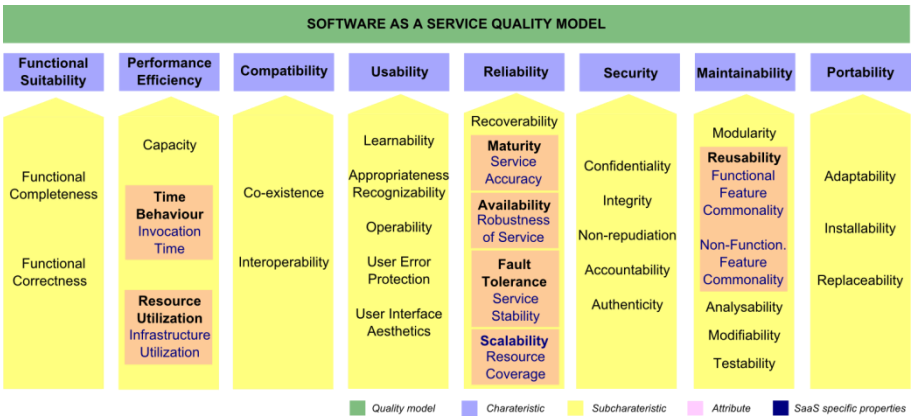

Fig. 3. Modelo de calidad SaaS.

Incorporación de Subcaracterísticas de Calidad. Las propiedades *reusabilidad*, *disponibilidad* y *escalabilidad* se integraron como parte del modelo bajo la forma de subcaracterísticas. La estructura original no se ve afectada por la existencia de *reusability* y *availability*, ya que existe una correspondencia directa entre estas propiedades y las subcaracterísticas propias de la taxonomía (ubicadas como parte de *maintainability* y *reliability*, respectivamente). Sin embargo, no ocurre lo mismo en el caso de *scalability*. Teniendo en cuenta que la propiedad *escalabilidad* (tanto de producto como de servicio) no se encuentra definida en el modelo original, su incorporación al modelo SaaS se logra identificando una nueva subcaracterística. La subcaracterística *scalability* se anexa dentro de la característica de calidad *reliability*. En términos generales, la *escalabilidad* es la capacidad de un sistema, red, proceso o servicio de manejar grandes cargas de trabajo y/o de ser fácilmente ampliado. Considerando que

la *confiabilidad* es la capacidad de un sistema o componente de software de llevar a cabo sus funciones cuando se lo utiliza bajo condiciones y periodos de tiempo determinados, es claro que posibilidad de adaptarse a los cambios en función de la demanda es un aspecto a evaluar en esta característica. Luego, la *escalabilidad* forma parte de la *confiabilidad*.

Incorporación de Atributos de Calidad. Las propiedades *tiempo de invocación* y *utilización de infraestructura* se integraron al modelo en forma de atributos de calidad relacionados con las subcaracterísticas *comportamiento temporal* y *utilización de recursos* respectivamente (ambas incluidas dentro de la característica *rendimiento*). Dado que *time behaviour* refiere a la evaluación de tiempos de respuesta, tiempos de procesamiento y tasas de throughput en condiciones de ejecución específicas; resulta natural incluir la propiedad *invocation time* como atributo de esta subcaracterística. Lo mismo ocurre con la propiedad *infrastructure utilization* ya que, la subcaracterística *resource utilization* trata con la cantidad y tipo de recursos usados cuando el software es ejecutado en contextos específicos mientras que la propiedad SaaS propone examinar de qué forma son utilizados los recursos de la infraestructura cloud.

Dentro de la característica *confiabilidad* se incorporaron cuatro atributos, cada uno de los cuales se asocia a una subcaracterística particular. La propiedad *precisión del servicio* se integró como atributo de *madurez*. La subcaracterística *maturity* refiere a la capacidad del sistema o servicio de satisfacer las necesidades de confiabilidad en condiciones normales de ejecución. En este sentido, en un contexto ideal, un servicio maduro debería tener mayor exactitud en sus respuestas que un servicio que recién inicia. De esta forma, *service accuracy* corresponde a un descriptor de dicha subcaracterística. Por otra parte, la propiedad *robustez del servicio* se incorporó como atributo de *disponibilidad*. Teniendo en cuenta que *availability* trata la capacidad del sistema o servicio de estar operativo y accesible para su uso en un instante de tiempo dado, es claro que la propiedad *robustness of service* forma parte de dicha subcaracterística. En contraposición, la propiedad *estabilidad del servicio* se asoció con la subcaracterística *tolerancia a fallos*. Dicha subcaracterística corresponde a la habilidad del software para operar según lo previsto ante la presencia de fallos (tanto de hardware como de software). Dado que *service stability* propone analizar el comportamiento del servicio sin fallas, es apropiado colocar dicha propiedad como atributo de *fault tolerance*. En último lugar, la propiedad *cobertura de recursos* (*resource coverage*) se integró como atributo de la subcaracterística *escalabilidad* (*scalability*), ya que la cantidad de recursos que el servicio ha utilizado para alojar componentes funcionales en relación a la cantidad de recursos solicitados para tales fines, permite dar cuenta del nivel de escalabilidad que el servicio ha tenido durante su ejecución.

