UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

SOA2Cloud: Un marco de trabajo para la migración de aplicaciones SOA a Cloud siguiendo una aproximación dirigida por modelos

Tesis de Máster en Ingeniería del Software,
Métodos Formales y Sistemas de Información
(MISMFSI)

Grupo de Ingeniería del Software y Sistemas de Información (ISSI)
Departamento de Sistemas Informáticos y Computación (DSIC)
Universidad Politécnica de Valencia (UPV)

Febrero, 2014

Miguel Botto Tobar

Director:
Dr. Emilio Insfrán Pelozo

SOA2Cloud:
Un marco de trabajo para la migración de aplicaciones SOA a Cloud siguiendo una aproximación dirigida por modelos

Miguel Ángel Botto Tobar
Impreso en Valencia, España
2014

AGRADECIMIENTOS

A mi director de tesis Emilio Insfrán por todo su apoyo, consejos, enseñanzas y ayuda en la elaboración de este trabajo.

A mi país por brindarme la oportunidad para prepararme académicamente, y permitir mi desarrollo profesional proporcionándome los medios necesarios para cumplir este propósito.

A mi mami (*abuela*) Amada, a mi madre Yoconda, y a mi tío Pascual por el apoyo que me ofrecieron, por la formación y por fomentar en mí el espíritu de querer aprender siempre.

Por último, a mis compañeros becarios del Gobierno de Ecuador y del Grupo ISSI.

CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen

Las aplicaciones software son consideradas actualmente un elemento esencial e indispensable en toda actividad empresarial, por ejemplo, intercambio de información y motor de redes sociales. Sin embargo, para su construcción y despliegue se utilizan todos los recursos que estén disponibles en ubicaciones remotas y accesibles de la red, lo que conlleva a realizar operaciones ineficientes en el desarrollo y despliegue, y enormes gastos en la adquisición de equipos de TI.

La presente tesina de máster pretende contribuir a la mejora del contexto anterior proponiendo SOA2Cloud, un marco de trabajo para la migración de aplicaciones basadas en SOA a entornos Cloud, haciendo uso de la aproximación del Desarrollo de Software Dirigido por Modelos (DSDM). SOA2Cloud tiene la finalidad de proporcionar mecanismos para la migración de aplicaciones SOA especificadas a través del estándar SoaML de la OMG, incorporando los Acuerdos de Nivel de Servicios (SLA) a entornos Cloud Computing. El marco de trabajo propuesto hace uso de un modelo de la aplicación SOA, definido conforme a SoaML, y un modelo de acuerdos de servicios definido conforme a un metamodelo genérico de SLA para la generación de un modelo conforme a un metamodelo para aplicaciones Cloud, a través de transformaciones de modelos. Este modelo generado, es sometido a una nueva transformación de modelos, para la obtención del modelo de la plataforma Azure, conforme a su metamodelo genérico construido para este trabajo de investigación. Una vez concluidas las transformaciones de modelos, el modelo obtenido es sometido a una transformación de modelo a texto para la obtención del código fuente, y de esta forma ser testeado y desplegado en la plataforma seleccionada para este trabajo de investigación Windows Azure.

Esta propuesta se apoya en un amplio estudio del estado del arte, realizado mediante la conducción de un mapeo sistemático, acerca de las estrategias de migración de aplicaciones SOA a entornos Cloud Computing. Los resultados obtenidos aportaron de una forma significativa en la definición del proceso de migración en el marco de trabajo.

Finalmente, se desarrolló un ejemplo de aplicación que muestra la viabilidad de nuestro enfoque. Este ejemplo muestra en detalle como el marco de trabajo para la migración de aplicaciones SOA a entornos Cloud propuesto. Los resultados muestran que nuestra propuesta permitiría mejorar el enfoque de algunos investigadores y profesionales del área al realizar migraciones de aplicaciones SOA a entornos Cloud, haciéndolas a través de este marco de trabajo que aprovecha los beneficios del Desarrollo de Software Dirigido por Modelos.

Abstract

Software applications are currently considered an element essential and indispensable in all business activities, for example, information exchange and social network. Nevertheless, for their construction and deployment to use all the resources that are available in remote and accessible locations on the network, which leads to inefficient operations in development and deployment, and enormous costs in the acquisition of IT equipment.

The present master thesis aims to contribute to the improvement of the previous context proposing SOA2Cloud, a framework for migration of applications based on SOA to Cloud environments, making use Model-Driven Software Development approach. SOA2Cloud aims to provide mechanisms for the migration of SOA applications specified through the OMG SoaML standard, incorporating the service level agreements (SLA) to Cloud Computing environments. The framework proposed to makes to use a SOA application model, defined to conform to SoaML metamodel, and a model of service level agreements defined according to SLA generic metamodelo, to generation a model according to Cloud metamodel, through models transformations. This generated model, over again to model transformation, for obtaining the model Azure platform, according to their generic metamodel built for this research work. At the conclusion model transformations, the obtained model over again a model to text transformation to obtain the source code, and thus be tested and deployed in the platform selected for this research Azure work.

This proposal is based on a comprehensive study of the state of the art, made by conducting a systematic mapping, about strategies for migrating applications SOA to Cloud Computing environments. The results contributed in a meaningful way in the definition of the process of migration in the framework.

Finally, an example of application that shows the feasibility of our approach was developed. This example demonstrates in detail as the framework for migrating applications proposed SOA to Cloud environments. The results show that our proposal may allow improving the strategy mainly used by researchers and professionals in the area to perform migrations of SOA applications into Cloud environments. This will be through our proposed migration framework which exploits the benefits of Model-Driven Software Development.

Capítulo 1. Introducción

Los sistemas software son cada vez más complejos, y el éxito de su desarrollo ya no depende del esfuerzo individual y del heroísmo, y sólo puede lograrse mediante el uso de un proceso bien definido. Muchos de los enfoques de desarrollo de software se han introducido para hacer frente a los problemas de complejidad, lo que reduce el tiempo y el coste de desarrollo.

Sin embargo, estas técnicas son de poca utilidad sin unas directrices bien definidas que proporcionen la ayuda necesaria a los ingenieros de software en el proceso de desarrollo de software.

1.1. Motivación

Con la aparición de Internet y su rápida expansión, se ha creado un medio ideal para el desarrollo de aplicaciones que realizan intercambio de información, transacciones electrónicas, venta de productos y servicios en línea, entre otras funcionalidades. Esto ha impulsado a las organizaciones a aumentar su interacción con clientes y otras organizaciones.

El reto de la Ingeniería del Software es la construcción y despliegue de aplicaciones utilizando los recursos que estén disponibles en ubicaciones remotas y accesibles de la red. Este desafío ha sido abordado desde diferentes enfoques, tales como la Arquitectura Orientada a Servicios (SOA), Cloud Computing, entre otros; que han tratado en los últimos años de mejorar la implementación de software y la productividad en la industria de TI.

La arquitectura orientada a servicios (SOA) como un marco de referencia o paradigma, en el ámbito de la arquitectura software en el desarrollo de aplicaciones informáticas, se basa en el desarrollo de servicios y su reutilización, con un negocio claro y una funcionalidad conocida, independiente y sin acoplamiento, ofrecidos a través de un conjunto de servicios claramente definidos e interfaces acopladas que puede ofrecer de la misma manera en otras aplicaciones [10].

SOA está relacionada con el uso de estándares abiertos como: XML, SOAP, WSDL, UDDI, BPEL y BPMN por mencionar algunos de los más conocidos, que facilitan la integración de servicios creados e implementados en diferentes contenedores de aplicaciones J2EE o .NET.

Cloud Computing o computación en la nube está vinculado a un modelo novedoso de oferta en la red de los distintos tipos de recursos TI (a partir de recursos de informática básica, elementos de memoria, almacenamiento y redes, plataformas para construir, desplegar y ejecutar aplicaciones, o incluso sus propias aplicaciones o datos) que forman parte de un conjunto de recursos que se pueden asignar, aprovisionarse y ser entregados rápidamente, con una mínima interacción entre el cliente y el proveedor de servicios [105].

Se considera por lo general qué la automatización del abastecimiento, la gestión y el control del servicio prestado al cliente, los recursos de múltiples capacidades que ofrece, la ubicuidad en el acceso a ellos, la facilidad de escalabilidad de las capacidades ofrecidas y midiendo la posibilidad de utilizarlos, son elementos esenciales para considerar que un servicio se ofrece bajo el modelo Cloud.

Cloud Computing expande a SOA añadiendo escalabilidad y Computación GRID [118]. La escalabilidad es requerida cuando el software se utiliza como un servicio debido al hecho de que a medida que más y más se crean una instancia del servicio, se utilizan los recursos de hardware y la escalabilidad se convierte en un requisito. Mediante la adopción de Computación GRID, la potencia de procesamiento para cada servicio se puede aumentar también.

Cloud Computing no está asociado a ninguna tecnología en particular, protocolo o proveedor. Permite a las aplicaciones en la nube dar servicio a los consumidores (por lo general a través de sitios web, aplicaciones cliente, etc.) que aseguran que el servicio tendrá un único punto de acceso y toda a escala, la computación paralela, la virtualización y lo que la tecnología use en la parte final será transparente para el cliente. Desde esta perspectiva, la computación en nube es un modelo en vez de una arquitectura.

Este trabajo ha sido definido teniendo en cuenta la necesidad de muchas empresas tanto públicas como privadas, que realizan operaciones ineficientes en la construcción y despliegue de sus aplicaciones, y enormes gastos en la adquisición de equipos de TI. Con la computación disponible como un servicio en el paradigma de Cloud Computing, muchas compañías consideran a Cloud Computing como una solución de suministros de servicios. Por lo tanto, muchas aplicaciones SOA son migradas al Cloud, de aquí la necesidad de construir un marco de trabajo que permita migrar aplicaciones desarrolladas en SOA a entornos Cloud Computing aplicando el paradigma del Desarrollo de Software Dirigido por Modelos.

1.2. Objetivos

El principal objetivo de la investigación es el desarrollo un marco de trabajo (*framework*) que permita migrar aplicaciones basadas en Arquitectura Orientada a Servicios (SOA) hacia entornos Cloud Computing, haciendo uso del paradigma del Desarrollo de Software Dirigido por Modelos.

Para lograr este objetivo, se ha definido las siguientes metas:

1. Analizar en profundidad los métodos y técnicas propuestos para migrar aplicaciones basadas en SOA hacia entornos Cloud Computing, con particular énfasis en aquellos que siguen el paradigma del desarrollo de software dirigido por modelos.
2. Analizar la estructura, componentes y servicios de las aplicaciones basadas en SOA y Cloud Computing que los investigadores y desarrolladores consideran relevantes para su desarrollo.

3. Definir un metamodelo para SOA, y un metamodelo para Cloud Computing respectivamente que recoja las características existentes de las aplicaciones de cada una, y cubra carencias encontradas en el estado del arte.
4. Definir un proceso genérico de transformaciones de modelos que dé cobertura a la migración de aplicaciones basadas en SOA a entornos Cloud Computing.
5. Evaluar el rendimiento y escalabilidad de las aplicaciones obtenidas a través del proceso de migración en su ejecución dentro de un entorno Cloud Computing. Tareas de Investigación

El método de investigación a seguir involucra una serie de pasos que nos permitirán obtener los resultados esperados. En este trabajo se ha seguido el método mostrado en la figura 1-1.

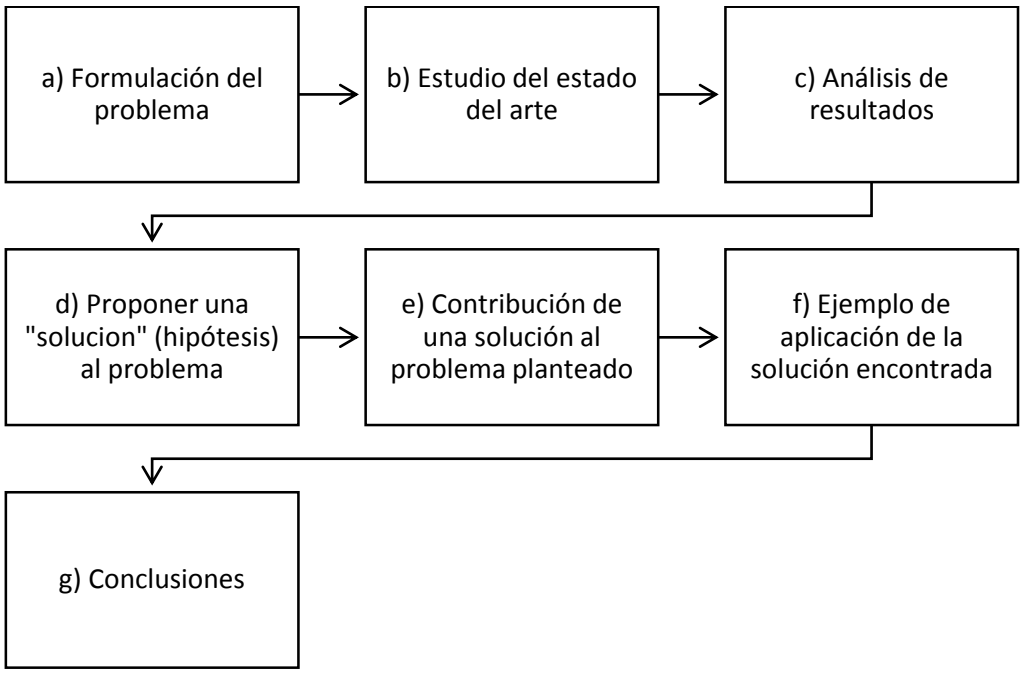

Figura 1-1. Tareas de investigación utilizadas.

A continuación se detalla cada uno de ellos:

a) **Formulación del problema.-** Se realizó una delimitación del mismo para poder tener un punto central y claro al cual atacar en el proceso de la búsqueda de la solución.
b) **Estudio del estado del arte.-** Esta tarea fue realizada mediante un mapeo sistemático por ser un método de investigación para obtener, evaluar e interpretar toda la información disponible que está relacionada con una pregunta de investigación o área de interés. Su propósito es proveer un aseguramiento objetivo en un área de investigación de forma fiable, rigurosa y metodológica.
c) **Análisis de resultados.-** A continuación se analizaron los resultados del estudio del estado del arte y se delimitaron los aspectos que se necesitan investigar en el presente trabajo de acuerdo a la literatura científica actual.

d) Establecimiento de la hipótesis.- Se encontró que existe una falta de estudios referentes a la migración de aplicaciones SOA a entornos Cloud Computing usando el paradigma del desarrollo de software dirigido por modelos, y en base a esto se estableció la hipótesis que se requiere esclarecer en la presente investigación.
e) Contribución de una solución al problema planteado.- Se elaboró un marco de trabajo que sirva de base para la migración de las aplicaciones SOA a entornos Cloud Computing.
f) Ejemplo de aplicación de la solución encontrada.- En este paso se aplicó el implemento el marco de trabajo migrando aplicaciones SOA a entornos Cloud Computing.
g) Conclusiones.- Se obtuvieron las principales conclusiones relacionadas al proceso de migración aplicaciones SOA a Cloud Computing.

1.3. Contexto de la Tesina

Esta investigación ha sido desarrollada dentro del grupo de investigación Ingeniería del Software y Sistemas de Información (ISSI), y ha sido garantizada con una Beca del "Programa Convocatoria Abierta 2011" por la Secretaria Nacional de Ciencia Tecnología e Innovación "Senescyt" del Gobierno de Ecuador.

1.4. Estructura del Documento

El presente documento está organizado en 7 capítulos, y su estructura se detalla de la siguiente forma:

Capítulo 1: Introduce al lector la motivación, objetivo, contexto y las metas que se pretende alcanzar con esta investigación.

Capítulo 2: Presenta el estado del arte acerca de los temas relacionados con la tesina: una revisión sistemática acerca de las estrategias de migración de aplicaciones basadas en SOA a entornos Cloud Computing.

Capítulo 3: Recoge brevemente los conceptos más importantes sobre la Arquitectura Orientada a Servicios, Cloud Computing y el paradigma del Desarrollo de Software Dirigido por Modelos, historia, estado actual, los modelos más utilizados para su representación, etc.

Capítulo 5: Presenta la generación de aplicaciones Cloud a partir de modelos SoaML y SLA, así como el proceso de migración y las respectivas correspondencias entre los elementos y sus transformaciones.

Capítulo 6: Describe un ejemplo de aplicación, el mismo que se destina a ilustrar la viabilidad de nuestra propuesta, y fue desarrollado aplicando el proceso de migración presentado en el capítulo anterior.

Capítulo 7: Describe las conclusiones generales, los trabajos futuros, las publicaciones generadas y los resultados obtenidos.

Existen tres apéndices:

- Apéndice A contiene la bibliografía del mapeo sistemático.
- Apéndice B contiene los atributos de calidad de servicio de SOA.
- Apéndice C contiene los posibles contenidos de un SLA.
- Apéndice D contiene las transformaciones entre modelos.
- Apéndice E contiene las transformaciones de modelo a texto.

Capítulo 2. Estado del Arte

Este capítulo describe los estudios realizados con el fin de obtener una perspectiva global sobre la migración de aplicaciones basadas en Arquitectura Orientada a Servicios (SOA) hacia entornos Cloud Computing, y que han sido de mucha utilidad para definir los objetivos de la presente tesina de máster.

En la Sección 2.1 se presenta brevemente una descripción del método aplicado para la recolección de la información que será útil en la sección 2.2.

En la Sección 2.2 se presenta un mapeo sistemático sobre las estrategias de migración de aplicaciones basadas en Arquitectura Orientada a Servicios (SOA) hacia entornos Cloud Computing existentes, y las ventajas que aportan. Luego de este análisis, se plantea una discusión sobre los principales hallazgos, las limitaciones encontradas a lo largo de esta investigación, y las implicaciones de los resultados obtenidos de cara al ámbito académico e industrial.

En la Sección 2.3 se presentan brevemente los principales grupos de investigación académicos e iniciativas industriales vinculados al desarrollo de métodos y técnicas que permitan migrar aplicaciones basadas en Arquitectura Orientada a Servicios (SOA) hacia entornos Cloud Computing.

2.1. Mapeo Sistemático

Un mapeo sistemático es una metodología frecuentemente usada en investigación médica y que ha sido recomendada para áreas de investigación donde existe una falta de estudios primarios relevantes de alta calidad [50].

Un mapeo sistemático en el área de la ingeniería del software es un método para construir un esquema de clasificación y estructurar un campo de interés en la ingeniería del software. El análisis de los resultados se centra en frecuencias de publicaciones por categorías [91].

Los resultados de un mapeo sistemático pueden identificar áreas en las cuales es necesario conducir una revisión sistemática.

Las principales diferencias entre un mapeo sistemático y una revisión son [50]:

- Los mapeos son generalmente manejados por preguntas de investigación extensas y a menudo responden múltiples preguntas de investigación.
- Los términos de búsqueda para mapeos sistemáticos suelen ser enfocados a un nivel más alto que las revisiones sistemáticas y retornan un gran número de estudios. Para un mapeo, sin embargo, es menos problemático trabajar con abundantes resultados.
- El proceso de extracción de datos para los mapeos sistemáticos es además mucho más amplia que la extracción de datos para revisiones sistemáticas y puede ser definido como un estado de clasificación o categorización.

- El propósito de este estado es clasificar los estudios con suficiente detalle para responder las preguntas de investigación más amplias y para identificar estudios para las subsecuentes revisiones.
- El estado de análisis para un estudio de mapeo sistemático consiste en resumir los datos de acuerdo a responder la pregunta de investigación. No es necesario incluir técnicas de análisis a profundidad tales como el meta-análisis y la síntesis narrativa, sino totales y resúmenes. Representaciones gráficas de distribución de estudios por tipos de clasificación pueden ser un tipo efectivo de mecanismo de presentación de resultados.
- La diseminación de los resultados de un mapeo puede ser más limitada que de una revisión sistemática.

Los pasos esenciales de un mapeo sistemático son la definición de la pregunta de investigación, la conducción de la búsqueda de estudios relevantes, los artículos, las palabras claves, los resúmenes y la extracción de datos y mapeos. Cada proceso tiene una salida, el resultado obtenido será un mapeo sistemático.

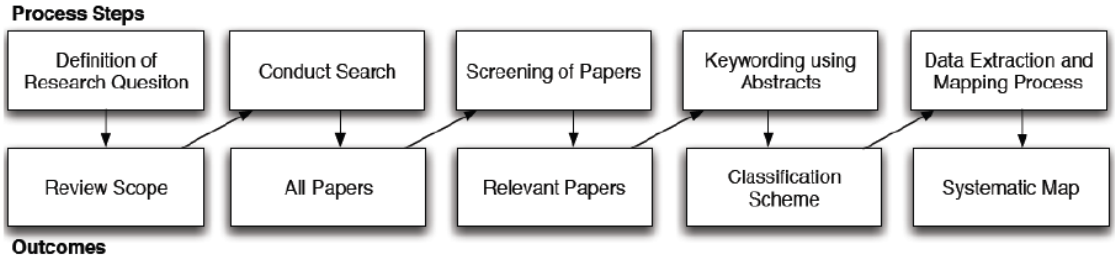

Figura 2-1. Pasos de un Mapeo Sistemático [90].