Finalmente, como parte de la subcaracterística *reusabilidad* (incluida en la característica *mantenibilidad*) se incorporaron tres atributos de calidad: *uniformidad en las características funcionales, uniformidad en las características no funcionales* y *personalización de servicios*. Debido a que la semejanza entre características (tanto funcionales como no funcionales) facilita el reúso de componentes y que un alto grado de personalización da lugar a un mayor reúso del servicio, es evidente que las propiedades *functional feature commonality, non-functional feature commonality* y *service customizability* son aspectos vinculados a la subcaracterística *reusability*.

Tabla 2. Métricas SaaS propuestas para relevar la calidad de los servicios de software.

MÉTRICA	DESCRIPCIÓN	
Comportamiento temporal percibido por el usuario - Time behavior from user perspective -	*Significado*	Relación entre el tiempo de ejecución y el tiempo total desde que se invocó el servicio hasta que finalizó la operación.
	Ecuación	TBU = ET / TSIT
	Variables de Cálculo	ET = Tiempo de procesamiento de una operación.
		TSIT = Tiempo transcurrido desde que se realizó la solicitud de la operación hasta que emitió su respuesta.
Uso de recursos de hardware - Hardware resources utilization -	*Significado*	Relación entre los recursos asignados y los recursos predefinidos.
	Ecuación	HRU = AR / PR
	Variables de Cálculo	AR = Cantidad de recursos de hardware asignados.
		PR = Cantidad de recursos de hardware predefinidos.
Precisión en respuestas - Replies accuracy -	*Significado*	Grado en el cual la respuesta a una solicitud es correcta.
	Ecuación	RA = CR / TR
	Variables de Cálculo	CR = Cantidad de respuestas correctas emitidas por el SaaS.
		TR = Cantidad total de solicitudes que han sido respondidas.
Robustez del servicio - Service robustness -	*Significado*	Relación entre el tiempo que el servicio está disponible para ser invocado y el tiempo total de operación.
	Ecuación	SR = AT / TT
	Variables de Cálculo	AT = Tiempo durante el cual el servicio puede ser invocado.
		TT = Cantidad total de tiempo que el sistema estuvo operativo.
Cobertura de tolerancia de defectos - Coverage of fault tolerance -	*Significado*	Razón de defectos (faults) que no son fallas (failures).
	Ecuación	CFT = FNF / TF
	Variables de Cálculo	FNF = Cantidad de defectos que no causan fallas.
		TF = Cantidad total de defectos.
Cobertura de recuperación de fallas - Coverage of failure recovery -	*Significado*	Proporción de fallas reparadas en un período de tiempo dado.
	Ecuación	CFR = RF / Tf
	Variables de Cálculo	RF = Cantidad de fallas reparadas.
		Tf = Cantidad total de fallas.
Cobertura de escalabilidad - Coverage of Scalability -	*Significado*	Recursos asignados en relación a la cantidad de recursos solicitados.
	Ecuación	$COS = 1/k * \sum AR_i / TR_i$
	Variables de Cálculo	AR_i = Cantidad de recursos asignados para la i-ésima solicitud.
		TR_i = Total de recursos solicitados para la i-ésima solicitud.
		k = Cantidad de solicitudes realizadas al servicio.
Similitud funcional - Functional commonality -	*Significado*	Ajuste de los requerimientos planteados al formular el servicio y las características funcionales resultantes.
	Ecuación	$FC = 1/n * \sum RFC_i / TR_i$
	Variables de Cálculo	RFC_i = Cantidad de requerimientos que refieren a la i-ésima característica funcional.
		TR_i = Total de requerimientos analizados en el dominio.
		n = Cantidad de características funcionales incluidas en el SaaS.
Similitud no funcional - Non-functional commonality -	*Significado*	Ajuste de los requerimientos planteados al formular el servicio y las características no funcionales resultantes.
	Ecuación	$NFC = 1/m *\sum RNFC_i / TR_i$
	Variables de Cálculo	$RNFC_i$ = Cantidad de requerimientos que refieren a la i-ésima característica no funcional.
		TR_i = Total de requerimientos analizados en el dominio.
		m = Cantidad de características no funcionales del SaaS.
Cobertura de variabilidad - Coverage of Variability -	*Significado*	Proporción de variantes del dominio incluidas en el SaaS.
	Ecuación	$CV = VP_{SaaS} / VP_D$
	Variables de Cálculo	VP_{SaaS} = Cantidad de puntos de variación que existen en el SaaS.
		VP_D = Cantidad total de puntos de variación que existen en el dominio asociado al SaaS.