Una vez revisado el estado del arte, se puede pasar a realizar la propuesta en base a los resultados obtenidos y se puede dar una idea clara del porqué del trabajo propuesto y de los gaps que existentes que podrían conllevar a investigaciones futuras.

2.2. Mapeo Sistemático sobre Estrategias de Migración de Aplicaciones SOA a Cloud Computing

Con el fin de elaborar un estado del arte preciso se ha realizado una exhaustiva investigación acerca de las estrategias de migración de aplicaciones basadas en Arquitectura Orientada a Servicios (SOA) a entornos Cloud Computing. Para ello se ha empleado un método que permita ubicar toda la información actual mediante un Mapeo Sistemático.

Método de Investigación

Se ha elaborado un mapeo sistemático considerando las recomendaciones provistas en trabajos de [50], [91].

Nuestro mapeo sistemático ha sido ejecutado en tres estados: planificación, conducción y reporte de resultados.

2.2.1. Fase de Planificación

Cloud Computing emerge de la crisis económica mundial como una opción para utilizar los recursos informáticos desde un punto de vista más racional. En otras palabras, una forma más barata para tener recursos TI. Un número cada vez mayor de las empresas de migrar sus sistemas a infraestructuras Cloud. Sin embargo, no ha habido mucha atención para proporcionar suficiente apoyo al proceso. Dado que los proyectos de migración probablemente encontrarán varias clases de desafíos, es importante identificar y compartir el proceso y requerimientos logísticos de proyectos de migración con el fin de construir un cuerpo de conocimientos de proceso adecuado, métodos y herramientas.

Existen pocos estudios que abordan la migración de aplicaciones a entornos Cloud. Sin embargo, no hay literatura disponible para el proceso de migración de aplicaciones basadas en SOA a entornos Cloud. En esta sección, se presenta un mapeo sistemático que categoriza y resume la información existente sobre una pregunta de investigación de una manera imparcial. La meta de este apartado es de conocer cuáles son las estrategias utilizadas para migrar aplicaciones a entornos Cloud.

Tanto los estudios de mapeo sistemático y Cloud Computing son muy nuevos en el campo de la ingeniería de software. Esta coincidencia hace difícil encontrar cualquier estudio mapeo sistemático existente que investigara la migración a la nube. Yunus presenta los costos y riesgos de la migración de aplicaciones [122], mientras que Louridas [57] discute la migración de aplicaciones a la nube, examina las principales características de las ofertas de nube basadas en la taxonomía de [95].

Khajeh-Hosseini, Greenwood y Sommerville [49] ilustran los beneficios potenciales y los riesgos asociados con la migración de un sistema de Amazon EC2 desde una amplia variedad de perspectivas de los actores en toda la empresa, trascendiendo así el análisis típico, pero estrecho, financiero y técnico ofrecidos por los proveedores.

Kothari y Arumugam introducen directrices para evaluar la viabilidad de migrar aplicaciones a la nube y sugieren una estrategia general de migración para aplicaciones [53], mientras que Sattaluri aborda los diferentes aspectos que deben considerarse durante la migración de aplicaciones [99]. Por otro lado, Mossburg enumera cuatro puntos importantes que conducen a una migración exitosa a la nube [74].

Estas obras son diferentes a nuestro trabajo en el sentido de que sólo proporcionan instrucciones muy generales mientras que nuestro trabajo da más detalles sobre qué hacer y cómo hacerlo para la migración de aplicaciones. Además, estas guías sólo son aplicables para el bajo nivel de IaaS (no se aplican a PaaS).

Con respecto a la metodología, los proveedores de servicios Cloud Computing tales como Cisco, Microsoft y Amazon también proporcionan directrices para migrar las aplicaciones legadas a su plataforma [21], [63], [87], [115].

Tran, Keung, Lui y Fekete [109] presentaron una taxonomía de factores críticos donde sugieren que un ejercicio de migración a una plataforma en la nube no es fácil: algunos cambios que deben hacerse para lidiar con las diferencias en el entorno de software, tales como modelo de programación y APIs de almacenamiento de datos, así como diversas prestaciones.

Andrikopoulos, Binz, Leymann y Strauch [5] presentaron su trabajo centrándose en los desafíos y soluciones para cada una de las capas al migrar diferentes partes de la aplicación a la nube. Clasificaron los diferentes tipos de migración e identificaron las necesidades potenciales de impactos y adaptación para cada uno de estos tipos en las capas de la aplicación. Y también investigar diversos temas transversales que deben ser considerados para la migración de la aplicación y su posición con respecto a los tipos de migración identificados.

Carvalho, Neto, Garcia, Assad y Durao [16] presentan un estado del arte de cloud computing, identificando brechas y desafíos, sintetizaron evidencias disponibles tanto de uso y desarrollo, y proporcionaron información relevante, clarificando cuestiones sin resolver y cuestiones discutidas comunes acerca ese modelo a través de la literatura.

Pregunta de Investigación

Se ha llevado a cabo el mapeo sistemático, considerando las guías sugeridas en [50], [90]. La meta de este estudio es examinar el uso actual de las estrategias de migración de aplicaciones SOA a entornos cloud computing desde el punto de vista de la siguiente pregunta de investigación:

RQ: ¿Cómo los investigadores y profesionales migran sus aplicaciones SOA a entornos Cloud Computing, y cuál es el efecto sobre la calidad?

Esta pregunta de investigación permitirá clasificar y resumir el conocimiento actual sobre la migración de aplicaciones a la nube, con el fin de identificar las brechas en la investigación actual para proponer áreas en una posterior investigación y proporcionar conocimientos útiles para profesionales novatos en la migración de aplicaciones.

Identificación y Selección de Estudios Primarios

Las principales fuentes que se han utilizado para la búsqueda de estudios primarios son las librerías digitales IEEE Xplore, ACM, Science Direct y Springer Link.

Adicionalmente, se realizaron búsquedas manuales en conferencias en las que previamente habían sido publicados estudios relevantes para el dominio de la migración de Cloud Computing:

- Cloud Computing (International Conference on Cloud Computing, GRIDs, and Virtualization).
- IEEE Cloud (International Conference on Cloud Computing).

La cadena de búsqueda definida para obtener los estudios fue la siguiente: “(migra* OR evolv* OR adopt* OR reus* OR mov*) AND (soa OR service*) AND cloud”. El símbolo (“*”) permitirá cualquier variación de palabra en cualquier cada término de búsqueda que lo contenga (por ejemplo, la búsqueda del término “migra*” incluye las siguientes palabras: migrate, migrating, migration o…).

Se experimentó con varias cadenas de búsqueda y ésta fue la que mayor cantidad de artículos relevantes obtuvo.

La búsqueda se la realizó mediante la aplicación de la cadena de búsqueda a los metadatos (por ejemplo, título, resumen y palabras clave) de cada artículo para todas las fuentes (la sintaxis de búsqueda fue adaptada en orden para que pueda ser aplicada en cada biblioteca digital).

Estos términos de búsqueda también fueron tomados en cuenta en las otras fuentes que fueron inspeccionados manualmente con el fin de realizar una búsqueda constante.

El período revisado incluye estudios publicados desde 2006 hasta 2013 (ambos inclusive). Esta fecha fue seleccionada porque 2006 fue el año en que Amazon Inc. lanzó oficialmente Amazon Web Services [4] y después seguir las referencias de los estudios preliminares "Cloud Computing" ha comenzado a aparecer en el campo de la Ingeniería Web.

Criterios de Inclusión y Exclusión

Cada estudio que fue obtenido de la búsqueda automática o manual fue evaluado por los dos autores para decidir si debe o no ser incluido al considerar su título, resumen y palabras claves. Las discrepancias en la selección se resolvieron por consenso entre los dos autores después de escanear todo el documento.

Se incluyeron los estudios que cumplieron al menos uno de los siguientes criterios de inclusión:

- Estudios que presentan estrategias de migración de aplicaciones SOA a entornos Cloud Computing.
- Trabajos de investigación que presentan ejemplos o cualquier estudio empírico (por ejemplo, casos de estudio, experimentos), sobre estrategias de migración a entornos Cloud Computing.

Se excluyeron los estudios que cumplieron al menos uno de los siguientes criterios de exclusión:

- Artículos introductorios para ediciones especiales, libros y talleres.
- Reportes duplicados del mismo estudio en diversas fuentes.
- Artículos cortos con menos de cinco páginas.
- Estudios no escritos en inglés.

Aseguramiento de la Calidad

Además de los criterios de inclusión/exclusión, se consideró fundamental la evaluación de la calidad de los estudios primarios. Se diseñó un cuestionario de tres puntos de la escala de Likert para proporcionar una evaluación de la calidad de los estudios seleccionados. El cuestionario las preguntas son:

a) ¿El estudio presenta estrategias para migrar aplicaciones SOA a entornos Cloud Computing?
b) ¿El estudio ha sido publicado en una revista relevante o conferencia?
c) ¿El estudio ha sido citado por otros autores?

La puntuación de cada pregunta cerrada será la media aritmética de todas las puntuaciones individuales de cada revisor. La suma de la puntuación de las tres preguntas cerradas de cada estudio proporciona un resultado final que no fue utilizado para excluir documentos del estudio del mapeo sistemático, pero fue más bien utilizado para detectar estudios representativos.

Estrategia de Extracción de Datos

La estrategia de extracción de datos que se empleó fue basada en proporcionar el conjunto de posibles respuestas para cada sub-pregunta de investigación que había sido definida. Esta estrategia asegura la aplicación de la extracción del mismo criterio de datos para todos los trabajos seleccionados y facilita su clasificación.

Pregunta de Investigación	Criterio	Opciones
RQ1: ¿Qué estrategias se utilizan para migrar aplicaciones de basadas en Arquitectura Orientada a Servicios a entornos Cloud Computing?	C1: Estrategias de migración	Convencional Enfoque MDD
	C2: Formas de migración	Rehost Refactorización Revisión Reconstrucción
	C3: Tipos de migración	Reemplazo de componentes Migración parcial Migración de la pila del software Migración completa (Cloudify)
	C4: Modelos de despliegue	Privado Comunitario Publico Hibrido
	C5: Modelos de servicio	SaaS (Software como Servicio) PaaS (Plataforma como Servicio) IaaS (Infraestructura como Servicio)

RQ2: ¿Cuáles son las consecuencias de la migración en la calidad del producto?	C5: Aspectos de calidad	Eficiencia/Rendimiento Compatibilidad Confiabilidad Seguridad Mantenibilidad Portabilidad
RQ3: ¿Qué tipo de soporte se utiliza para migrar las aplicaciones SOA a entornos Cloud Computing?	C7: Tipo de soporte usado	Automatizado Semi-automatizado Manual
RQ4: ¿Cómo es abordada la investigación por la academia y la industria sobre la migración de aplicaciones SOA a entornos Cloud Computing?	C8: Fases en la que los estudios están basados	Requerimientos Diseño Implementación
	C9: Artefactos involucrados	Modelos/Transformaciones Código Fuente Otros
	C10: Tipo de validación	Encuestas Casos de estudio Experimentos controlados Otros
	C11: Ámbito de uso	Academia Industrial
	C12: Entorno de uso	Aplicación móvil Aplicación Web Ubiquidad Extensión

Tabla 2-1. Criterios de extracción.

Con respecto a RQ1 (Estrategias usadas para migrara aplicaciones SOA a Cloud), un artículo puede ser clasificado en una de las siguientes respuestas:

- C1: Estrategias de Migración: un estudio puede clasificarse en dos estrategias de migración. Se empleó este término para lograr clasificar las posibles estrategias utilizadas por investigadores y profesionales.
  a) *convencional*: si un estudio utiliza estrategias de migración existentes que están específicamente diseñado para la nube, y no sean MDD.
  b) *MDD*: si un estudio usa estrategias de migración basada en el enfoque del desarrollo de software dirigido por modelos que están diseñado específicamente para la nube.
- C2: Vías de Migración: un estudio puede clasificarse en cuatro vías de migración. Se empleó esta clasificación de acuerdo a [117] para ilustrar algunas de las diferentes alternativas sugeridas por los expertos.
  a) *Rehost:* si la migración es específicamente para mover una aplicación sin necesidad de hacer cambios en su arquitectura.
  b) *Refactorizar:* si la migración es específicamente para mover las aplicaciones a un entorno de hardware diferente y cambiar la configuración de la infraestructura de la aplicación sin cambiar su comportamiento externo.

c) *Revisar:* si la migración es específicamente para modificar o ampliar la base de código existente para apoyar modernización de sistemas heredados.
d) Reconstruir: si la migración es específicamente para volver a generar la solución, descartando el código de una aplicación existente y re-diseñar la aplicación.

– C3: Tipos de migración: un estudio puede clasificarse en cuatro tipos de migración. Se empleó esta clasificación según [5] para ilustrar algunas de las diferentes posibilidades de adaptación que la nube permite para una aplicación existente.
  a) *Reemplazar componentes*: si el tipo de migración donde uno o más componentes (arquitectura) son reemplazados por servicios en la nube.
  b) *Migrar parcialmente:* si tipo de migración es específicamente cambiar algunas de las funciones de aplicación en la nube, como capas de aplicación y componentes arquitectónicos.
  c) *Migrar la pila del software*: si el tipo de migración es específicamente para mover la aplicación que está encapsulada en máquinas virtuales y se ejecutan en la nube.
  d) *Cloudify*: si el tipo de migración es específicamente hacer una migración completa de las tareas de aplicación.
– C4: Modelos de despliegue: un estudio puede ser clasificado en cuatro modelos de despliegue. Se empleó esta clasificación según [59] ya que propone algunos de los modelos de despliegues que han sido definido por consenso entre los expertos.
  a) Privado: si el modelo de despliegue es provisionado para el uso exclusivo de una sola organización que comprende varios consumidores (por ejemplo, unidades de negocio).
  b) Comunitario: si el modelo de despliegue es provisionado para el uso exclusivo de una comunidad específica de consumidores de organizaciones que han compartido preocupaciones (por ejemplo, misiones, requisitos de seguridad, políticas y consideraciones de cumplimiento).
  c) Público: si el modelo de despliegue es provisionado para el uso abierto al público en general. Puede ser de propiedad, administrado y operado por una empresa, académico, u organización gubernamental, o alguna combinación de ellos.
  d) *Hibrido:* si el modelo de despliegue es provisionado como una composición de dos o más distintas infraestructuras cloud (privado, comunidad o público) que siguen siendo las únicas entidades, pero están unidos entre sí por la tecnología propietaria o estandarizada que permite la portabilidad de datos y aplicaciones (por ejemplo, nube de ruptura de equilibrio entre nubes de carga).
– C5: Modelos de Servicio: un estudio puede ser clasificado en tres modelos de servicio. Se empleó esta clasificación según [35] ya que propone algunos de los modelos de despliegues que han sido definido por consenso entre los expertos.
  a) Software como Servicio (SaaS): si el modelo de servicio tiene la capacidad de proveer al consumidor el uso de las aplicaciones del proveedor en una infraestructura en la nube.

b) *Plataforma como Servicio (PaaS):* si el modelo de servicio tiene la capacidad de proveer al consumidor el despliegue de aplicaciones creadas o adquiridas en la infraestructura cloud utilizando lenguajes de programación, bibliotecas, servicios y herramientas soportadas por el proveedor.
c) *Infraestructura como Servicio (IaaS):* Si el modelo de servicio tiene la capacidad de proveer al consumidor la prestación de procesamiento, almacenamiento, redes y otros recursos de computación fundamentales donde el consumidor es capaz de implementar y ejecutar el software arbitrario, que pueden incluir sistemas operativos y aplicaciones.

Con respecto a RQ2 (C6: Aspectos de calidad considerados en la migración), un documento puede ser clasificado en una o más características de calidad del estándar ISO/IEC 25010 SQuaRE [23]. Se empleó esta norma, ya que propone un modelo actualizado la calidad del producto que ha sido definido por consenso entre los expertos.

Con respecto a RQ3 (C7: Tipo de soporte utilizado en la migración), un documento puede ser clasificado en una de las siguientes respuestas:

a) *Automatizado:* si presenta una herramienta que realiza automáticamente la migración de toda o una porción grande de la migración.
b) *Semi-automatizado*: si presenta una migración parcialmente automatizada por una herramienta software.
c) *Manual*: si presenta un enfoque que se realiza en forma manual, lo que significa que la migración puede ser asistido por una computadora pero que las principales tareas de migración deben ser realizadas por un ser humano.

Finalmente, con respecto a RQ4 (Direccionamiento de la migración), un documento puede clasificarse en una de las siguientes respuestas:

- C8: Fase(s) en que se basan los estudios: un documento puede ser clasificado en uno o más procesos de alto nivel según la norma ISO/IEC 12207 [24]:
  a) *Requisitos:* si los artefactos que se utilizan como entrada para la migración incluyen especificaciones de alto nivel de la aplicación (por ejemplo, modelos de tarea, casos de usos, los escenarios de uso).
  b) *Diseño:* si la migración se realiza en los artefactos intermedios que se crean durante el proceso de desarrollo (por ejemplo, modelos de navegación, modelos de interfaz de usuario abstractos, modelos de diálogo).
  c) *Implementación:* si la migración se lleva a cabo en la interfaz de usuario final o una vez que la aplicación esté completada.
- C9: Artefactos involucrados: un documento puede ser clasificado en uno o más artefactos:
  a) *Modelos/Transformaciones:* si los artefactos que se utilizan como entrada/salida para la migración incluyen el enfoque del desarrollo dirigido por modelos (por ejemplo, casos de uso, diagramas de clase, transformaciones).
  b) *Código fuente:* si los artefactos que se utilizan como entrada/salida para la migración incluyen cualquier colección de instrucciones de computadora escritas utilizando un lenguaje informático legible.

c) *Otros:* si los artefactos que se utilizan como entrada/salida para la migración incluyen cualquier pieza no mencionada anteriormente (por ejemplo, componentes, tareas, imágenes de máquinas virtuales, bases de datos).

- C10: Tipo de validación: un documento puede ser clasificado en una de los siguientes tipos de estrategias que pueden llevarse a cabo según el propósito de la validación y las condiciones para la investigación empírica [17]:
  a) *Encuesta:* si proporciona una investigación realizada en retrospectiva.
  b) *Casos de estudio:* si proporciona un estudio observacional en el que los datos se recogen en ambientes simulados/real.
  c) *Experimento controlado:* si proporciona una investigación formal, rigurosa y controlada que se basa en verificar las hipótesis.
  d) *Otros:* si proporciona otras formas no mencionados anteriormente (por ejemplo, los ejemplos).
- C11: Ámbito de uso: un documento puede ser clasificado según el contexto en el que la estrategia de migración se ha definido o se está utilizando actualmente (contexto industrial y/o contexto académico).
- C12: Entorno de uso: un documento puede ser clasificado según el ambiente en el cual la estrategia de migración se ha definido o se está utilizando actualmente (aplicaciones móviles, aplicaciones Web, Ubicuidad, extensión).

Métodos de Síntesis

Se ha aplicado métodos de síntesis cuantitativos y cualitativos. La síntesis cuantitativa está basada en:

- Contar los estudios primarios que son clasificados en cada respuesta desde nuestro criterio.
- Definir gráficos de burbujas para mostrar las frecuencias de la combinación de resultados desde diferentes sub-preguntas de investigación. Un gráfico de burbujas es básicamente un eje x-y con burbujas en las intersecciones de las categorías. Este es útil para proveer un mapa y dar una rápida visión de un campo de investigación [91].
- Contar el número de estudios encontrados en cada fuente bibliográfica por año.

La síntesis cualitativa está basada en incluir algunos estudios representativos para cada criterio considerando los resultados de cada evaluación de la calidad.

2.2.2. Fase de Conducción

La búsqueda para identificar los estudios primarios en las bibliotecas digitales IEEE Xplore, ACM, Science Direct y Springer Link, fue realizada el 23 de septiembre de 2013. La aplicación del protocolo de revisión trajo los siguientes resultados:

- La búsqueda bibliográfica en las base de datos identificó 2673 publicaciones potencialmente relevantes (1026 de IEEE Xplore, 308 de ACM, 152 de Science Direct y 1187 de Springer Link). Luego de aplicar los criterios de inclusión y exclusión documentados anteriormente, fueron finalmente seleccionados (29 de IEEE Xplore, 7 de ACM, 5 de Science Direct y 2 de Springer Link).