3.2 Métricas SaaS

Las métricas de software que permiten evaluar la capa de aplicación de una arquitectura cloud se denominan *métricas SaaS*. Existen cientos de métricas de software aplicables a costos, planificación, desarrollo, gestión, promoción y adquisición de servicios de software [24], por lo que la selección de las métricas a utilizar para relevar la calidad no es una trivialidad. Con este objetivo en mente, la Tabla 2 resume el conjunto de métricas elegidas para evaluar la calidad de servicios de software.

3.3 Esquema de Calidad SaaS

En base a la descripción ontológica formulada en la Figura 2, tomando en consideración el modelo de calidad propuesto junto con las métricas de software detalladas con anterioridad, se especificó un esquema de calidad genérico aplicable a servicios de software (Tabla Nº3).

El esquema resultante propone para cada atributo asociado a las subcaracterísticas de calidad del modelo diseñado, una métrica de software SaaS. De esta manera, establece no sólo los aspectos de interés que deben ser relevados para determinar la calidad de un servicio de software sino que además fija la forma en la cual estos aspectos deben ser evaluados en relación a otras propiedades de calidad.

Tabla 3. Resumen del esquema de calidad SaaS resultante.

SUBCARACTERÍSTICA	ATRIBUTO	MÉTRICA
Time behaviour	Invocation time	Time behavior from user perspective
Resource utilization	Infrastructure utilization	Hardware resources utilization
Maturity	Service accuracy	Replies accuracy
Availability	Robustness of service	Service robustness
Fault tolerance	Service stability	Coverage of fault tolerance
		Coverage of failure recovery
Scalability	Resource coverage	Coverage of Scalability
Reusability	Service customizability	Coverage of Variability
	Functional feature commonality	Functional commonality
	Non-functional feature commonality	Non-functional commonality

4 Beneficios y Utilidad

Mantener la calidad a lo largo del proceso de desarrollo es una tarea difícil. Frecuentemente, debido a que no existen mecanismos de soporte para las decisiones de calidad, la especificación de estos atributos se pierde entre las diferentes etapas de desarrollo. Suele suceder además que el equipo de trabajo desconoce los vínculos existentes entre los distintos atributos, las entidades que estos abarcan y la influencia de los mismos en la calidad resultante del producto o servicio. En este contexto los QS dan

lugar a nuevas estrategias de documentación para la información de calidad básica relacionada con un software específico.

El QS presentado puede aplicarse en servicios de software (nuevos o existentes) no sólo para el entendimiento y documentación de la calidad, sino también para su medición. Su formalización corresponde a una instanciación de la propuesta ontológica presentada en [7], lo que posibilita el análisis de cada uno de sus elementos en relación a la información disponible. De esta manera, la propuesta posibilita el estudio sistemático de servicios de software brindando además la posibilidad de comparar resultados de acuerdo a criterios uniformes de decisión.

5 Conclusiones

No se dará un aumento significativo de la calidad del software hasta que exista un modelo global de calidad aplicable a todos los productos de software disponibles [25]. En este trabajo se ha presentado un esquema de calidad que sintetiza las relaciones existentes entre los diferentes aspectos de calidad de servicios de software y las métricas disponibles para su relevamiento.

La definición del esquema resultante se basa en un modelo de calidad de producto de software existente, el cual es ampliado y modificado de acuerdo a propiedades específicas de este tipo de servicios. Sin embargo, esta especificación no es taxativa. El modelo de calidad propuesto puede ser ampliado con nuevas características, subcaracterísticas y/o atributos de calidad, mientras que el conjunto de métricas sugeridas puede ser reemplazado de acuerdo a necesidades específicas. De esta manera, el esquema desarrollado es un modelo mixto que sienta las bases sobre los aspectos de calidad que pueden/deben ser analizados en un servicio de software junto con la forma en la cual pueden ser relevados, no excluyendo a futuro la incorporación de nuevos elementos componentes.