- La revisión manual de la bibliografía de otras fuentes identificaron otras 15 publicaciones potencialmente relevantes. Luego de aplicar los criterios de inclusión y exclusión, se seleccionaron 5 estudios (1 de CLOUD COMPUTING y 4 de IEEE CLOUD).

Un total de 48 estudios de investigación fueron seleccionados por nuestros criterios de inclusión/exclusión. Algunos estudios han sido publicados en más de una revista/conferencia. En este caso, se ha seleccionado solamente la versión más completa del estudio. Los estudios que aparecen en más de una fuente fueron tomadas en cuenta solamente una vez.

Fuente	Estudios potenciales	Estudios seleccionados
Busqueda automatica		
IEEEXplore (IEEE)	1026	29
ACM DL (ACM)	308	7
Science Direct (SD)	152	5
Springer Link (SL)	1187	2
Total	2673	44
Busqueda manual		
CLOUD COMPUTING	8	1
IEEE CLOUD	7	4
Total	15	5
Resultados generales de ambas búsquedas	2688	48

Tabla 2-2. Resultados de la fase de conducción.

2.2.3. Fase de Resultados

Los resultados generales, que se basan en contar los estudios primarios se clasifican en cada una de las respuestas a nuestras sub-preguntas de investigación, se presentan en la Tabla 2-3. La lista completa de los estudios incluidos en este estudio sistemático se hace referencia en el Apéndice A.

Los resultados para el criterio C1 (Estrategias de migración) revelaron que aproximadamente el 96% de los documentos revisados presentan una estrategia convencional sobre la migración de aplicaciones a la nube. Por ejemplo, se encontró ejemplos representativos de esta estrategia en Babar et al. [S04], Chauhan et al. [S08], Nussbaumer et al. [S35] y Tran et al. [S42].

Babar et al. [S04] reportó sus experiencias y observaciones adquiridas al migrar un software de código abierto, *Hackystat*, a Cloud Computing. *Hackystat* es un entorno de código abierto utilizado para la recolección automatizada de datos de proceso y producto. Los objetivos de este trabajo es compartir sus experiencias y observaciones adquiridas a través de este proyecto y análisis crítico de la literatura con los que pretenden migrar sistemas de software en general y SOA basado en sistema en particular a Cloud Computing.

Chauhan et al. [S08] propuso contribuir al creciente conocimiento de cómo migrar los sistemas existentes a Cloud Computing generalizando su esfuerzo dirigido a migrar un *framework* Open Source Software (OSS), *Hackystat*, a Cloud Computing. Además, este trabajo reporto los principales pasos, el proceso y desafíos técnicos y algunas de las estrategias convencionales para migrar sistemas existentes a Cloud Computing.

Nussbaumer et al. [S35] propuso un nuevo marco que ayuda a las PYMES a la migración relacionada con multitudes cuestiones y desafíos que se presentan durante la fase de migración al Cloud. Además proporciona una metodología genérica para el análisis de los procesos de negocios y su migración al Cloud considerado importante, los requisitos específicos del Cloud para las PYMEs.

Sin embargo, la metodología de migración no consideró los factores económicos de la migración a la nube.

Tran et al. [S42] presentó una taxonomía de las tareas de migración involucradas, y mostraron el desglose de los costos entre categorías de tareas, para un caso de estudio que migra una aplicación de *n* niveles de .NET a Windows Azure. Determinaron los niveles de esfuerzos que son necesarios para diferentes tipos de nube, así como la identificación de las tareas faltantes.

El restante 4% de los estudios informó el uso de estrategia MDD. Estos resultados pueden indicar que hay pocos estudios que utilizan esta estrategia para migrar sistemas existentes a entornos Cloud Computing. Por ejemplo, se encontró en Guillen et al. [S18] y Mohagheghi et al. [S33].

Guillen et al. [S18] propuso un marco de trabajo llamado MULTICLAPP (*MULTICLoud* aplicaciones interoperables y migración). El marco sigue un proceso de tres etapas de desarrollo donde las aplicaciones pueden ser modeladas y codificadas sin que los desarrolladores estén familiarizado con la especificación de cualquier plataforma en la nube. Los enfoques MDE dependen de modelos como un medio de abstraer el proceso de desarrollo de las peculiaridades de cada plataforma en la nube.

Mohagheghi et al. [S33] presentó un proyecto de investigación llamado REMICS para definir la metodología y herramientas para la migración dirigida por modelos de aplicaciones legadas a una arquitectura orientada a servicios con el despliegue en la nube. El objetivo principal del proyecto fue desarrollar un conjunto de métodos y herramientas basadas en modelos que apoyen a las organizaciones con sistemas heredados para modernizarlos según el "paradigma de Servicio Cloud".

Los resultados para el criterio C2 (Vías de migración) revelaron que la migración más frecuentemente usada es *Rehost*, con alrededor del 44% de los documentos revisados. Estos resultados pueden indicar que la mayoría de las migraciones se realizan utilizando esta forma. Se identificó algunos ejemplos representativos, como Li et al. [S30] y Zhou et al. [S48].

Li et al. [S30] propuso un enfoque de colaboración flexible, llamado *CyberLiveApp*, para permitir el intercambio de una aplicación de escritorio virtual, basado en una infraestructura de virtualización y Cloud. Este enfoque es compatible con aplicaciones seguras de uso compartido y la migración en demanda entre múltiples usuarios o equipos. Para lograr los objetivos de uso directo compartido y migración entre máquinas virtuales, un enfoque de redirección de presentación basado en protocolo VNC y un servicio de clonación de máquinas virtuales basado en la interfaz *Libvirt* se utilizan. Estos enfoques han sido evaluados preliminarmente en un extendido *MetaVNC*.

Zhou et al. [S48] presentó un servicio de migración de aplicaciones tradicionales (servicio FTP) llamado *CloudFTP* a la nube. Implementaron el servicio FTP en la plataforma de Windows Azure junto con la función auto-escalamiento de la nube dado que *CloudFTP* sigue el modelo de desarrollo de aplicaciones en general sugerido Azure.

Refactorizar representa alrededor del 40% de los documentos revisados. Se identificaron los siguientes ejemplos representativos de esta manera en Beserra et al. [S05] y Chee et al. [S10].

Beserra et al. [S05] presentó *Cloudstep*, un proceso paso a paso de decisión para apoyar la migración de aplicaciones heredadas a la nube. El proceso se basó en la creación de perfiles basados en una plantilla de caracterización de la organización, la aplicación obsoleta de destino y los proveedores Cloud candidatos.

Chee et al. [S10] propusieron un enfoque basada en patrones *Cloud Transformation Advisor* (CTA) que ayuda a los usuarios a seleccionar los patrones de activación apropiados de una base de conocimientos de mejores prácticas al realizar planificación de transformación.

Esta base de conocimientos utiliza una representación estructurada para capturar información de la aplicación, información de capacidad de la plataforma Cloud e información del patrón de habilitación con el fin de facilitar la selección de patrón.

Reconstruir representa alrededor del 16% de los documentos revisados. Se identificaron los siguientes ejemplos para esta instancia en Cai et al. [S07] y la Song et al. [S39].

Cai et al. [S07] presentó los problemas con sistemas de información existentes y propuso el redesarrollo, que reescribe las aplicaciones existentes; una envoltura, que proporciona una interfaz nueva a un componente, por lo que es fácilmente accesible por otros componentes de software; y la migración.

Los resultados para el criterio C3 (Tipos de migración) revelaron que alrededor del 53% de los documentos revisados presentan *Cloudify* como tipo de migración de aplicaciones a la nube. Por ejemplo, se encontró ejemplos representativos de este tipo en Chauhan et al. [S09] y Lamberti et al. [S29].

Chauhan et al. [S09] presentó un marco de proceso para facilitar la migración a cloud Computing basado en sus experiencias de la migración de un sistema de código abierto (OSS), *Hackystat*, a dos plataformas de Cloud Computing diferentes (Amazon Web Services y Google App Engine).

Las principales actividades involucradas en el proceso de migración incluyen la identificación de las necesidades y potenciales de cada plataforma en la nube, el análisis de compatibilidad de las aplicaciones con los posibles entornos en la nube, la identificación de soluciones de arquitectura potenciales, evaluación de entornos Cloud para atributos de calidad específicos en el Cloud, análisis de la compensación de posibles soluciones de arquitectura, selección de modificaciones de arquitectura que deben ser incorporadas y la refactorización del sistema para incorporar nuevas modificaciones de la arquitectura.

Lamberti et al. [S29] presentó una implementación de prototipo, que consiste en un marco que permite a las aplicaciones de hoy ponerse en la nube aprovechando la automatización de la interfaz de usuario y accesibilidad, información incrustada en gráficas modernas basadas en la ventana de herramientas. Este marco está organizado como una arquitectura de tres niveles incluye el programa remoto particular que queremos pasar a la nube, una puerta del lado del servidor y un cliente web.

Migrar la pila del software cuenta alrededor del 32% de los documentos revisados. Se identificó el siguiente ejemplo representativo de este tipo, en Suen et al. [S40] propuso y evaluó las técnicas de almacenamiento basados en instancias y en volumen en infraestructura de nube pública y privada para almacenamiento y transferencia eficiente y eficaz de imágenes de máquinas virtuales. El objetivo fue el movimiento de máquinas virtuales entre la infraestructura en la nube pública y privada.

Por último, *migrar parcialmente* representa alrededor del 15% de los documentos revisados. Desde esta manera centrado en cambiar algunas de las funcionalidades de la aplicación. Se identificaron los siguientes ejemplos para esta instancia en Gerhards et al. [S15] y Juan-Verdejo et al. [S23].

Gerhards et al. [S15] direcciono la demanda de un marco consistente que permite una mezcla de cálculos dentro y fuera de las instalaciones para la migración de sólo determinadas partes al Cloud. Se utiliza el concepto de flujos de trabajo para presentar como las tareas de flujo de trabajo individuales pueden migrarse al Cloud mientras que las restantes tareas son ejecutadas dentro de las instalaciones.

KeywordsJuan-Verdejo et al. [S23] propuso un marco de migración al Cloud para asistir en la mudanza de la aplicación de destino siguiendo un modelo de despliegue local y en el cloud en lugar de un enfoque de todo o nada. Además este marco prevé que las partes de la aplicación son guardadas localmente mientras que otras partes se migran a infraestructuras en la nube.

Los resultados para el criterio C4 (Modelos de despliegue) revelaron que aproximadamente 9% de los documentos revisados selecciona el *modelo privado* como modelo de despliegue. Por otro lado, aproximadamente el 27% de los documentos revisados seleccionan el *modelo público* como modelo de despliegue. Por ejemplo, se encontró un ejemplo representativo de este modelo en Khajeh-Hosseini et al. [S27] proporciona dos herramientas que pretenden apoyar decisiones durante la migración de los sistemas al cloud público. Uno de ellos es una herramienta de modelado permite a los arquitectos IT modelar sus aplicaciones, datos y requisitos de infraestructura además de sus patrones de uso de recursos computacionales. El objetivo es apoyar la toma de decisiones durante la migración de sistemas informáticos a nubes públicas.

Finalmente, el *modelo híbrido* cuenta alrededor del 64% de los documentos revisados. Puesto que esta manera se centra en una composición de dos o más infraestructuras cloud distintas (privado, comunidad o público). Se identificaron los siguientes ejemplos para esta instancia Fan et al. [S13] y Hajjat et al. [S19].

Fan et al. [S13] propone un marco de migración de servicios automáticos e inteligente sobre una nube híbrida basada en la tecnología de agentes. El prototipo integró una nube privada con nube pública, una técnica de agente móvil es explotada para administrar todos los recursos, monitorea el comportamiento del sistema y negocia todas las acciones en la nube híbrida, con el fin de lograr la migración de servicios automáticos e inteligente entre las nubes.

Hajjat et al. [S19] aborda desafíos en migrar servicios empresariales en implementaciones basadas en una nube híbrida, donde las operaciones de la empresa son en parte alojadas en las instalaciones y otra parte en la nube. Las arquitecturas híbridas permiten a las empresas beneficiarse de arquitecturas basadas en la nube. Además desarrollaron un modelo para explorar los beneficios de un enfoque de migración híbrido. Este modelo toma en cuenta las limitaciones específicas de la empresa, el ahorro de costes y retrasos en el incremento de transacciones y costos de comunicación de área amplia que puedan resultar de la migración.

Los resultados para el criterio C5 (Modelos de Servicios) revelaron que alrededor del 53% de los documentos revisados presentan *infraestructura como servicio (IaaS)*. Por ejemplo, se encontró ejemplos representativos de esta estrategia en Khajeh-Hosseini et al. [S26], Lloyd et al. [S31] y Menzel et al. [S32].

Khajeh-Hosseini et al. [S26] presentó los beneficios potenciales y los riesgos asociados con la migración de un sistema a cloud (IaaS) a través de un caso de estudio. La infraestructura del sistema en el caso de estudio habría costado 37% menos durante 5 años en Amazon EC2, y usando de cloud computing podría tener potencialmente eliminado un 21% de ayuda que pide este sistema. Las desventajas incluyen los riesgos para la satisfacción del cliente y el servicio global de calidad debido a la difusión del control a terceros.

Lloyd et al. [S31] analizaron desafíos de la migración de aplicaciones multinivel a nubes de infraestructura como servicio (IaaS) en la realización de una investigación experimental mediante la implementación de un procesador consolidado y una variante del encuadernado de entrada/salida de RUSLE2. RUSLE2 es un modelo de erosión como un servicio web basado en la nube. Dos variantes del modelo RUSLE2 de erosión fueron probadas para investigar la migración de aplicaciones a la nube en IaaS.

Plataforma como servicio (PaaS) cuenta alrededor del 27% de los documentos revisados. Se identificó el siguiente ejemplo representativo de este modelo de servicio, Menzel et al. [S32] presenta un marco genérico denominado CloudGenius que proporciona una ayuda en el proceso y decisión de migración. Este marco permite migrar aplicaciones Web a la nube a lo largo de un proceso cíclico que sugiere imágenes de máquinas virtuales VM del cloud, y servicios de infraestructura cloud según los ingenieros de requisitos y los objetivos de los ingenieros Web.

Finalmente, *Software como Servicio (SaaS)* cuenta alrededor del 20% de los documentos. Puesto que esta manera se centra en proveer a los consumidores el uso de las aplicaciones del proveedor en una infraestructura cloud. Se identificó el siguiente ejemplo para esta instancia, Azeemi et al. [S03] propuso un modelo conceptual más holístico basado en modelos D&M para medir con éxito la migración de un sistema de información al cloud en escenarios pre y post migración. Este trabajo utiliza un caso de estudio de una organización del sector educativo que migró su sistema de información a un SaaS.

Los resultados para el criterio C6 (Aspectos relativos a la calidad) revelaron que los aspectos de calidad más frecuentemente son *rendimiento y eficiencia*, y *seguridad* que representan alrededor del 31% y 24% respectivamente. La razón es debido a la característica de elasticidad de las aplicaciones donde se requiere normalmente la implementación rápida y segura.

Otros aspectos de la calidad como la *mantenibilidad* y *compatibilidad* representan alrededor del 3% y 9% respectivamente. Esto es respecto con algunas afirmaciones indicadas por otros investigadores como "aspectos de calidad como la mantenibilidad juega un papel menor porque los proveedores de la nube son los responsables de esta parte en sus plataformas". Se ha encontrado algunos ejemplos de estos aspectos en Guillen et al. [S17].

Por otro lado, *confiabilidad y portabilidad* recibieron menos consideraciones, cuentan con 21% y 10% respectivamente. Se ha identificado un ejemplo representativo para estos casos en Babar et al. [S04] que indica que un ejemplo de tal requisito restringe las localizaciones geográficas de los lugares de almacenamiento de datos. Para algunas aplicaciones, es necesario que no se deberían almacenar los datos fuera de una región en particular. En nubes de IaaS, los datos se mantienen en un lugar diferente para la fiabilidad y la rentabilidad.

Los resultados para el criterio C7 (Tipo de soporte) revelaron que alrededor del 4% de los documentos revisados consideran *MDD semi-automatizado* como tipo de soporte. La justificación es porque sólo dos estudios seleccionaron usan desarrollo de software dirigido por modelos para implementar la migración a la nube, por ejemplo en Guillen et al. [S18] y Mohagheghi et al. [S33]. Por otro lado, aproximadamente el 25% de los documentos presentan un soporte *manual convencional*. Los resultados son porque no se usaron ninguna de las herramientas para llevar a cabo la migración a la nube.

Finalmente, los tipos de soporte más utilizados fueron *automatizados y semi automatizados* con estrategia *convencional* cuentan alrededor del 46% y 28% respectivamente. La razón es porque la mayoría de los estudios usan herramientas que permiten la migración a la nube con la intervención de los desarrolladores, o sólo la herramienta implementa la migración a la nube. Se identificaron algunos ejemplos representativos de estos casos en Juan-Verdejo et al. [S23], Kempf et al. [S25] y Khajeh-Hosseini et al. [S27].

Los resultados para el criterio C8 (fase en las que se basan los estudios) indicaron que las fases menos abordada fueron *requisitos y diseño* con alrededor del 4% y 6% respectivamente de los documentos revisados, tales como por ejemplo en Andrikopoulos et al. [S02] y Qiu et al. [S38].

Andrikopoulos et al. [S02] presenta los desafíos y soluciones para cada capa cuando se migran a diferentes partes de una aplicación a la nube.

Qiu et al. [S38] presenta un marco de optimización de diseño basado en la confiabilidad para migración de aplicaciones heredadas a la nube.

Finalmente, la mayoría de los estudios revisados se basan en la *aplicación*. Se ha identificado un ejemplo representativo en Chen et al. [S11] propuso un Trusted Cloud based on Security Level (TCSL), que está integrado, asegurado y confiado en arquitectura basada en la unión lógica de máquinas virtuales, para separar las máquinas virtuales con diferentes necesidades de sensibilidad y seguridad del entorno de la nube y cumplir con los requisitos de seguridad de diferentes clientes.

Los resultados para el criterio C9 (Artefactos involucrados) revelaron que lo más frecuentemente artefactos involucrados fueron *otros (por ejemplo, arquitectura, componentes, VMs, servicios)* representan alrededor de 76% de los estudios revisados. Se identificó un ejemplo representativo para este caso en Beserra et al. [S05]. Por otro lado, *modelos/transformación* cuentan alrededor del 4% de los documentos revisados. Se identificó un ejemplo representativo de esta instancia en Mohagheghi et al. [S33]. Por último, aproximadamente el 20% de los documentos revisados implican *código fuente*. Se encontró un ejemplo representativo en Chauhan et al. [S08] presentó en la fase de modernización de Hackystat, que implementa un algoritmo simple que lee la información sobre los servicios disponibles de un archivo de propiedades y distribuyen las solicitudes del cliente a los servicios de una manera simple.

Los resultados para el criterio C10 (Tipo de validación) revelaron que alrededor del 44% de los documentos revisados presentan casos de estudio con el fin de validar sus planteamientos. Este es un resultado alentador puesto que mejora la situación descrita en una revisión sistemática presentada en [61] que presentó la falta de estudios empíricos rigurosos de investigación en Ingeniería Web. Un ejemplo de caso de estudio puede encontrarse en Vu et al. [S45]. Sin embargo, *otros (por ejemplo, ejemplos)* representan alrededor del 16%. Se identificaron en el siguiente ejemplo en Venugopal et al. [S44]. Por otro lado, los *experimentos* representan alrededor del 38%. Los experimentos deben ser más empleados ya que proporcionan un alto nivel de control y son útiles para evaluar enfoques de una forma más rigurosa. Se encontró un ejemplo representativo de esta instancia en Lamberti et al. [S29]. Finalmente, las *encuestas* son el estudio menos preferido para los investigadores, cuenta con el 2%.

Los resultados para el criterio C11 (Ámbito de enfoque) revelaron que la mayoría de los estudios se han realizado desde el punto de vista de *investigación académica*, los mismos que representan alrededor del 74%. Se identificó un ejemplo representativo por ejemplo en Hao et al. [S20]. Sin embargo, también es importante tener en cuenta que un digno 26% de los estudios se realizaron desde el punto de vista de *investigación de la industria*. Se encontró el siguiente ejemplo para este caso en Pfitzmann et al. [S36] interesados en técnicas de direccionamiento para el mapeo de entornos de TI para plantillas cloud multi-imagen, que se han detectado como relevantes para los profesionales de la industria.

Los resultados para el criterio C12 (Entorno de uso) revelaron que un 69% de los documentos revisados se han realizado en *aplicaciones web*. Se identificó un ejemplo representativo de esta instancia en Chauhan et al. [S08]. Por otro lado, aproximadamente el 10% de los artículos revisados han sido implementados en *aplicaciones móviles*. Se encontró el siguiente ejemplo, en Amoretti et al. [S01] ilustra un enfoque basado en la *movilidad* de servicio, que permite a los sistemas para hacer frente a las condiciones ambientales altamente dinámicas. Por último, el 20% de los estudios se han cumplido en la *extensión*, un ejemplo de este tipo se pueden encontrar en Suen et al. [S40].

Criterio	Posibles respuestas	Resultados	
		# Estudios	Porcentaje (%)
C1: Estrategias de Migración	Convencional	46	95,83
	MDD	2	4,17
C2: Vías de Migración	Rehost	20	44,44
	Refactorizar	18	40,00
	Revisar	-	-
	Reconstruir	7	15,56
C3: Tipos de Migración	Reemplazar	-	-
	Migrar parcialmente	7	14,89
	Migrar la pila del software	15	31,91
	Cloudify	25	53,19

C4: Modelo de despliegue	Privado		1	9,09
	Comunidad		-	-
	Publico		3	27,27
	Hibrido		7	63,64
C5: Modelos de Servicio	SaaS (Software como Servicio)		10	20,41
	PaaS (Plataforma como Servicio)		13	26,53
	IaaS (Infraestructura como Servicio)		26	53,06
C6: Aspectos de Calidad	Rendimiento eficiencia		44	30,56
	Compatibilidad		14	9,72
	Fiabilidad		31	21,53
	Seguridad		35	24,31
	Mantenibilidad		5	3,47
	Portabilidad		15	10,42
C7: Tipo de soporte empleado	Automatizado	Convencional	13	27,66
		MDD	-	-
	Semi- Automatizado	Convencional	20	42,55
		MDD	2	4,26
	Manual	Convencional	12	25,53
		MDD	-	-
C8: Fase(s) en que los estudios están basados	Análisis		2	4,08
	Diseño		3	6,12
	Implementación		44	89,80
C9: Artefactos usados	Modelos / Transformaciones		2	4,35
	Código fuente		9	19,57
	Otros		35	76,09
C10: Tipo de validación	Encuesta		1	2,22
	Caso de estudio		20	44,44
	Experimento		17	37,78
	Otros		7	15,56
C11: Ámbito de enfoque	Industria		12	25,53
	Academia		35	74,47
C12: Entorno de uso	Aplicación móvil		5	10,42
	Aplicación Web		33	68,75
	Otros (Ubicuidad)		-	-
	Extensión		10	20,83

Tabla 2-3. Resultados del mapeo sistemático.

Es digno de mencionar que el análisis de la serie de estudios de investigación sobre la migración a la nube mostró que ha habido un aumento del interés sobre este tema desde 2009. La figura 2-2 muestra el número de publicaciones seleccionadas por año y la fuente. Se cree que este interés creciente apoya la importancia de realizar estudios basados en la evidencia en esta área.

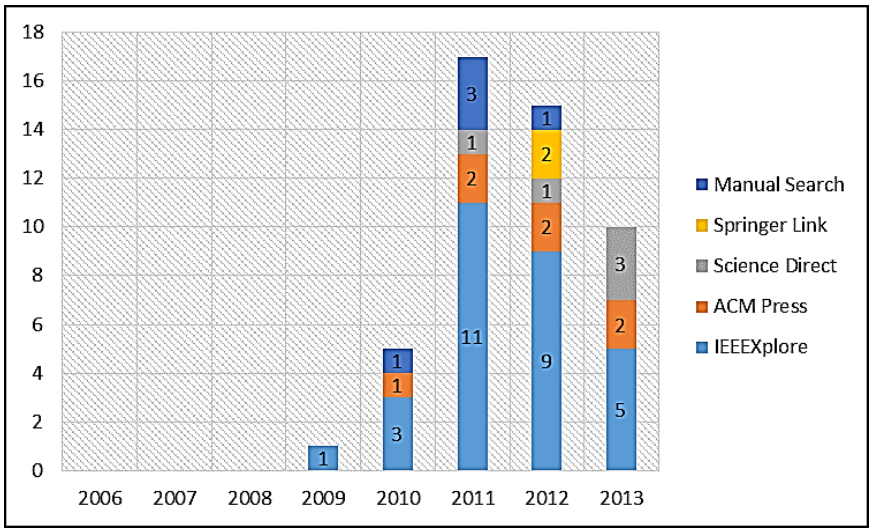

Figura 2-2. Número de publicaciones por año y fuente.

Se combinaron algunos criterios para establecer una correspondencia con el objetivo de proporcionar una visión general de la migración hacia la nube. Esta correlación nos permite obtener más información acerca de cómo los resultados de cada criterio se relacionan con los demás, y cuáles son las posibles brechas en la investigación. Por razones de espacio, la figura 2-3 muestra solamente una de las parcelas de la burbuja que se relaciona con el criterio de comparación C1 "estrategias de migración" en contra de las "vías de migración" C2, "artefactos usados" C9 y C10 "tipo de validación".

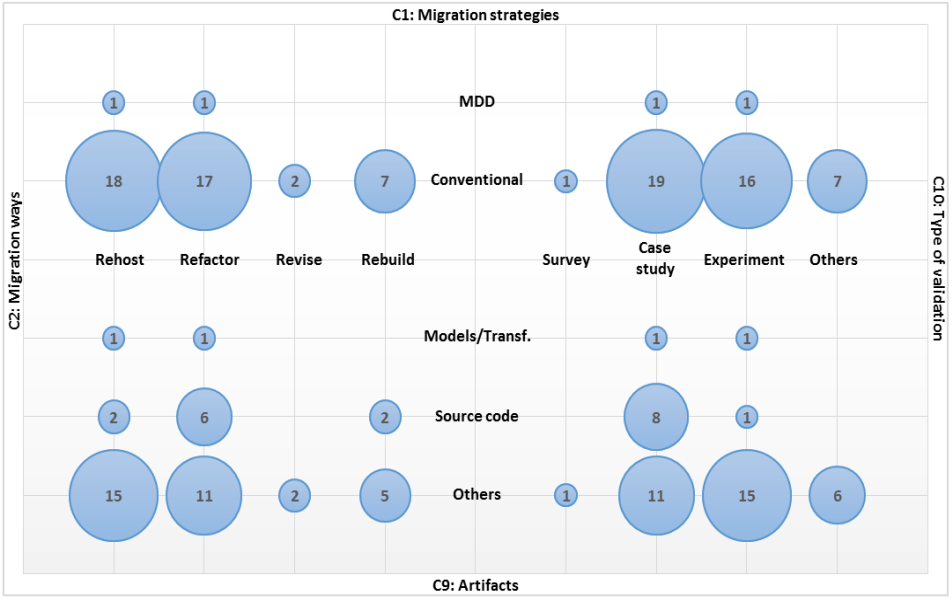

Figura 2-3. Resultados de los mapas obtenidos a partir de la combinación de C1 contra C2, C9 y C10.

2.3. Grupos de Investigación e Iniciativas Industriales

En esta sección se presentan brevemente los principales grupos de investigación académicos y empresariales vinculados al desarrollo de métodos y técnicas que permitan la migración de aplicaciones a entornos Cloud Computing.

2.2.1. Académicos

A nivel nacional son varios los grupos que trabajan, directa e indirectamente, en la materia especifica del trabajo de investigación o en materias afines. Entre ellos cabe citar los siguientes:

- DSA-Research [22] grupo de investigación de la Universidad Complutense de Madrid quienes trabajan en temas de IaaS (Infraestructura como Servicio), en particular, arquitectura Cloud, eficiencia energética de Cloud Computing, Cloud federadas e interoperabilidad, y computación de alto rendimiento (HPC).
- AST [112] grupo de investigación de la Universidad Rovira i Virgili, trabajan en temas de sistemas distribuidos, Cloud Computing, infraestructura basadas en la Web, y las comunicaciones móviles e inalámbricas.
- Laboratorio de Sistemas Distribuidos (LSD) [92] de la Universidad Politécnica de Madrid, abarcan diversos aspectos de la teoría y práctica de los sistemas distribuidos, computación en la nube, sistemas de bases de datos escalables, sistemas de procesamiento de eventos escalables, PaaS (Plataforma como Servicio), middleware escalable e infraestructura orientada a servicios.
- QUERCUS [30] grupo de investigación de la Universidad de Extremadura, quienes trabajan en diferentes líneas de investigación dentro del ámbito de la Ingeniería de Software: arquitecturas orientadas a servicios, cloud computing, visualización de datos e inteligencia de negocios, desarrollo de software dirigido por modelos, desarrollo de software orientado a aspectos, ingeniería web, inteligencia ambiental, modelado de aplicaciones empresariales, ontologías, web semántica y linked data, y ciudades inteligentes.

A nivel internacional, en el ámbito europeo existen gran cantidad de grupos de investigación que trabajan en distintos aspectos de la Ingeniería del Software y Cloud Computing, en particular se puede destacar:

- Distributed Systems Group [108] de la Universidad Técnica de Viena, quienes trabajan en modelado y caracterización de sistemas distribuidos de servicios híbridos, framework de integración end-to-end, modelado y programación de mecanismos incentivos, directivas de programación para computación elástica, simulación y análisis de las aplicaciones/sistemas híbridos, plataforma de entrega de servicios M2M elástica empresarial, procesamiento adaptativo de datos para sistemas generalizados de larga escala, sustentabilidad del gobierno basada en Cloud.
- Department of Computer Science [110] de la Universidad Umeå, quien aborda arquitectura para cloud y grid distribuidas y federadas, herramientas básicas para la gestión de cómputo y aplicaciones de uso intensivo de datos, control de elasticidad dinámico, control de admisión para el recurso overbooking seguro, colocación de las máquinas virtuales y datos, migración de las máquinas virtuales a gran escala.

También abroda la gestión de sistemas holístico basado en los objetivos de nivel de negocio, diseño de principios para aplicaciones grid y cloud , contabilidad en tiempo real aplicación de cuota de uso de la aplicación, programación distribuida y jerárquica de FairShare, intermediación de recursos, trabajo y la gestión del flujo de trabajo distribuido.

- Institute of Architecture of Application Systems (IAAS) [113] de la Universidad de Stuttgart, quien trabaja en computación orientada a los servicios y middleware, workflow y gestión de procesos de negocio, cloud computing, procesamiento de transacciones, tecnología de integración y patrones de arquitectura.
- P.A.R.Se.C. Research Group [111] de la Universidad de Nápoles, quien aborda el descubrimiento y gestión del conocimiento, web semántica y servicios web semánticos, recuperación de la información base semántica, cloud computing, computación de alto rendimiento y arquitecturas, agentes móviles e inteligentes y computación móvil, ingeniería inversa, análisis del programa automatizado y transformación, reconocimiento de patrones algorítmicos y programa comprensión, análisis de imagen.
- Institute e-Austria Timisora [94] de la Universidad del West Timisoara, quien trabaja en computación distribuida (cloud y grid), computación paralela (HPC y clúster).
- Software Development Group [44] de la Universidad de Copenhague IT, quien aborda IT para el cuidado de la salud, tecnología de bases de datos, optimización y transporte de contenedores, cloud computing, tecnología y herramientas de lenguajes de programación, seguridad en la red y tecnologías de la comunicación.
- Lero – The Irish Software Engineering Research Centre [55], quienes aborda temas en sistemas autónomos y adaptables, rendimiento del software, metodologías y estándares, seguridad y privacidad, alta integridad del software.
- Research in Computer Science [86] de la Universidad Özyeğin, quien aborda temas en cloud computing, modelos de minería de datos y bases de datos, flujo de datos y procesamiento de eventos complejos, servicios web, SOA, gestión de sesiones, archivo y sistemas de almacenamiento, almacenamiento en caché, redes de computadoras.
- School of Computing [25] de la Universidad de la Ciudad de Dublin, quienes aborda temas en modelos de negocio y mejores prácticas, seguridad y gestión, servicios de cloud computing, desarrollo y computación autonómica, aplicaciones cloud.

2.2.2. Iniciativas Industriales

A nivel nacional e internacional son varios los grupos que trabajan, directa e indirectamente, en la materia especifica del trabajo de investigación o en materias afines. Entre ellos cabe citar los siguientes:

- Cloud Computing Research Group [69] es un grupo de investigación de la multinacional Microsoft en el grupo de *Xtreme Computing* en MSR que investiga muchos aspectos de la infraestructura de software de computación en la nube.
- Fraunhofer Cloud Computing Alliance [31] es un grupo de seis institutos que llevan a cabo proyectos de investigación e industriales que se centran en diferentes aspectos de la computación en nube y temas relacionados, tales como grid computing, utility computing y la arquitectura orientada a servicios.
- The InIT Cloud Computing Lab [42] es un grupo de investigación dedicado a la nube informática en el área de Ingeniería de Servicios.

Capítulo 3. Marco Tecnológico

Este capítulo presenta una breve introducción de los temas esenciales para este trabajo de tesis, con el objetivo de tener una visión clara de cada uno de ellos.

En la Sección 3.1 se presenta a la Arquitectura Orientada a Servicios (SOA).

En la Sección 3.2 se presenta a Cloud Computing, una forma de proporcionar potencia de cálculo como un servicio en lugar de ser un producto.

En la Sección 3.3 se presenta el paradigma del Desarrollo de Software Dirigido por Modelos y éste en relación con la computación en la nube.

3.1. Arquitectura Orientada a Servicios (SOA)

La computación orientada a servicios (*Service Oriented Computing*) es una nueva generación de plataformas para la computación distribuida. Esta incluye muchos conceptos, como su propio paradigma de diseño (orientación a servicios) que ofrece normas y directrices para realizar ciertas características de diseño en una aplicación. La computación orientada a servicios tiene sus propios principios y patrones de diseño, un propio modelo de arquitectura que es conocida como Arquitectura Orientada a Servicios, junto a muchos conceptos, tecnologías y marcos de trabajo asociados. El término computación orientada a servicios y arquitectura orientada a servicios no son sinónimos, pero la literatura muy a menudo no siempre los trata como a uno [29].

La computación orientada a servicios es altamente afectada por diferentes paradigmas y tecnologías tales como: orientada a objetos, computación distribuida, gestión de procesos de negocios, servicios Web, e integración de aplicaciones empresariales (EAI). La computación orientada a servicios es guiada por numerosos estándares de la industria. Tanto por organizaciones de normalización (incluyendo OMG, W3C, OASIS y WS-I) y organizaciones de la industria (tales como Microsoft, IBM, Sun Microsystems, Oracle, Hewlett-Packard, Fujitsu-Siemens y SAP) han desarrollado activamente la computación orientada a servicios, y están en constante desarrollo de nuevas normas y especificaciones [29], [45].

3.1.1. Definición

La arquitectura orientada a servicios (*Service Oriented Architecture*) es un modelo arquitectónico que tiene como objetivo aumentar la eficiencia, agilidad y productividad de una empresa, haciendo hincapié en los **servicios** como el principal medio, a través del cual, la lógica del negocio es representada en soporte de la realización de los objetivos estratégicos, asociados con la computación orientada a servicios (SOC). Como una forma de arquitectura de tecnología, una implementación de SOA puede consistir en una combinación de tecnologías, productos, APIs, soportando extensiones de infraestructura, y algunas otras partes [29]. SOA también puede ser visto como un estilo de arquitectura para la construcción de aplicaciones de software que utilizan los servicios de una red [37].

Existen numerosas definiciones para SOA: Linthicum define a SOA como: "Un marco estratégico de tecnología que permite a todos los sistemas interesados, dentro y fuera de una organización, exponer y tener acceso a servicios bien definidos, e información ligada a aquellos servicios, además que pueden ser abstraídos y procesados por capas y aplicaciones compuestas para el desarrollo de soluciones [56]. En esencia, SOA añade el aspecto de la agilidad con la arquitectura, lo que nos permite tratar con cambios en el sistema utilizando una capa de configuración en vez de constantemente tener que volver a desarrollar estos sistemas" [56]. OASIS define a SOA como: "Un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios de propiedad. Proporciona un medio uniforme para ofrecer, descubrir, interactuar y utilizar las capacidades para producir los efectos deseados en consonancia con las condiciones previas y expectativas medibles" [76]. Y por último Josuttis la define como: "Un paradigma arquitectónico para hacer frente a los procesos de negocio distribuidos en un amplio panorama de los sistemas heterogéneos existentes y las nuevos que se encuentran bajo el control de diferentes propietarios" [45].

SOA es principalmente para la construcción de aplicaciones de negocios [40]. Los procesos de software en SOA están estrechamente ligados al modelado de procesos de negocio, y los procesos de negocio manejan el diseño de SOA. La lógica de la aplicación de una solución SOA está débilmente acoplada y está residiendo en varios equipos que pueden estar dentro o fuera de la organización. La sustitución y el rediseño de los componentes y la reutilización de la lógica de aplicación existente en una solución SOA debe ser más bien una tarea fácil. En SOA, los componentes se comunican a través de una única interfaz y los sistemas normalmente sirven para múltiples usuarios remotos [8]. De acuerdo con Josuttis, con el fin de establecer SOA con éxito, se necesitan tres componentes: la infraestructura, la arquitectura y los procesos. Una plataforma técnica, tales como los servicios Web son necesarios para formar la infraestructura [45]. Con base en los conceptos de SOA, estándares y herramientas, algunas decisiones tienen que hacerse para formar la arquitectura del sistema SOA. Diferentes procesos incluyen el modelado de procesos de negocio, los ciclos de vida de servicios y gobernabilidad SOA, es decir, una de las tareas más importantes en una SOA con éxito [45].

3.1.2. Elementos

La ideología fundamental detrás de SOA, es que existe un gran problema que necesita ser resuelto con el software al dividirlo en trozos más pequeños, en unidades individuales de la lógica conocida como **servicios** que están diseñados con los principios de la Computación Orientada al Servicio.

Un **servicio** es una entidad independiente que encapsula una lógica relacionada con una determinada tarea de negocio u otra agrupación lógica, e intercambia información con otros servicios para lograr la meta deseada. Un servicio puede combinarse con otros servicios con el fin de crear composiciones de servicios. Los servicios pueden ser distribuidos, lo que significa que ellos pueden residir dentro o fuera de la empresa.

Los servicios existen autónomamente pero no están aislados, mantienen un grado de uniformidad y estandarización [28], [29]. Cuando se utilizan los servicios, los servicios no se compran como un paquete de software [37].

La comunicación entre servicios requiere que los servicios estén conscientes de los otros. Esto sucede con la descripción de los servicios que contiene el nombre y la ubicación del servicio y, entradas y salidas de datos requisitos del servicio de intercambio. Los servicios son combinados libremente a través de descripciones del servicio. Se necesita un marco de comunicación para establecer la comunicación entre servicios [29]. Este marco define la estructura de los mensajes, las políticas y protocolos que son usados en la comunicación. Como los servicios los mensajes también son autónomos. Ellos también contienen suficiente inteligencia para gobernar sus partes de la lógica de procesamiento. Después de enviar un mensaje, el remitente no es capaz de determinar, lo que va a suceder con el mensaje [28], [29]

Las **interfaces de aplicación** son los actores activos en la arquitectura, su función es iniciar y controlar todas las actividades de los sistemas de la empresa. Las interfaces de aplicación más utilizadas son:

- **Interfaces gráficas de usuario:** Éste tipo de interfaz permite a los usuarios finales interactuar directamente con la aplicación, las interfaces gráficas pueden ser aplicaciones web o clientes ricos.
- **Programas de lotes o procesos:** Los programas o procesos de larga vida invocan su funcionalidad de manera periódica o son el resultado de acontecimientos concretos.

Sin embargo, es posible que una interfaz de aplicación delegue gran parte de la responsabilidad a servicios o procesos de negocio.

Los **servicios de contrato** proporcionan una especificación informal de la finalidad, funcionalidad, restricciones y el uso del servicio [54]. La forma de esta especificación puede variar, dependiendo del tipo de servicio. Un elemento no obligatorio de los servicios de contrato es una definición de interfaz formal basada en lenguajes como son el lenguaje de definición de Interface (IDL) o el lenguaje de descripción del servicio Web (WSDL) [54]. Estos elementos proporcionan abstracción e independencia de tecnología, incluyendo el lenguaje de programación, el protocolo de middleware de la red y su entorno de ejecución.

El contrato puede imponer la semántica detallada en las funciones y parámetros que no están sujetos a las especificaciones IDL o WSDL [54]. Es importante comprender que cada servicio requiere un servicio de contrato en particular si no hay una descripción formal basada en una norma como WSDL o IDL.

La funcionalidad de los servicios es expuesta a los clientes por el **servicio de interfaz**, los clientes deben estar conectados al servicio utilizando una red [54].