Referencias

1. Deissenboeck, F., Juergens, E., Lochmann, K., Wagner, S.: Software quality models: Purposes, usage scenarios and requirements. In: ICSE Workshop on Software Quality, 2009. WOSQ ’09. pp. 9–14 (2009).
2. IEEE Std. 1061-1998 IEEE Standard for a Software Quality Metrics Methodology – Description (1998).
3. Pressman, R.: Software Engineering: A Practitioner´s Approach. McGraw-Hill (2010).
4. Albin, S.: The art of software architecture: design methods and techniques. John Wiley & Sons (2003).
5. Boukouchi, Y., Marzak, A., Benlahmer, H., Moutachaouik, H.: Comparative Study of Software Quality Models. In: IJCSI International Journal of Computer Science Issues. pp. 309–314 (2013).
6. AL-Badareen, A.B., Selamat, M.H., Jabar, M.A., Din, J., Turaev, S.: Software Quality Models: A Comparative Study. In: Zain, J.M., Mohd, W.M. bt W., and El-Qawasmeh, E. (eds.) Software Engineering and Computer Systems. pp. 46–55. Springer Berlin Heidelberg (2011).

7. Blas, M.J., Gonnet, S.: An Ontological Approach to Analyze the Data Required by a System Quality Scheme. Proc. Argent. Symp. Ontol. Their Appl. SAOA 2015. 111–121 (2015).
8. ISO/IEC 25010:2011 - System and software quality models (2011).
9. Hu, F., Qiu, M., Li, J., Grant, T., Taylor, D., McCaleb, S., Butler, L., Hamner, R.: A Review on Cloud Computing: Design Challenges in Architecture and Security. CIT J. Comput. Inf. Technol. 19, 25–55 (2011).
10. Khan, A.N., Mat Kiah, M.L., Khan, S.U., Madani, S.A.: Towards secure mobile cloud computing: A survey. Future Gener. Comput. Syst. 29, 1278–1299 (2013).
11. Mell, P.M., Grance, T.: SP 800-145. The NIST Definition of Cloud Computing. National Institute of Standards & Technology, Gaithersburg, MD, United States (2011).
12. Fehling, C., Leymann, F., Retter, R., Schupeck, W., Arbitter, P.: Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer Science & Business Media (2014).
13. Fan, H., Hussain, F.K., Younas, M., Hussain, O.K.: An integrated personalization framework for SaaS-based cloud services. Future Gener. Comput. Syst. 53, 157–173 (2015).
14. Fonseca, B.: SaaS Benefits Starting to Outweigh Risks. Computerworld. 42, 12–13 (2008).
15. Carvallo, J., Franch, X., Grau, G., Quer, C.: QM: A Tool for Building Software Quality Models. Presented at the RE -INTERNATIONAL CONFERENCE- (2004).
16. Breu, R., Kuntzmann-Combelles, A., Felderer, M.: New Perspectives on Software Quality IEEE Software, 2014, no 1, p. 32-38, (2014).
17. Lee, J.: A quality model for evaluating software-as-a-service in cloud computing. Presented at the 7º International Conference on Software Engineering Research, Management and Applications (2009).
18. Zhao, B., Zhu, Y.: Formalizing and validating the web quality model for web source quality evaluation. Expert Syst. Appl. 41, 3306–3312 (2014).
19. Khan, I.A., Singh, R.: Quality Assurance And Integration Testing Aspects In Web Based Applications, International Journal of Computer Science, Engineering and Applications, vol. 2, no 3, p. 109 (2012).
20. Wen, P.X., Dong, L.: Quality Model for Evaluating SaaS Service. In: 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT). pp. 83–87 (2013).
21. Gao, J., Pattabhiraman, P., Bai, X., Tsai, W.T.: SaaS performance and scalability evaluation in clouds. In: 2011 IEEE 6th International Symposium on Service Oriented System Engineering (SOSE). pp. 61–71 (2011).
22. Muñoz, C.C., Velthuis, M.G.P., de la Rubia, M.Á.M.: Calidad del producto y proceso software. Editorial Ra-Ma (2010).
23. Kläs, M., Lampasona, C., Münch, J.: Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results. (2013).
24. Singh, R., Bhagat, A., Kumar, N.: Generalization of Software Metrics on Software as a Service (SaaS). In: 2012 International Conference on Computing Sciences (ICCS). pp. 267–270 (2012).
25. Dromey, R.G.: A Model for Software Product Quality. IEEE Trans Softw Eng. 21, 146–162 (1995).