Aunque la descripción de la interfaz es parte del servicio de contrato, la implementación física de la interfaz consta de esbozos del servicio, que están incorporados en los clientes de un servicio y un despachador [54].

Los **servicios de implementación** proporcionan físicamente la lógica de negocios requerida y los datos que son apropiados. Esto es la realización técnica que cumple con servicio de contrato. El servicio de implementación consiste de uno o más artefactos como son programas, datos de configuración y bases de datos [54].

Los **servicios de lógica de negocio** son los encargados de encapsular la lógica de negocio como parte de su implementación. Ésta se encuentra disponible a través de interfaces de servicios. Sin embargo, la programación en contra de las interfaces es deseable, si se aplica un planteamiento orientado al servicio [54].

Un servicio también puede incluir datos. En particular, este es el propósito de los **servicios de datos** céntricos [54].

3.1.3. Principios de diseño

Varios principios de diseño pueden ser seguidos en el diseño de una solución orientada a servicios. Al diseñar un servicio orientado a solución y construyendo SOA, algunas preguntas importantes son: ¿Cómo los servicios deben ser diseñados?, ¿Cómo debe ser definida la relación entre servicios?, ¿Cómo deben diseñarse las descripciones de los servicio? y ¿Cómo deben diseñarse los mensajes?. Ocho principios de diseño de SOA se han definido [29]. No todos los principios de diseño pueden realizarse simultáneamente, porque los principios se interrelacionan. Esto significa que para lograr un principio se puede requerir la realización de otro principio. Por ejemplo, para alcanzar el acoplamiento flexible, el servicio de contrato debe estar presente. Cinco principios establecen las características de diseño servicio concreto y los tres restantes actúan más como influencias reguladoras [28], [29].

3.1.4. Abstracción de la capa de servicios

Con el fin de lograr la reutilización, no es posible crear servicios ágiles que se adaptan tanto para negocios como para aplicaciones de consideraciones lógicas simultáneamente. Por lo tanto, las capas especializadas de servicios tienen que ser construidas. Estas capas son: la *capa de servicios de aplicación*, la *capa de servicios de negocio* y la *capa de orquestación de servicios*. La figura 3-1 muestra estas capas entre los procesos de negocio y la capa de aplicación [28].

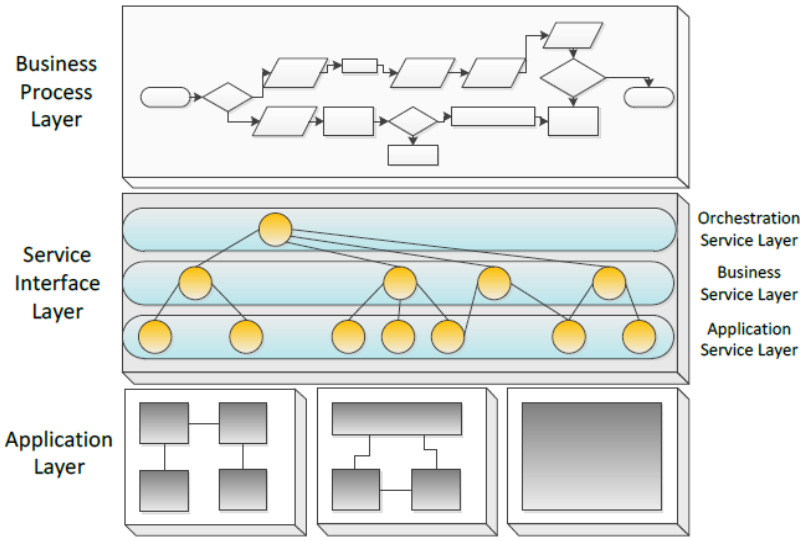

Figura 3-1. Capas de servicio en SOA [20].

La ***capa de servicios de aplicación*** proporciona funciones reutilizables que están relacionadas con el procesamiento de datos en un entorno de aplicaciones. Los servicios de aplicación son altamente específicos de tecnología. Los tipos de servicios de aplicaciones más comunes son *servicios de utilidad* y *servicios de envoltura* [29]. Los servicios de utilidad son servicios genéricos que proporcionan las operaciones reutilizables para servicios de negocio permitiendo completar las tareas centrada en el negocio. Los servicios de envoltura se utilizan típicamente para los propósitos de integración. Ellos encapsulan la lógica de los sistemas heredados. Los servicios que contienen tanto aplicación como lógica del negocio se denominan *servicios híbrido*s [28].

El propósito de la ***capa de procesos de negocio*** es introducir servicios que se concentran en la representación de la lógica de negocio [28]. Los servicios de negocios siempre son implementaciones del modelo del servicio de negocio. Sin embargo, los servicios de negocio también pueden clasificarse como un *servicio de controlador* o un *servicio de utilidad*. Es muy probable que los servicios empresariales actúen como reguladores que se componen de varios servicios de aplicaciones para ejecutar su lógica de negocio.

Los modelos de servicios de negocio incluyen los modelos de negocio *centrados en las tareas* y los modelos de negocio *centrados en la entidad* [28]. Los servicios de negocio centrados en las tareas encapsulan una lógica de negocio que se específica de un determinado proceso de trabajo o negocio. Los servicios de negocios centrados en la entidad encapsulan una entidad de negocios específicos (por ejemplo, una hoja o una factura) teniendo así un mayor potencial de reutilización [28].

La ***capa de orquestación*** consiste en uno o más servicios de proceso que componen los servicios en los niveles inferiores (servicios de negocio y aplicaciones) según las reglas de negocio y la lógica empresarial que está alojada dentro de las definiciones del proceso [28].

La capa de orquestación evita la necesidad de otros servicios, administra los detalles relacionados con la interacción que se requieren para garantizar la correcta ejecución de las operaciones de servicio. Los servicios de procesos residen en la capa de orquestación y otros servicios que proporcionan conjuntos específicos de funciones, independientes de las reglas de negocio y la lógica específica del escenario a componer [28]. Los servicios de procesos pueden clasificarse como *servicios de control* porque componen otros servicios para ejecutar la lógica del proceso de negocio. Los servicios de procesos también pueden convertirse en servicios públicos si el proceso que se ejecuta puede considerarse reutilizable. Las reglas de negocio y servicios de ejecución de la lógica de la secuencia se abstraen de otros servicios de orquestación [28].

3.1.5. Servicios Web

La evolución de los servicios web puede ser vista como una fuerza conductora en la evolución del servicio de orientación, y ha dado forma considerablemente a la computación orientada al servicio. Aunque SOA como una arquitectura de tecnología y la orientación de servicio como un paradigma ambos están en una implementación neutral, los servicios Web son muy a menudo asociados con ellos [28], [45]. Con el fin de construir una solución SOA, es necesaria una plataforma de aplicación. La tecnología de los servicios Web ofrece este tipo de plataforma [28]. Los servicios Web se utilizan para implementar tareas empresariales compartidas entre las empresas, y son la tecnología que permite conectar diferentes sistemas desacoplados a través de varias plataformas, lenguajes de programación y aplicaciones. También pueden ser utilizados para la integración de aplicaciones empresariales [89]. La definición de un servicio Web es: *"Un servicio Web es una plataforma independiente, acoplamiento flexible, autónomo, programable, habilitado para aplicaciones Web, que puede ser descrito, publicado, descubierto, coordinado y configurable utilizando artefactos XML (estándares abiertos) con el fin de desarrollar aplicaciones interoperables distribuidas"* [89].

3.1.5.1. Roles de los Servicios Web

Los servicios Web difieren entre sí en función de las tareas que realizan y el alcance de la tarea. Un servicio Web puede ser una tarea de negocio (por ejemplo, los fondos de retiro de un servicio de depósito), un completo proceso de negocio (una compra automatizada de suministros de oficina), una aplicación (una previsión de demanda o solicitud de reposición) o puede ser un recurso que es activado por un servicio Web (por ejemplo, el acceso a una base de datos que contiene registros médicos) [89]. Las funciones de los servicios Web pueden variar desde simples peticiones para completar aplicaciones de negocio que accedan y combinan información de muchas fuentes diferentes.

Los servicios Web a veces se mezclan con las aplicaciones Web. Cuatro diferencias clave entre los servicios Web y aplicaciones Web pueden realizarse [89]:

1. Los servicios Web pueden actuar como recurso para otras aplicaciones, de manera que los servicios Web puedan utilizar otros servicios sin la intervención humana [89].
2. Un servicio Web conoce sus propiedades funcionales y no funcionales, y puede decirles a usuarios y otros servicios web. Las propiedades funcionales incluyen funciones que el servicio Web puede realizar, que entradas son requeridas para producir sus salidas [89]. Las propiedades no funcionales incluyen costos de utilización del servicio, medidas de seguridad, características de rendimiento (representación) e información de contacto [89].
3. El estado del servicio Web puede monitorizarse mediante el uso de sistemas de control y gestión de aplicaciones externas. De esta forma los servicios Web son más visibles y manejables que las aplicaciones basadas en la Web [89].
4. Los servicios Web también pueden ser negociados o subastados. Un corredor puede elegir un servicio Web conveniente para sus fines entre varios servicios realizando la misma tarea. Una opción puede basarse, por ejemplo, en el costo, rapidez y grado de seguridad del servicio Web [89].

3.1.5.2. Interfaz e Implementación

La interfaz y la implementación de los servicios Web están separados. La interfaz del servicio es visible para los usuarios del servicio. Define la funcionalidad del servicio y cómo acceder a ella [29]. La interfaz también es denominada como un contrato de servicios. Consiste en la definición del WSDL (lenguaje de descripción de servicios Web) y la definición del esquema XML (eXtended Markup Language) y puede incluir definición de políticas de WS (servicio Web) [29]. Los detalles de implementación del servicio están ocultos a los usuarios del servicio. La implementación del servicio se divide en lógica de programación lógica y el procesamiento del mensaje. La lógica de programación puede ser lógica heredada que es envuelta por un servicio web o puede ser lógica que especialmente desarrollada para el servicio Web [29]. En este caso, la lógica es a menudo llamada lógica de servicio básico o lógica de negocios. Los depuradores, procesadores y agentes de servicio componen la lógica de procesamiento de mensajes de un servicio Web. Ellos pueden manejar los mensajes enviados o recibidos por los servicios Web. Algunas de estas lógicas son proporcionadas por el entorno de ejecución pero también puede ser hecha a la medida. La figura 3-2 representa los componentes de un servicio Web [29].

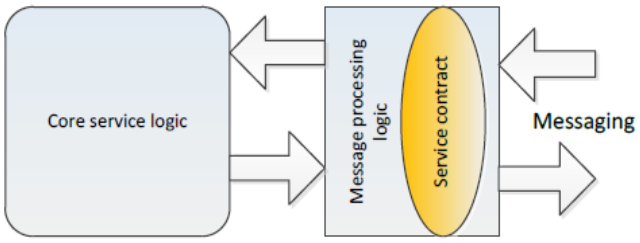

Figura 3-2. Componentes de un servicio Web [19].

La misma interfaz del servicio puede implementarse por diferentes prestadores de servicios mediante el uso de cualquier lenguaje de programación [89]. Las implementaciones pueden ser diferentes: una implementación puede proporcionar cierta funcionalidad mientras que otra implementación puede usar una combinación de otros servicios para proporcionar la misma funcionalidad [89]. La separación entre la interfaz del servicio e implementación puede ser tan radical que no son organizaciones, las cuales proporcionan interfaces de servicio no son necesarias en las mismas organizaciones que implementan los servicios [89]. Diferentes proveedores de SOA han desarrollado sus plataformas para utilizar la tecnología de los servicios Web. Actuales plataformas y tecnologías industriales incluyen potentes soporte de servicios Web y XML. Estas tecnologías incluyen IBM Websphere Toolkit, Sun's Open Net Environment y JINITM technology, Microsoft .NET y Novell's One Inititatives, HP e-speak y BEA's WebLogic Integration [39].

3.1.5.3. Estándares de los Servicios Web

El último objetivo de la tecnología de los servicios Web es permitir la cooperación entre aplicaciones sobre protocolos estandarizados sin necesidad de intervención humana [37]. La pila de tecnología de servicios Web consiste en numerosos estándares definidos por la comunidad de proveedores, y están evolucionando constantemente. La normalización resuelve muchos problemas de interoperabilidad que han sido un problema en anteriores tecnologías distribuidas. Además, la estandarización de SOA, permite a los proveedores crear herramientas que automatizan ciertos pasos en el proceso de desarrollo de software [37]. La capa de red y mensajería XML proporcionan las bases para la interacción entre servicios Web. Los datos que pasan por los servicios mutuamente están en XML con tipos de datos especificados por el esquema XML [39]. En la capa de transporte, los servicios web utilizan comúnmente las ventajas de HTTP (Hypertext Transfer Protocol) [89]. Los estándares de los servicios Web pueden ser clasificados en ocho principales grupos:

1. Estándar de tecnología.
2. Estándar de servicios básicos.
3. Estándar de gestión de contextos y transacción.
4. Estándar de servicio de composición y colaboración.
5. Fiabilidad, enrutamiento, y archivos adjuntos.
6. Políticas y metadatos.
7. Perfiles WS.
8. Seguridad.

Los estándares de servicios básicos incluyen Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP) y Universal Description Discovery and Integration (UDDI). En un escenario típico de SOA, las descripciones WSDL son publicadas en un registro de servicio por los proveedores de servicios [28], [88].

Los clientes o consumidores del servicio acceden a este registro dinámicamente y combinan servicios que cumplan con las necesidades determinadas. SOAP establece un canal de comunicación entre servicios Web [28], [88].

Los servicios necesitan información sobre el otro para comunicarse. WSDL ofrece una forma estándar para representar interfaces de servicio (contratos de servicio). WSDL define la gramática XML para describir los servicios Web como comunicación de extremos que pueden intercambiar mensajes con otros servicios [89]. La descripción del servicio incluye la ubicación de un servicio, cómo se utiliza el servicio y las operaciones y métodos que ofrece el servicio. WSDL proporciona los metadatos para los consumidores de servicios que son capaces de enviar mensajes válidos para prestadores de servicios [28]. WSDL define también cómo el servicio reacciona al mensaje recibido. Ambas definiciones tanto abstractas y concretas constituyen un documento WSDL. La definición abstracta describe las propiedades de la interfaz sin una referencia a la tecnología subyacente. Para poder ejecutar su lógica, la definición abstracta debe conectarse a una definición concreta que incluye el protocolo de transporte que se utiliza así como la dirección física del servicio. WSDL es extensible y permite la descripción de los extremos del servicio y sus mensajes sin importar los formatos del mensaje o protocolos de red usados [28], [102], [116].

Los servicios Web se comunican usando el protocolo SOAP que implementa un modelo de solicitud de respuesta para la comunicación. SOAP es un protocolo ligero basado en XML que es independiente de cualquier modelo de programación y otros temas específicos de la aplicación [102], [116]. SOAP permite que las aplicaciones se ejecuten en diferentes sistemas operativos e implementados con diferentes plataformas de tecnología y lenguajes de programación, para comunicarse con los demás [102], [116]. SOAP tiene dos propiedades fundamentales: puede enviar y recibir HTTP (u otro) transportar paquetes de protocolo y procesar los mensajes XML. Una meta importante de SOAP es ser extensible con características incluyendo los patrones de intercambio de confiabilidad, seguridad, enrutamiento y mensaje. Las WS-Extensions pueden utilizarse para añadir más características en los mensajes SOAP [89]. REST (Representation State Transfer) es una tecnología que puede utilizarse para construir servicios web en lugar de SOAP [45].

Servicio de descubrimiento y registro son importantes funciones en SOA [89]. Los servicios Web son útiles sólo si un usuario potencial es capaz de encontrar los servicios que necesita. A medida que el número de servicios Web crece, es necesario tener un registro de servicio para realizar un seguimiento de que servicios tiene para ofrecer y cuáles son las características de esos servicios. UDDI se creó para abordar estas cuestiones. Se basa en estándares de la industria incluyendo HTTP, XML, XML Schema y SOAP. Es una base de datos pública de descripciones de servicio y permite a los servicios Web ser encontrados por los servicios clientes [89].

UDDI contiene servicios que apoyan la descripción y el descubrimiento de las empresas, organizaciones y otros proveedores de servicios Web. UDDI ofrece interfaces técnicas para acceder a servicios Web. Puede utilizarse para servicios disponibles al público y que se utilizan internamente dentro de una organización [77].

3.1.6. SOA Contemporánea

Thomas Erl separa los conceptos de primitiva (o primera generación) y contemporánea (o de segunda generación) de SOA [29]. SOA primitiva se basan en normas de servicio pero carece de la presencia, por ejemplo, las cuestiones de seguridad. Esto se convierte en un problema serio en el desarrollo de funciones de nivel empresarial misión crítica. Cuestiones como la seguridad a nivel de mensaje, cruce de transacciones, y mensajería confiable debe abordarse. En la lista a continuación, se enumeran algunos requisitos para las capacidades que deben mejorarse en SOA primitiva [29]:

- La capacidad para realizar las tareas en forma segura, protegiendo el contenido de un mensaje, así como el acceso a los servicios individuales.
- Permitiendo que las tareas deban realizarse confiablemente para que la entrega de mensajes o notificación de entrega puede ser garantizada.
- Realización de requisitos para asegurarse de que la parte superior de un mensaje SOAP y contenido XML procesado no inhibe la ejecución de la tarea.
- Capacidades transaccionales para proteger la integridad de las tareas específicas del negocio con una garantía de que si falla la tarea, se ejecuta la lógica de excepción.

Según Erl, SOA contemporánea es una plataforma arquitectónica compleja y sofisticada que tiene un potencial significativo para resolver muchos de los problemas históricos y actuales. La definición de SOA contemporáneo es: *"SOA contemporáneo representa una arquitectura abierta, extensible, federada, componible que promueve la orientación al servicio y está compuesto de diversos servicios, interoperable, detectable y potencialmente reutilizables, autónomo, con capacidad de QoS, diversos proveedores, implementados como servicios Web"* [29].

SOA contemporánea se basa en una SOA primitiva extendiéndola con otras tecnologías de servicios Web (etiquetada como WS-* *Extensions*). El propósito de estas extensiones es mejorar las brechas de calidad de la SOA primitiva. Cada WS-* *Extension* cuenta ciertos principios de diseño [29]. Es notable, que como principios de diseño, también se superponen las extensiones y algunas de ellas están muy relacionadas entre sí. Las extensiones están en constante evolución, que presenta más desafíos en el diseño de SOA. Además, clientes y socios de negocios pueden tener diferentes niveles de madurez en las especificaciones. Esto crea nuevos retos en el desarrollo de SOA [28]. En la figura 3, se destaca la primera generación SOA con color amarillo. Algunos de WS-* *Extensions* disponibles, que son necesarios para construir SOA contemporánea destacan como azul.

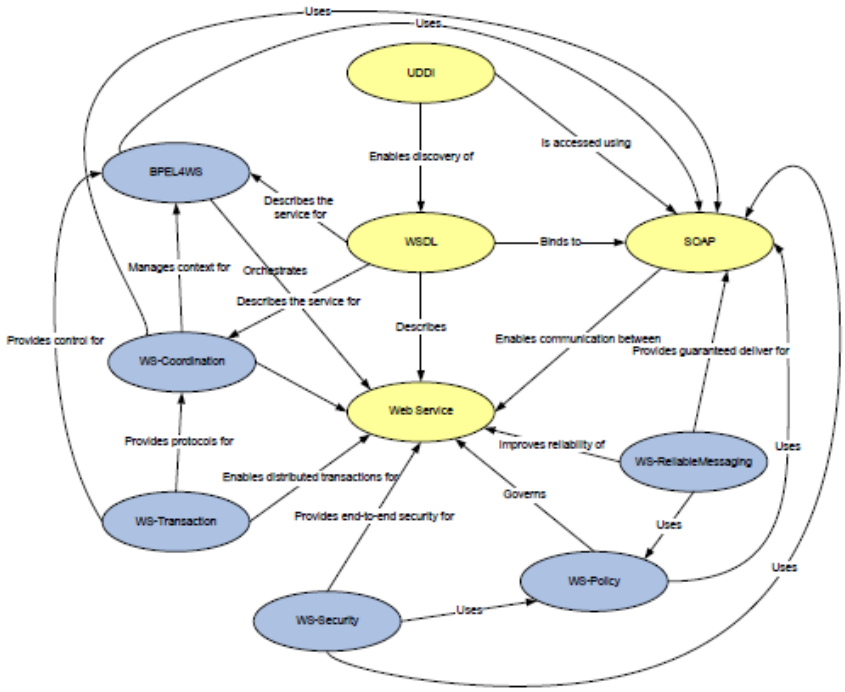

Figura 3-3. Relaciones Web entre estándares de servicios Web [54].

3.1.7. Proceso de entrega SOA

Hay dos tareas principales, y por separado en el proceso de entrega SOA [37]. El primero es para crear la infraestructura de servicios con servicios básicos, gestión y seguridad. El otro es para construir aplicaciones empresariales reales que se basan en la infraestructura. Ambas estas tareas tienen sus características de desarrollo propio [37].

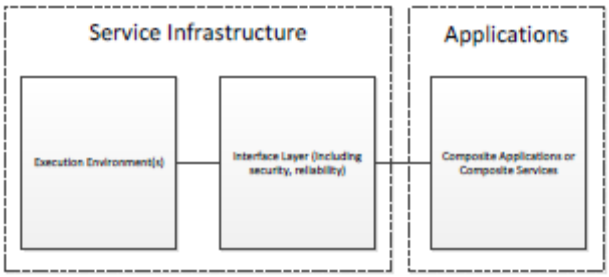

Figura 3-4. Dos mayores tareas de SOA [24].

La infraestructura de servicio también es llamada el Enterprise Service Bus (ESB) [45]. El ESB permite la comunicación de servicios entre sistemas heterogéneos con responsabilidades incluyendo transformación de datos, enrutamiento inteligente, tratando con seguridad y fiabilidad, servicio de monitoreo y gestión [45]. La infraestructura se compone de la capa interfaz y de entornos de ejecución, donde se ejecutan las aplicaciones SOA. Construir la interfaz requiere una perspectiva global a largo plazo. Debido a la doble naturaleza y características de SOA, el proceso de desarrollo de SOA difiere dramáticamente de un desarrollo de software tradicional [37].

Entregando aplicaciones empresariales SOA o creación de arquitectura empresarial con SOA requiere establecer una estrategia de entrega. El equilibrio correcto entre la migración a largo plazo y deben establecerse los requisitos a corto plazo. Existen diferentes estrategias de entrega SOA [28]. Por ejemplo, en un enfoque *top-down*, se realizan todas las fases del proceso de entrega de SOA desde principio a fin. Estrategias más ágiles pueden ser usadas para construir SOA en iteraciones más pequeñas. Fases de la estrategia de implementación SOA se describen en la Figura 3-5. Es notable, que estas fases son muy similares a las fases de un proceso de desarrollo de software tradicional [28].

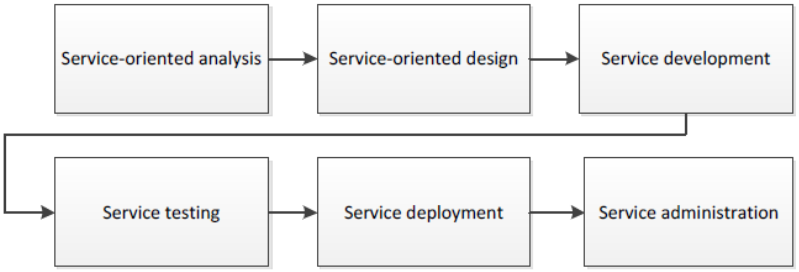

Figura 3-5. Proceso de entrega SOA [20].

Tradicionalmente, se han desarrollado sistemas de manera que el analista recoge los requisitos del negocio que luego son entregados a los arquitectos y desarrolladores, quienes luego construyen el sistema. En SOA, existe una relación directa entre el negocio y la tecnología, lo que significa cooperación intensiva entre los analistas y diseñadores [28]. Diseño y análisis orientado al servicio responden a preguntas: ¿Qué lógica debe ser representada por los servicios?, ¿Cómo los servicios deben relacionarse con la lógica de la aplicación existente?, ¿Cómo pueden los servicios representar mejor la lógica del proceso del negocio? y ¿Qué servicios pueden ser construidos y posicionados para promover la agilidad? [28].

El ***Análisis Orientado al Servicio*** es una fase donde se determina el alcance de SOA. Las capas de servicio y los servicios individuales son mapeados para formar un SOA preliminar [28]. ***Servicio Orientado al Diseño*** determina cómo están diseñados los servicios candidatos desde la fase de análisis mediante el uso de estándares, especificaciones de servicios Webs, y principios de diseño SOA [28]. El ***Desarrollo de Servicios*** incluye una plataforma de desarrollo específico en temas tales como se construyen los servicios que son producidos por las fases anteriores. Plataformas para la construcción de servicios incluyen .NET y J2EE [28]. ***Servicio de Prueba*** es una parte crucial en la entrega SOA que se realiza antes de implementar servicios [28]. ***Servicio de Despliegue*** está determinada por la plataforma tecnológica elegida. Esta fase incluye la instalación y configuración de componentes distribuidos, interfaces de servicio y posiblemente asociados con productos middleware en servidores de producción [28].

Administración de Servicios incluye aplicación en temas de gestión. Monitoreo de las propiedades no funcionales de los servicios, control de versión de la descripción de los servicios, y seguimiento y gestión de mensajes [28]. Gestión de composiciones de servicio complejos requieren herramientas convenientes para el análisis, pruebas y supervisión de los servicios [39]. Al realizar SOA también es importante identificar, que servicios son servicios de negocios que deben ser implementados internamente y que servicios deben ser adquiridos desde socios de negocio [121].

3.1.8. Calidad de Servicio (QoS)

Un requisito importante para una aplicación basada en SOA es el nivel deseado de calidad de servicio (QoS) que se refiere a veces como la calidad de servicio Web (QoWS) [120]. Significa la capacidad del servicio Web para realizar sus tareas basadas en las expectativas mutuas del proveedor del servicio y el consumidor. Factores de calidad necesitan ser considerados para mantener el negocio viable y competitivo. El cliente puede elegir un servicio Web basado en los atributos de QoS prometidos. Debido a la naturaleza impredecible y dinámica de Internet y el aumento de la complejidad de acceso y gestión de servicios, QoS se convierte en un desafío significativo [120]. Aplicaciones con diferentes características y requisitos compiten por recursos de la red. Cambios en los patrones de tráfico, fallos de hardware y problemas de confiabilidad en la web crean una necesidad para asegurar las transacciones de negocios de misión crítica y gestión de QoS [120].

Tradicionalmente, QoS se mide por el grado en que las aplicaciones, sistemas, redes y todos los demás elementos de la infraestructura IT soporten la disponibilidad de servicios en un nivel de desempeño bajo diferentes condiciones de acceso y carga requerido [120]. Centrarse sólo en las propiedades funcionales de los servicios individuales no es suficiente todo el entorno donde se alojan los servicios debe ser considerados. Esto significa que las capacidades no funcionales de servicios deben ser descritas [120]. En el contexto de servicios Web, QoS puede verse como dar seguridad en un conjunto de propiedades de calidad de servicios funcionales y no funcionales. El comportamiento general de un servicio Web debe entenderse para que otros servicios puedan usar como parte de un proceso de negocios. QoS tiempo de ejecución negociable es una funcionalidad avanzada de servicios web. Esto significa elegir dinámicamente un proveedor de servicios entre diferentes proveedores de servicios según criterios de calidad de servicio [120].

La calidad de servicio Web se puede dividir en dos tipos: calidad de ejecución y calidad de negocios. La calidad de ejecución significa capacidad del servicio para ejecutar sus operaciones [120]. La calidad de negocios permite la evaluación de la operación del servicio desde una perspectiva empresarial. Las mediciones tanto cuantitativas como cualitativas pueden utilizarse para evaluar la QoWS [120]. Apéndice B enumera los elementos claves para apoyar la QoS en entorno de servicios Web.

3.1.8.1. Gestión y Monitorización

El sistema de gestión de servicios Web (WMS) debe ser establecido con el fin de supervisar y controlar la calidad de los servicios web [120]. El seguimiento del comportamiento de un servicio Web es necesario para valorar sus parámetros de calidad de servicio y asegurar la garantía de que el servicio se mantiene en un nivel de calidad prometido.

Esta actividad se denomina gestión de monitoreo [120]. La calidad del servicio también se puede mejorar con un conjunto de mecanismos de control. El control de gestión incluye la transacción de servicios Web y la coordinación, optimización de servicios web y gestión del cambio. Las transacciones ayudan a mejorar la tolerancia a fallos y fiabilidad de los servicios Web [120].

La optimización de los servicios Web permite a los clientes identificar aquellos servicios que mejor satisfagan los requisitos deseados. En la gestión del cambio, ciertas acciones se llevan a cabo para identificar los cambios que se realizan en el sistema orientado al servicio y aprobar las operaciones apropiadas cuando sea necesario [120].

3.1.8.2. Acuerdo de Nivel de Servicio (SLAs)

Un acuerdo de nivel de servicio (SLAs) es un contrato formal entre el cliente y el proveedor de servicios [9]. SLAs describe los detalles del servicio Web incluyendo precios, contenido, proceso de entrega, aceptación y criterios de calidad y posibles sanciones por violar el acuerdo [9].

SLAs se utiliza para asegurar que el proveedor de servicios entregue un nivel garantizado de calidad del servicio. Además, describe cómo se mide la calidad del servicio [9]. Apéndice C representa los contenidos de un típico SLA.

3.1.9. Services Oriented Architecture Modelling Language (SoaML)

La Arquitectura Orientada a Servicios (SOA) es una forma de describir y entender las organizaciones, las comunidades y los sistemas para maximizar la agilidad, la escala y la interoperabilidad. El enfoque SOA es simple: las personas, las organizaciones y los sistemas proporcionan servicios entre sí. Estos servicios nos permiten hacer algo sin hacerlo nosotros mismos o incluso sin saber cómo hacerlo; que nos permite ser más eficientes y ágiles. Los servicios también nos permiten ofrecer nuestras capacidades a otros en cambio de algún valor, estableciendo así una comunidad, proceso o mercado. El paradigma SOA funciona igual de bien para la integración de las capacidades existentes, así como la creación y la integración de nuevas capacidades [83].

SoaML (Lenguaje de modelado de la Arquitectura Orientada a Servicio) es una especificación que proporciona un metamodelo y un perfil UML para la especificación y diseño de servicios dentro de una arquitectura orientada a servicios. Abarca extensiones de UML 2.1.2 para soportar el modelado de las siguientes capacidades [83]:

- Identificación de servicios, los requisitos que deben cumplir, y las dependencias anticipadas entre ellos.
- Especificación de servicios incluyendo las capacidades funcionales que proporcionan, capacidades que se espera que los consumidores proporcionen, los protocolos o reglas para el uso de ellos, y la información de servicio intercambiada entre los consumidores y proveedores.
- Definición de servicios de consumidores y proveedores, solicitudes y servicios que consumen y proveen, cómo están conectados y cómo las capacidades funcionales del servicio son utilizadas por los consumidores e implementadas por los proveedores de una manera compatible con los protocolos de especificación de servicio y los requisitos cumplidos.
- Las políticas para el uso y prestación de servicios.
- La habilidad para definir esquemas de clasificación teniendo aspectos para apoyar una amplia gama arquitecturas, esquemas de particionamiento físico y organizacional, y restricciones.
- Definición de servicio y requerimientos de uso de servicio, y vincularlos a metamodelos relacionados a la OMG, tales como el BMM, BPDM, UPDM o UML.
- SoaML se centra en los conceptos de modelado de servicios básicos y la intención es usar esto como una base para las extensiones relacionadas con la integración con otros metamodelos de la OMG como BPDM y BPMN 2.0, así como SBVR, OSM, ODM y otros.

El concepto clave de SoaML es el *servicio*. Un servicio es el intercambio de valor desde un participante a otro. El conocimiento de cómo usar el servicio y el acceso a utilizar es proporcionado por un contrato de servicio. En un servicio existe un proveedor de servicios y un consumidor de servicios, uno de los cuales es el iniciador de la interacción de servicios. Un servicio viene dado por un participante, y el participante puede ser cualquier entidad: un ser humano, una organización o un sistema informático.

3.1.9.1. Uso

SoaML integra capacidades de modelado para apoyar usando SOA en diferentes niveles y con diferentes metodologías. En particular soporta un enfoque "basado en contrato" y un enfoque "basado en interfaz ", que en general, siguen los elementos del "ServiceContract" y "ServiceInterface" del perfil SoaML, respectivamente.

SoaML soporta diferentes enfoques para SOA, lo que resulta diferente, pero que coinciden con los elementos del perfil. Antes de abordar las diferencias, vamos a revisar algunas de las similitudes y terminología.

Participantes: Los participantes son entidades específicas o clases de entidades que ofrecen o utilizan los servicios. Los participantes pueden representar personas, organizaciones o componentes del sistema de información.

Los participantes pueden proporcionar cualquier número de servicios y pueden consumir cualquier número de servicios [48].

Puertos: Los participantes proporcionan o consuman servicios a través de los puertos. Un *puerto* es la parte o característica de un participante, es el punto de interacción para un servicio, donde es provisto o consumido. Un puerto donde se ofrece un servicio es llamado como un "puerto de servicio", y el puerto donde se consume un servicio es llamado como un "puerto de respuesta" [48].

Descripción del servicio: Es la descripción de cómo interactúa el participante para proporcionar o utilizar un servicio, está encapsulado en una especificación para el servicio. Hay tres formas de especificar una interacción de servicio: una interfaz de UML, una ServiceInterface y un ServiceContract. Estas diferentes formas de especificar un servicio se relacionan con el enfoque SOA y la complejidad del servicio, pero en cada caso resultan en comportamientos que definen cómo el participante proporcionará o utilizará un servicio a través de puertos e interfaces. Las descripciones del servicio son independientes, pero consistentes cómo el proveedor proporciona el servicio o cómo (o por) el consumidor lo consume. Esta separación de preocupaciones entre la descripción del servicio y cómo se implementa es fundamental para SOA. Una especificación de servicio especifica cómo los consumidores y los proveedores se espera que interactúan a través de sus puertos para representar un servicio, pero no cómo lo hacen [48].

Capacidades: Los participantes que proveen un servicio deben tener una capacidad de proporcionarlo, pero distintos proveedores pueden tener diferentes capacidades para ofrecer el mismo servicio, algunos incluso pueden "externalizar" o delegar la implementación del servicio a través de una solicitud de servicios a otros. La capacidad detrás del servicio proporcionará el servicio según la descripción del servicio, y también puede tener las dependencias de otros servicios para proporcionar esa capacidad. La capacidad de servicio es con frecuencia parte integral de los procesos de negocio del proveedor. Las capacidades pueden verse desde dos perspectivas, capacidades que tiene un participante que pueden ser explotadas para proporcionar servicios, y capacidades que necesita una empresa que pueden utilizarse para identificar servicios candidatos [48].

3.1.9.2. Enfoques

SoaML ofrece dos enfoques principales para la Arquitectura Orientada a Servicios: los contratos de servicio, y las interfaces de servicios basadas en SOA. Además, el modelador puede utilizar el método de interfaz simple.

El enfoque de la ***simple interface*** es la conocida interfaz del lenguaje orientado a objetos pero utilizado típicamente en un entorno de servicios Web. Existen dos tipos de usos: estilo documento y estilo RPC. La primera es una sola operación típica de datos y en el uso de la interfaz no es necesario ningún protocolo. El último es más estilo OO con múltiples variables de entrada y valor devuelto [48].

El enfoque de interfaz es una comunicación unidireccional donde el proveedor del servicio no es necesario conocer al consumidor. Este enfoque sigue siendo importante tener en cuenta cuando se diseña las interfaces y usan mensaje tipos en una comunicación.

Figura 3-6. Interfaz simple [48].

Figura 3-7. Uso de una interfaz simple [48].

El enfoque del ***ServiceInterface*** amplía la interfaz simple con una interfaz bidireccional. La ServiceInterface es en sí misma una construcción abstracta que realiza una interfaz y otros usos. La interfaz proporcionada es aquella que es realizada y la interfaz que es usada usa una funcionalidad "callback" para la comunicación asincrónica. El proveedor de un servicio define una ServiceInterface y posiblemente un protocolo para su uso, y proporciona este servicio a través de un ServicePort. Un consumidor entonces se adhiere a esta interfaz de servicio con el uso de la misma, pero opuesta a las interfaces o un conjunto de interfaces compatibles a través de un RequestPort [48].

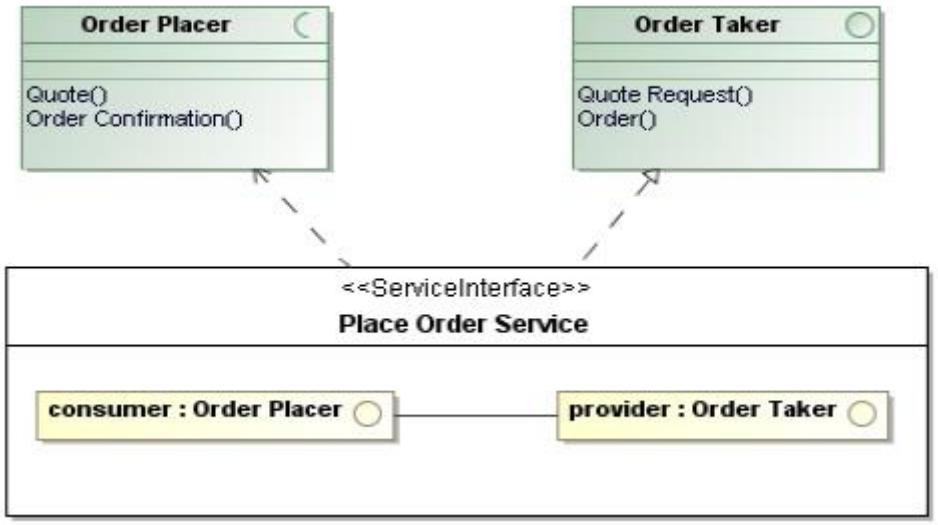

Figura 3-8. Definición de una ServiceInterface [48].

El enfoque *ServiceContract* define un contrato que especifica cómo los proveedores y consumidores trabajen juntos para el intercambio de valor. Además, define los roles de que cada participante desempeña en el servicio (tales como proveedor y consumidor) y las interfaces que implementan para desempeñar ese papel en el servicio. Estas interfaces son entonces los tipos de puertos en el participante, que obliga a los participantes para poder desempeñar ese papel en el contrato de servicio. El contrato de servicio representa un acuerdo entre las partes para que el servicio sea proporcionado y consumido. Este acuerdo incluye las interfaces, coreografía y cualquier otro término y condiciones [48].

Figura 3-9. Ejemplo de un ServiceContract [48].

3.1.9.3. Soporte Top-down / Bottom-up

SoaML puede ser utilizado para servicios independientes de contexto básico como "Get stock quote" o "get time", ejemplos populares en servicios Web. Los servicios básicos se centran en la especificación de un único servicio sin tener en cuenta su contexto o dependencias. Desde un servicio básico es independiente del contexto puede ser más simple y más apropiados para la definición de "bottom-up" de los servicios.

SoaML puede usarse "in the large" donde se permite a una organización o comunidad trabajar más eficazmente utilizando una serie de servicios interrelacionados. Dichos servicios se ejecutan en el contexto de la empresa, proceso o la comunidad y así dependen de acuerdos capturados en la arquitectura de servicios de esa comunidad. Un ServicesArchitecture muestra cómo varios participantes trabajan juntos, proporcionan y utilizan servicios para permitir que metas o procesos.

En cualquier caso, los servicios de tecnología pueden ser identificados, especificados, implementados y finalmente realizados en un entorno de ejecución. Hay una variedad de enfoques para la identificación de los servicios que son compatibles con SoaML. Estos diferentes enfoques están destinados a apoyar la variabilidad en el mercado. Los servicios pueden identificarse mediante:

- Diseñar una arquitectura de servicios específica una comunidad de interacción de los participantes y los contratos de servicio que reflejan los acuerdos de cómo van a interactuar con el fin de lograr un propósito común.

- Organizar las funciones individuales en las capacidades dispone en una jerarquía mostrando las dependencias de uso previsto, y utilizando esas capacidades para identificar las interfaces de servicio que exponen a través de servicios.
- Utilizando un proceso de negocios para identificar las capacidades funcionales necesarias para lograr un propósito, así como los roles interpretados por los participantes. Los procesos y servicios son diferentes desde la misma vista del sistema, se centra en cómo y por qué las partes interactúan para proporcionar unos a otros con productos y servicios y otros se enfoquen en qué partes se realizan las actividades para proporcionar y utilizar esos servicios.
- Identificación de servicios de los activos existentes que pueden utilizarse por los participantes para adaptar esas capacidades y exponerlos como servicios.
- Identificación de datos comunes y los flujos de datos entre las partes y agrupar estos servicios.

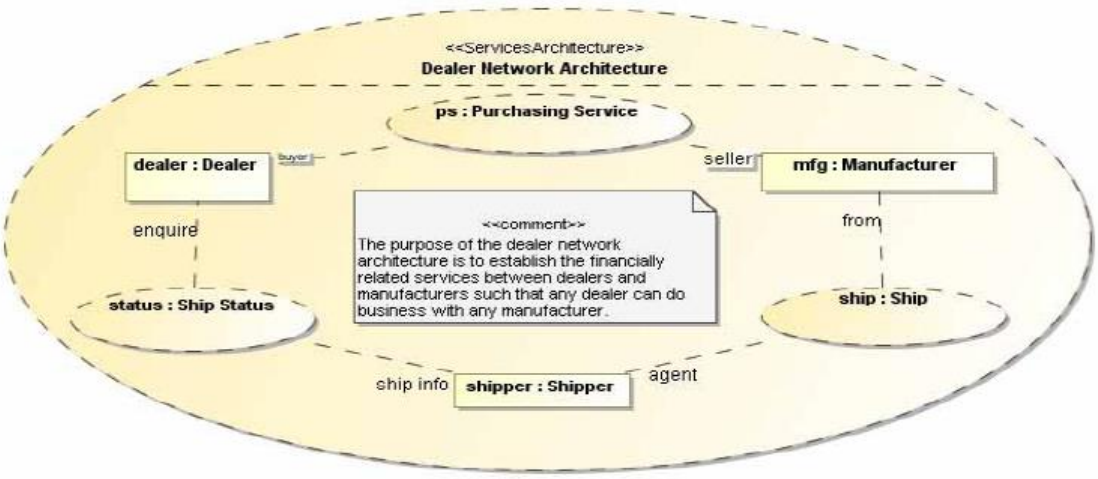

Figura 3-10. Ejemplo de una comunidad de arquitectura de servicios con participantes y servicios [48].

3.1.9.4. Herramientas de soporte

Las herramientas disponibles para dar soporte a la especificación, son las siguientes:

- IBM Rational Software Architect [41],
- ModelPro [72],
- Modelio Case Tool [73],
- Magic Draw [43],
- Enterprise Architect [103],
- Proyecto Minerva [23].

3.1.10. Beneficios de SOA

SOA organiza los sistemas de modo que ejecuten necesidades del negocio de manera eficiente. Si fue implementado correctamente, SOA resulta una arquitectura empresarial que es ágil y reutilizable.

Esto significa que se puede utilizar una y otra vez la misma lógica de aplicación y los procesos de negocio pueden cambiarse sin los cambios necesarios en todos los sistemas [28], [29]. En la figura 6, se muestra tanto el modelo tradicional y el modelo de la automatización de un proceso de negocios SOA.

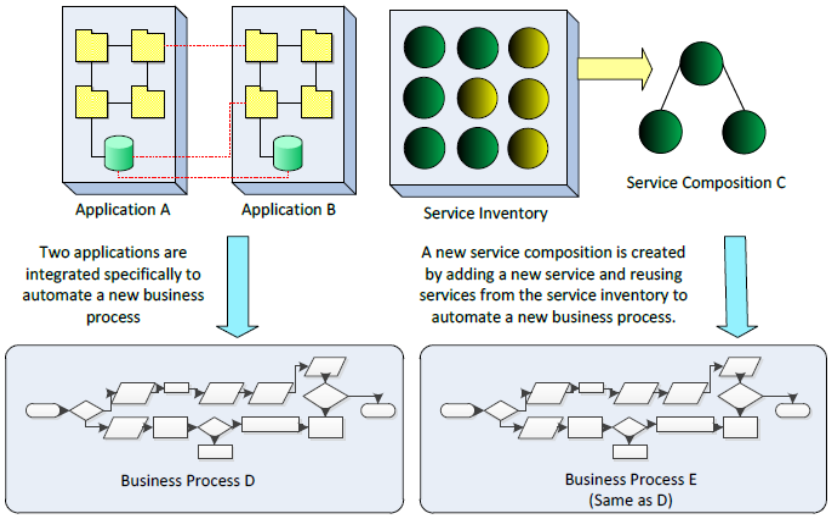

Figura 3-11. Modelo Tradicional vs. Modelo SOA [19].

Para automatizar tradicionalmente los procesos de negocio D, se integran dos aplicaciones con varias conexiones punto a punto. Esto es problemático si el proceso del negocio cambia y aparece la necesidad de nuevas funcionalidades en el futuro. Como resultado, habrá un sistema que es difícil de administrar. Con el fin de automatizar los procesos de negocio E (que son los mismo que el D) pueden utilizarse los servicios existentes encontrados en inventario de servicios [28], [29].

La cantidad de nuevo código escrito es probablemente menos que en el modelo tradicional tan sólo unas poca cantidad de nueva funcionalidad es necesario ser creado. Como resultado, a la cuestión de la reacción rápida a los cambios en la actividad empresarial se ha cumplido con el sistema con el que es menos complicado de gestionar [28], [29].

Es importante tener en cuenta que SOA no es una arquitectura técnica compuesta por servicios Web, o simplemente implementar los servicios web no conduce a todos los beneficios que pueden lograrse a través de una verdadera SOA. En su lugar, SOA requiere un cambio de lógica de negocios en el que se ve desde un contexto orientado hacia el servicio [28], [29]. Los factores claves para del éxito de SOA son: la comprensión, la gobernanza, la gestión y el apoyo [45].

Siguiendo una transformación *top-down* y la aprobación de un cambio cultural con visión y compromiso es la forma de lograr una verdadera arquitectura orientada a servicios con sus beneficios. Las siete metas (representado en la figura 3-7) de SOA pueden categorizarse en dos grupos, los objetivos estratégicos y beneficios resultantes [28], [29].

Los objetivos estratégicos son aumento de la federación, aumento de la interoperabilidad intrínseca, incremento de las opciones de diversidad de proveedores, aumento de negocios y alineación de la tecnología. Las tres ventajas resultantes pueden llegar, dependiendo de cómo sean realizadas [28], [29].

Estos beneficios incrementan el ROI (Return of Invesment), la reducción de la carga de TI y un incremento en la agilidad organizacional [28], [29].

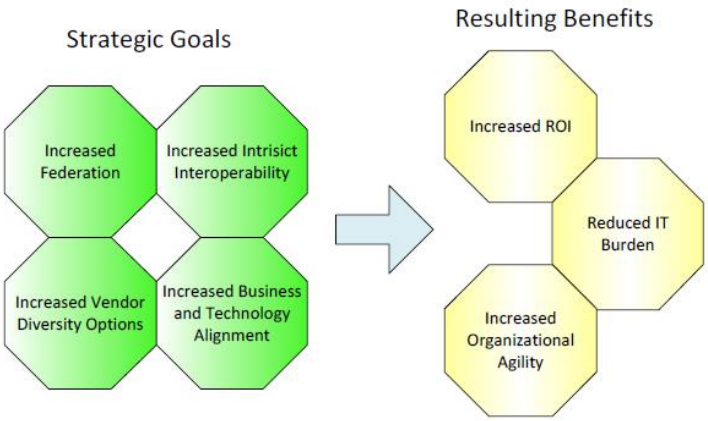

Figura 3-12. Beneficios de SOA [19].

SOA requiere un cambio significativo del clima organizacional, así como una cantidad significativa de tiempo y esfuerzo. SOA no es una panacea y no es conveniente para todas las situaciones [45]. Algunas organizaciones han sufrido decepciones con sus proyectos SOA [37].

También, se pueden encontrar críticas contra SOA en algunos blogs. Algunos expertos [15], [58] dicen que SOA no ha podido cumplir las promesas y expectativas que se han establecido para ello. Todavía creen que aunque SOA no exista ya sus principios se quedarán, y serán adoptados por nuevas tecnologías, como SaaS y Cloud Computing.

Según Haines y Rothenberg, la pregunta más importante es cómo SOA puede hacerse bien en una situación actual o es posible o razonable hacerlo en absoluto [37].

3.1.11. Limitaciones de SOA

En un mundo tecnológico de constantes cambios, se espera que las empresas provean más con menos recursos. SOA provee a las empresas de desarrollo de software la habilidad de responder rápida y eficientemente a las solicitudes de servicio. Sin embargo, SOA no es compatible con todas las aplicaciones. A continuación se detallan las principales limitaciones de esta arquitectura:

- SOA depende de la implementación de estándares. Sin estándares, la comunicación entre aplicaciones requiere de mucho tiempo y código.
- SOA no es para: aplicaciones con alto nivel de transferencia de datos, aplicaciones que no requieren de implementación del tipo *request/response* y para aplicaciones que tienen un corto periodo de vida.

- Incrementalmente se hace difícil y costoso el ser capaz de cumplir con los protocolos y hablar con un servicio.
- Implica conocer los procesos del negocio, clasificarlos, extraer las funciones que son comunes a ellos, estandarizarlas y formar con ellas capas de servicios que serán requeridas por cualquier proceso de negocio.
- En la medida en que un servicio de negocio, vaya siendo incorporado en la definición de los procesos de negocio, dicho servicio aumentara su nivel de criticidad. Con lo cual cada vez que se requiera efectuar una actualización en dicho servicio (por ejemplo, un cambio en el código, una interfaz nueva, etc.), deberá evaluarse previamente el impacto y tener mucho cuidado con su implementación. Sin embargo, parte de la problemática anterior, puede ser solventada en virtud a un buen diseño del servicio.

3.2. Cloud Computing

Cloud Computing o computación en la nube es una de las mayores tendencias de IT en el momento. Es el resultado de décadas de investigación y los avances en las tecnologías de virtualización, sistemas paralelos y distribuidos, utilidad informática así como los avances recientes en redes, servicios Web y SOA. La idea principal detrás de Cloud Computing es proporcionar a la computación como una utilidad. La naturaleza de la utilidad es que los usuarios accedan cuando se necesitan y no importa de dónde venga o cómo se entrega [14], [35]. En 1961, se sugirió que la tecnología informática en tiempo compartido podría llevar a un futuro donde el poder de la computación e incluso de específicas aplicaciones podría ser vendido a través de un modelo de utilidad tipo de negocio [96].

El tiempo compartido fue una elección razonable debido a los altos costos de la computación y la necesidad de las personas especializadas en mantener los sistemas. Debido a la evolución del hardware, software, protocolos de red y comunicaciones; una cantidad significativa de capacidad computacional no utilizada en los centros de datos de las organizaciones; infraestructura de comunicaciones de datos de Internet de alta velocidad, así como las empresas necesitan concentrarse en sus competencias básicas, tiempo compartido renace con un nuevo nombre, el Cloud Computing [26].

Cloud Computing se ve muy afectado por otros paradigmas y arquitecturas y tecnologías, tales como Grid Computing, procesamiento paralelo, la computación en clúster y virtualización de servidores. Tecnologías y conceptos como Clúster Computing, Grid Computing y Utility Computing pueden considerarse como bloques de construcción con el fin de lograr el modelo de Cloud Computing moderno que representa el siguiente paso natural en la evolución de la informática y servicios de TI.

La computación en nube se dice que tiene el potencial de crear un cambio de paradigma en la forma los recursos de TI se utilizan y distribuyen. También es la causa del cambio de modelo de entrega de aplicaciones de escritorio tradicionales para modelo de software como servicio [14], [35].

Cloud Computing es un nuevo modelo de provisión y soporte IT que proporciona acceso a la red bajo demanda a una piscina de recursos informáticos compartidos. Esta piscina de recursos es llamada *nube*. La nube puede ubicarse en uno o más centros de datos privados, en instalaciones operadas por terceros prestadores de servicios o a través de la combinación de estos dos [105]. Diferentes definiciones para Cloud Computing se pueden encontrar en la literatura.

Según Kommalapati, Cloud Computing es *"la capacidad de cómputo entregada como una utilidad a través de protocolos y estándares de Internet"* [52]. La definición de Cloud Computing se prevé que evolucionará a medida de como las soluciones en la nube se vuelvan más maduras y aparezcan nuevos modelos de servicio [97].

Vaquero & Al., determinó la necesidad de establecer una definición clara de Cloud Computing mediante la recopilación de más de 20 definiciones, y hacer su propia identificando los elementos claves que aparecieron en la mayor parte de las definiciones existentes [114].

Según el NIST (National Institute of Standards and Technology) Cloud Computing es *"un modelo ubicuo, conveniente y bajo demanda mediante red a un conjunto compartido de recursos de cómputo configurables (i.e., redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provisionados y liberados con un mínima esfuerzo de gestión o interacción con el proveedor del servicio"* [59].

3.2.1. Propiedades de Cloud Computing

Los modernos entornos IT requieren formas para aumentar la capacidad y agregar capacidades en infraestructura dinámicamente, sin invertir dinero en la compra de nueva infraestructura, capacitación del nuevo personal y licencias de nuevo software. Cloud Computing puede ofrecer una solución a estas necesidades con el siguiente conjunto de características de la computación en nube [96].

Acceso bajo demanda significa el rápido cumplimiento de la demanda informática como sea necesaria y una habilidad para satisfacer esa necesidad al instante. ***Elasticidad*** es la cantidad de recursos informáticos requeridos y que se desechan cuando ya no son necesarios. ***Pago por uso*** significa que un servicio en la nube tiene un modelo de precios que se cobra al cliente basado en la cantidad de recursos informáticos utilizados. ***Conectividad*** significa que todos los servidores están conectados a una red de alta velocidad que permite el flujo de datos a Internet (para y desde la nube) entre los recursos informáticos y el almacenamiento de elementos que está dentro de la nube [26],[96].

Piscina de recursos implica que la nube es compartida por muchos clientes proporcionando economía de escala en las capas de computación y servicios. *Infraestructura abstraída* esconde la ubicación exacta y el tipo de hardware en la nube que apuntan a las aplicaciones del cliente [26], [96].

El proveedor de la nube ofrece las métricas de desempeño para garantizar un nivel mínimo de rendimiento. *Poco o ningún compromiso* significa que mediante el uso de una solución Cloud, algunas de las responsabilidades de mantenimiento y los compromisos de recursos se puedan mover al vendedor [26], [96].

3.2.1.1. Beneficios de Cloud Computing

Cloud Computing es también un término muy polémico y se ha convertido en una verdadera moda en los medios de comunicación. Para algunos, podría parecer como muy orientada a la comercialización término inventado por los vendedores [96]. Técnicamente, la nube informática ofrece nada nuevo, y similares promesas han sido escuchadas antes por gestión de los recursos asignados, Grid Computing, Computación bajo demanda y clúster de computación [96]. También está generando una gran confusión acerca de lo que en realidad significa es sólo algunas viejas ideas que se venden bajo un nuevo mandato de la computación en la nube. Con el fin de obtener los beneficios de Cloud Computing, es importante entender lo que realmente es, cómo ha evolucionado y qué tipo de oportunidades son ofrecidos por los proveedores de la nube de hoy [96].

La principal motivación para la utilización de Cloud Computing en una empresa es la reducción de los costes totales de propiedad de los centros de datos [105]. Los gastos de capital y los costes de mantenimiento se reducen ya que los recursos IT se centralizan en una nube y la necesidad de hardware propio especializado disminuye. Esto puede significar grandes ahorros para una compañía que ahora es capaz de utilizar el mayor procesamiento de datos y la capacidad de almacenamiento sin tener que invertir grandes cantidades de capital [56]. Empresas que pagando sólo por la cantidad de recursos informáticos y almacenamiento que se utiliza en realidad, no están obligadas a tener que organizar un ambiente que puede manejar las necesidades cambiantes. El modelo de facturación también permite a una empresa ahorrar en costos en contratos de proveedores largos que ya no son necesarios [56].

Las soluciones en la nube pueden ofrecer infraestructuras flexibles, escalables y distribuidas que permiten que las aplicaciones *escalar* bajo demanda [56]. Instantáneamente pueden implementar entornos para necesidades específicas (por ejemplo en cuanto a pruebas de software) se pueden implementar en plazos más cortos que antes, permite a una empresa una entrada más rápida al mercado [56]. Cloud Computing está causando una transformación del departamento IT desde mantenimiento y puesta en práctica centrada en la innovación de la computación en la nube [56], [105] [105].

El tiempo que se dedica en la personalización de los marcos de aplicación o construcción y mantenimiento de infraestructuras IT podrían utilizarse para crear más valor al centrarse en competencias básicas y mejorar la lógica de negocio [96].

Las empresas con grandes centros de datos se han dado cuenta que la nube vende su capacidad como un servicio adicional y aumenta la eficiencia del servidor y la utilización produciendo un mayor ROI [56], [105].

Como pequeñas y medianas empresas no podrán necesariamente invertir en sus propios centros de datos, y tal vez no tengan los conocimientos necesarios para administrarlos, la externalización de los centros de datos es cada vez más popular en forma de la computación en la nube [105]. Pequeñas y medianas empresas pueden entrar a la misma competición con grandes empresas, porque ahora son capaces de acceder a grandes recursos computacionales con ninguna inversión de capital y porque estos recursos ahora se pueden acceder desde cualquier lugar y en cualquier momento [36].

3.2.1.2. Riesgos de Cloud Computing

Además de todos los beneficios y oportunidades, Cloud Computing presenta varios riesgos y desafíos que deben evaluarse. Estos temas incluyen seguridad, falta de control, cuestiones de privacidad, integridad de datos y disponibilidad así como aceptabilidad de negocios [36].

La externalización de parte del propio funcionamiento a terceros provoca falta de control. Hay que tener en cuenta, ¿Cómo va el negocio a conservar y mantener el control de datos, y manejar el impacto del tiempo de inactividad posible?, así como cambios tecnológicos posibles o las decisiones tomadas por el proveedor de nube [36]. Siempre hay una posibilidad de que la información confidencial terminará en manos equivocadas.

La integridad y confidencialidad de los datos son también notables preocupaciones, hace que un negocio mantenga la privacidad de sus usuarios y hace que los datos valiosos queden intactos en un entorno de nube [36]. Las soluciones en la nube crítica empresariales deben estar disponibles todo el tiempo y el proveedor de servicios de alguna manera debe garantizar la disponibilidad. También debe estimarse la idoneidad de la solución de terceros [6]. Si pensando en una mayor escala de idoneidad, en regiones donde hay muchos recursos IT y el ancho de banda, y están haciendo frente con pocos conocimientos técnicos, la utilización de Cloud Computing puede ser difícil [36].

3.2.2. Arquitectura de Cloud Computing

La virtualización es muy a menudo relacionada con Cloud Computing. Sin embargo, técnicamente es posible construir una nube sin la tecnología de virtualización, especialmente en los casos de SaaS [105]. La flexibilidad y la automatización de la virtualización son los elementos clave que crean un alto nivel de adaptabilidad y entornos Cloud dinámicos que son capaces de necesitar diversas necesidades de tipos de negocios que pueden ser personalizados por el usuario final [105].

La elasticidad del Cloud Computing también es posible por la tecnología de virtualización. Los componentes típicos de la nube incluyen software de virtualización, balanceadores de carga y dispositivos de red. Los balanceadores de carga pueden utilizar de forma dinámica las peticiones de la aplicación directa del usuario a los servidores que pueden manejar mejor las solicitudes en el momento[105].

Los centro de datos de redes de alta capacidad son necesarios para proporcionar la constante reorganización de los recursos de la red que se puede acceder a través de LAN (red de área local), SAN (red de área de almacenamiento), y WAN (red de área amplia) [105].

Cloud Computing establece importantes demandas en la infraestructura de red y las actualizaciones pueden ser necesarias a fin de prestar apoyo a una mayor cantidad tráfico de la red elástico y en tiempo real [105].

La idea de la virtualización es simple: un servidor con CPU (Unidad Central de Procesamiento), discos e interfaz de red pasan gran cantidad de tiempo funcionando en vacío. Por tanto, es razonable utilizar recursos gratuitos para ofrecer más servicios con el mismo hardware [105]. Con la virtualización, el sistema operativo y las aplicaciones de software están desvinculados de la plataforma de hardware subyacente y de las demás máquinas virtuales [105]. Los recursos de un servidor físico pueden ser usados para crear máquinas virtuales diferentes, lo que permite crear muchos dominios de usuario en una máquina [105].

El software de virtualización puede ser migrado en una nube entre varios servidores sin la más mínima interrupción o incluso se puede mover entre diferentes nubes, siempre que los ciclos informáticos estén disponibles en el área de infraestructura física en el momento. Esto permite que los servidores y centros de datos ser consolidados y la tasa de utilización de los servidores físicos sea incrementada [105].

El software de virtualización reside entre el equipo físico y el sistema operativo lo que permite que un único servidor físico u otra pieza de hardware, como por ejemplo disco duro o el dispositivo de red aparezcan como muchos dispositivos distintos. En un centro de datos moderno es típico utilizar dispositivos de red virtualizados [104].

El usuario final puede instalar las aplicaciones deseadas en el entorno de la nube. Los proveedores pueden ofrecer el acceso a las aplicaciones que se ejecutan en máquinas virtuales o pueden proporcionar acceso a máquinas virtuales [14]. Los proveedores tienen que gestionar el entorno virtualizado de modo que la asignación inteligente de recursos físicos en función de las demandas de recursos se gestionen correctamente [14].

Los consumidores de servicios de Cloud Computing exigen tiempos de respuesta rápidos. Distribuir servicios de Cloud Computing en diversos lugares y dividir la carga de trabajo de solicitud de estos lugares puede resultar un mejor tiempo de respuesta [14].

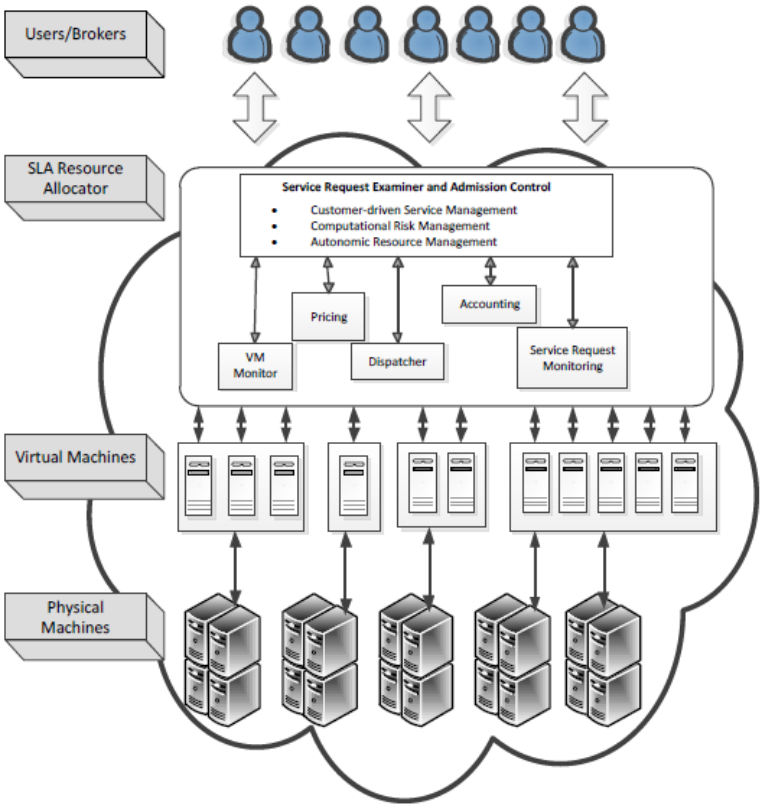

Figura 3-13. Una arquitectura de alto nivel de una economía orientada hacia el mercado Cloud [14].

En la figura 3-8, los usuarios y los intermediarios acceden a la nube a través de la Internet. El recurso SLA Allocator es la interfaz entre el centro de datos y sus usuarios. Se requiere el soporte de varios mecanismos que permitan a SLA la administración de los recursos. Cuando una petición llega, el examinador de peticiones y el mecanismo de control de admisión examinan los requisitos de QoS (como tiempo, coste, fiabilidad, confianza, seguridad) de la petición antes de determinar sí aceptar o rechazar la solicitud [14].

El mecanismo de monitorización de la máquina virtual da la última información del estado acerca de la disponibilidad de recursos. Información sobre el mecanismo de carga de trabajo es también necesaria para el mecanismo monitor de peticiones. A continuación, la petición se asigna a VMs [14]. El mecanismo de fijación de precios determina, cómo la petición es cobrada. La fijación de precios puede estar basada en la hora de envío (pico/no pico), precios (fijo/cambiante) o la disponibilidad de recursos (oferta/demanda) [14]. El mecanismo de contabilidad realiza un seguimiento de la utilización real de los recursos para que el costo total pueda ser calculado y cargado a los usuarios. El monitor VMs realiza un seguimiento de la disponibilidad de VMs y su derecho de recurso. El despachador comienza la ejecución de la solicitud de servicio que es aceptada y asignada a VMs [14]. El monitor de peticiones hace un seguimiento de los progresos de la solicitud de servicio ejecutado.

Cualquier cantidad de máquinas virtuales que residen en un único equipo físico pueden ser iniciadas y paradas en la demanda para cumplir con las peticiones de servicio aceptado. Las máquinas virtuales proporcionan flexibilidad para configurar los recursos en la misma máquina física a diversos requisitos de las solicitudes de servicio, porqué las VMs están aisladas unas de otras, y pueden correr aplicaciones basadas en diferentes sistemas operativos simultáneamente en una única máquina física [14].

3.2.3. Tipos de Cloud Computing

Hay diferentes tipos de nubes: privadas, públicas, comunidades/federadas e híbrida. Las ***nubes privadas*** son para uso exclusivo de una empresa o un consorcio de empresas [51]. En este caso, la nube puede ser gestionada por la propia organización o por un tercero. Según Kommalapati, *"cualquier centro de datos que esté a cargo de una empresa de gran tamaño puede llamarse una nube privada si utiliza el modelo de recursos unificados permitido por la virtualización, almacenamiento y redes como una piscina de recursos homogéneos y aprovecha los procesos altamente automatizados para el sistema operativo"* [51].

Las ***nubes públicas*** están disponibles para cualquier persona. La infraestructura de la nube está en manos de una organización gran tamaño como Amazon, Google, IBM o Microsoft. La plataforma Windows Azure, Amazon Web Services, Google App Engine y Force.com son ejemplos de nubes públicas [51].

La ***nube híbrida*** es una combinación de los servicios de la nube privada y pública. Una empresa que está operando su propia nube privada puede obtener algunos recursos de las nubes públicas, por ejemplo, en una situación cuando la potencia de procesamiento necesaria excede las capacidades propias. Esto se llama *cloudbursting* [57]. Esta mezcla de modelos de Cloud puede cambiar dinámicamente como cambian los requerimientos del cliente. Las nubes comunitarias/federadas son compartidas por varias organizaciones que tienen intereses y necesidades similares. Por ejemplo, el personal de seguridad pública podría compartir las nubes de respuesta de emergencia para actividades de socorro en casos de desastre [57].

3.2.4. Modelos de Servicio en Cloud Computing

Existen diferentes modelos de servicio de Cloud Computing, en función del tipo de la función proporcionada por la nube. Infraestructura como servicio (IaaS), Plataforma como servicio (PaaS) y Software como servicio (SaaS) son los principales modelos de servicio de Cloud Computing introducido en la mayor parte de la literatura. La computación bajo demanda es un modelo de empresa en el que los recursos informáticos se pondrán a disposición de los usuarios, según sea necesario.

Tradicionalmente, los recursos informáticos se mantienen de lado del cliente pero en el modelo de demanda, los recursos son en su mayoría mantenidos por el proveedor de servicios [96]. Como los recursos se obtienen a partir de una nube, la responsabilidad de administrar los recursos cambia desde el cliente al proveedor [19].

La figura 3-9 muestra la responsabilidad de gestión de ciertos activos de IT en diferentes modelos de servicio Cloud.

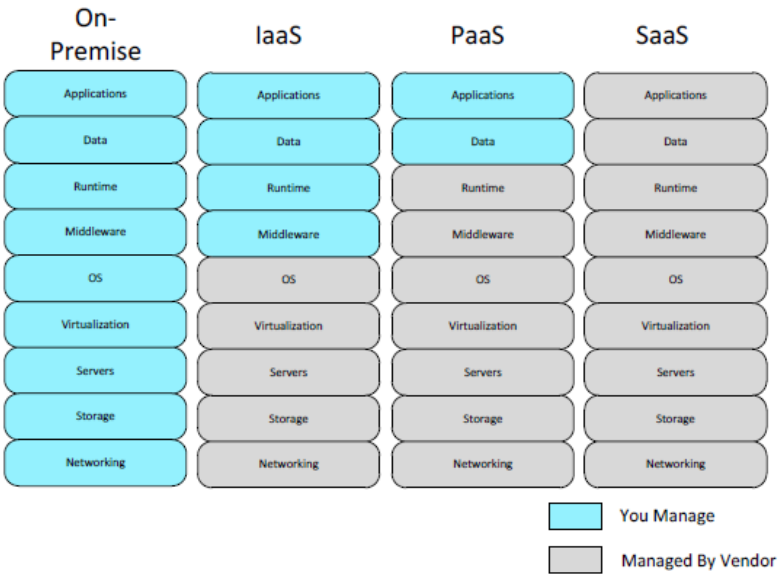

Figura 3-14. Responsabilidad de gestión en diferentes modelos de servicio Cloud [15].

El término XaaS (Everything as a Service) puede verse muy a menudo en la literatura de computación en nube. Estas no son consideradas en las definiciones actuales de Cloud Computing solo es una tendencia evolutiva. Algunos investigadores creen que la computación en la nube en el futuro abarcará modelos de servicio adicional, por ejemplo HuaaS (Humans as a service) y STaaS (Software Testing as a Service) [97].

3.2.4.1. Infraestructura como Servicio (IaaS)

En el modelo IaaS, la infraestructura informática predefinida y estandarizada, que suele ser un entorno de virtualización de plataformas, se entrega como un servicio. El proveedor de IaaS da un cliente de la empresa una base de hardware genérico en que un cliente puede instalar y ejecutar sus propios sistemas operativos, aplicaciones y datos de almacenamiento [105]. Esta función consta de hardware informático (que generalmente es una red escalable), red (routers, firewalls, balanceo de carga), conectividad a Internet, funcionamiento de plataformas de entorno de virtualización para máquinas virtuales que son especificados por el cliente y los SLAs y facturación basada en el uso (pago por uso) [14].

IaaS generalmente se ofrece como un servicio administrado. Una empresa alquila servidores físicos y virtuales como sea necesario, y el cliente tiene un completo control de la configuración del software, lo que significa tanto la propiedad y la gestión de la aplicación. Los proveedores de IaaS, ejecutan y mantienen la plataforma y gestionar la transición de las aplicaciones del cliente en la infraestructura. Los proveedores también son responsables de la gestión de las infraestructuras y la descarga de las operaciones [105].

IaaS evita a los clientes realizar inversiones en servidores, dispositivos de almacenamiento y conexiones de red, así como los clientes crecen y cambian los requisitos [105]. De esta forma, los clientes solo alquilan dichos componentes como un servicio externalizado, y el cliente sólo paga por los recursos consumidos, por ejemplo, una vez al mes. Los clientes son capaces de crear, probar y desplegar aplicaciones con plena conciencia de las configuraciones de hardware y software de los servidores alojados por los proveedores IaaS. IaaS permite a los clientes transferir las aplicaciones heredadas a la nube [51].

Diferentes infraestructuras proporcionan diferentes abstracciones y mecanismos para describir y acceder a las máquinas virtuales y al almacenamiento. La compatibilidad entre los diferentes proveedores IaaS puede convertirse en un problema. Los desarrolladores que utilizan IaaS también tienen que considerar algunos detalles específicos de la plataforma, por ejemplo, el apoyo para el paralelismo y detalles de E/S ofrecidos por un proveedor IaaS antes de mover aplicaciones a la nube [57].

3.2.4.2. Plataforma como Servicio (PaaS)

Las aplicaciones necesitan una plataforma en la que se ejecuten. La plataforma ofrece servicios a los desarrolladores para crear aplicaciones y almacenar datos [17]. PaaS (Platform as a Service) es otro tipo de servicio en la nube no administrado. PaaS ofrece todos los componentes necesarios, incluyendo bibliotecas, herramientas, aplicaciones, interfaces de programación de aplicaciones (APIs), protocolos, sistemas operativos, y sistemas de almacenamiento que se requieren para el completo ciclo de vida de construcción y despliegue de las aplicaciones Web desde Internet [51], [105].

Los desarrolladores de IaaS son capaces de crear una instancia del sistema operativo específico para ejecutar aplicaciones que se han desarrollado en su propio entorno. Esta es la principal diferencia entre IaaS y PaaS donde los desarrolladores sólo pueden centrarse en el desarrollo basado en la Web sin importar qué sistema operativo utilizan. Los desarrolladores pueden acceder al poder de computación masivo y las aplicaciones pueden ser desplegadas globalmente alcanzando un gran segmento de usuarios [51].

Los desarrolladores PaaS pueden utilizar la nube para almacenar su código y datos, y también pueden utilizar la nube como un canal para distribuir el software a los consumidores [105]. Los consumidores de PaaS facturan con el modelo de pago por uso, y les evita pagar enormes capitales iniciales y las inversiones en licencia de software, que a veces puede ser una barrera a la innovación. Los desarrolladores deben ser conscientes de lenguajes disponibles para plataformas específicas, así como el número de versión de los marcos de programación. También, un soporte a las bases de datos relacionales puede diferir entre los proveedores de PaaS. Los desarrolladores deben familiarizarse con las abstracciones que pueden ser específicas de la plataforma de programación [57].

PaaS puede proporcionar las abstracciones específicas para elementos tales como almacenamiento de datos y acceso, consultas de datos y colas de mensajes. Estas abstracciones no pueden portables a través de los proveedores de PaaS [57].

3.2.4.3. Software como Servicio (SaaS)

Software como servicio (SaaS, o algunas veces llamado software entregado en línea) es una alternativa para modelo de entrega de software tradicional llamado Software como un producto (SaaP) donde el cliente adquiere las licencias de software, instala las aplicaciones en sus locales y administra las configuraciones, parches y actualizaciones de versión [20], [24].

En el modelo SaaS, las aplicaciones están disponibles en forma de servicio basado en la red y los usuarios acceden a ellos vía Internet por un navegador Web. El cliente no compra una licencia de software, en cambio, contrata el uso de la aplicación que es desarrollada, organizada, gestionada y vendida por un proveedor de SaaS [105]. El cliente no tiene que instalar ni mantener ningún tipo de software [93].

El usuario de la aplicación SaaS no le importa dónde está alojada la aplicación, cual es el sistema operativo subyacente en que se ejecuta, o si la aplicación está escrita en .Net, Java o PHP. SaaS por lo general se considera como un servicio completamente gestionado [105]. El termino SaaS muy a menudo es referenciado como software en una nube y también es considerado como un modelo de servicio de Cloud Computing. Sin embargo, no todas las soluciones SaaS son soluciones en la nube [93].

SaaS es una manera económica para un cliente obtener acceso al software necesario mediante el uso de licencias sin la complejidad y posiblemente altos costos iniciales en comparación con modelos de entrega tradicionales [105]. Las características principales de SaaS son la libre disponibilidad a través de un navegador Web, así como las condiciones de pago en función del uso, y el mínimo de exigencias [24], [93], [98].

Algunos de los beneficios del modelo SaaS pueden ser identificados desde el punto de vista del cliente y el proveedor. Con la adquisición de software como un servicio en línea, el cliente puede concentrarse en sus funciones de competencias básicas. El software puede usarse más pronto, porque no se necesita ninguna instalación en el entorno del cliente.

SaaS libera al cliente desde hace mucho tiempo hacer contratos con el proveedor del servicio. Mediante el pago de cuotas mensuales, el cliente puede dejar de usar la aplicación en cualquier momento y cambiar a otro proveedor [24], [85]. No se requiere inversión para adquirir licencias y posiblemente nuevo hardware para ejecutar el software. Esto puede ser una verdadera oportunidad para las pequeñas empresas que necesariamente no son capaces de comprar SaaP debido a que no tienen la capacidad de sufragar costos iniciales. Los proveedores de software independientes (ISV) son capaces de acceder a nuevos mercados de clientes más pequeños, que anteriormente no fueron capaces de comprar el software [24], [85].

SaaS crea nuevos desafíos para los proveedores de software independientes, ofrece servicios de alojamiento y de clientes. Los desafíos técnicos incluyen el soporte a múltiples propietarios, el grado de personalización y gestión de los niveles de servicio [85]. El modelo SaaS no viene sin riesgos implicados. No todos los tipos de aplicaciones son convenientes para ofrecerse como un servicio. Sobre todo, los sistemas críticos comerciales pueden no ser muy óptimos para adquirirse como SaaS. La seguridad se hace una preocupación principal ya que los datos confidenciales se están almacenando en las premisas del proveedor.

La capacidad del cliente para personalizar las aplicaciones SaaS varían. Si se requiere una aplicación totalmente personalizable, puede ser un problema, las limitaciones del modelo SaaS y PaaS o IaaS necesitan ser consideradas [51]. Algunos proveedores de SaaS permiten ciertas opciones de personalización, por ejemplo, Salesforce.com ofrece CRM (*Customer Relationship Management*) y otros software de negocios en el modelo SaaS [98]. Otro ejemplo de SaaS es Google Apps que ofrece ciertas aplicaciones de office para el usuario [34].

3.2.5. Proveedores de Servicios Cloud

Los proveedores, como Microsoft, Google y Amazon han jugado un papel importante en la evolución de la nube de computación [96]. Existen numerosos servicios en la nube de incrementando el número de proveedores de servicios. En los capítulos siguientes, las se examinan brevemente las plataformas Windows Azure, Amazon EC2 y Google App Engine.

3.2.5.1. Windows Azure

La plataforma Windows Azure es un conjunto de tecnologías que proporciona un conjunto específico de servicios a desarrolladores de aplicaciones [17]. La plataforma Windows Azure ofrece una plataforma donde el ISV (proveedor independiente de software) y las empresas pueden hospedar sus aplicaciones y datos. Las aplicaciones pueden bridar servicios a empresas, consumidores o ambos. Las aplicaciones Windows Azure se ejecutan en los centros de datos de Microsoft y se acceden a través de Internet [18]. Las aplicaciones desarrolladas para Windows Azure pueden escalar mejor, ser más confiables y requieren menos administración que las aplicaciones desarrolladas utilizando el modelo de programación tradicional de Windows Server. Las aplicaciones pueden ser creadas, configuradas y monitoreadas vía un portal accesible desde el navegador. Azure proporciona un entorno que es interoperable y se basa en normas y protocolos como HTTP/HTTPS, REST (Representational State Transfer), SOAP, y XML. REST y SOAP permiten a los desarrolladores de interfaces entre herramientas y tecnologías Microsoft y de otras empresas [51]. Los clientes son facturados por el tiempo que utilicen los recursos computacionales, almacenamiento y ancho de banda [17].

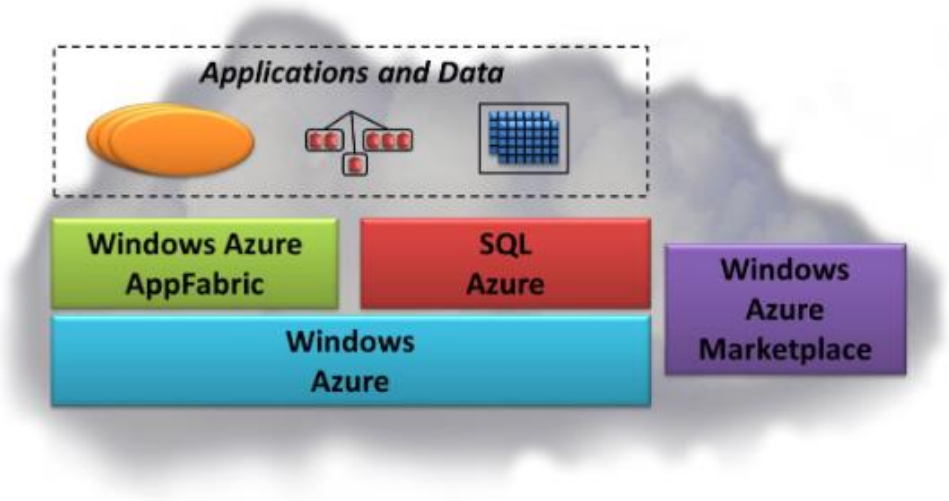

Figura 3-15. Principales componentes de la plataforma Windows Azure [13].

La figura 3-10 describe los principales componentes de la plataforma Azure. ***Windows Azure*** es un entorno Windows de ejecución de aplicaciones y almacenamiento de datos en computadoras que residen en centros de datos de Microsoft [17]. ***SQL Azure*** se basa en SQL server y ofrece servicios de datos relacionales en la nube. ***Windows Azure AppFabric*** es un conjunto de servicios de infraestructura que puede ser utilizada por aplicaciones que se ejecutan en la nube o local.

AppFabric proporciona a las aplicaciones medios para nombrar, descubrir, exponer, garantizar y coordinar los servicios Web, permitiendo a los desarrolladores concentrarse en su lógica de la aplicación en lugar de crear e implementar su propio servicios de infraestructura basado en la nube [17]. AppFabric incluye tres componentes: bus de servicio, control de acceso y almacenamiento en caché. Azure Marketplace es un servicio online desde donde las aplicaciones basadas en la nube se pueden comprar. Todos los componentes se ejecutan en centros de datos de Microsoft en el mundo. Los desarrolladores son capaces de controlar, el centro de datos que utiliza para ejecutar sus aplicaciones y almacenar sus datos [17].

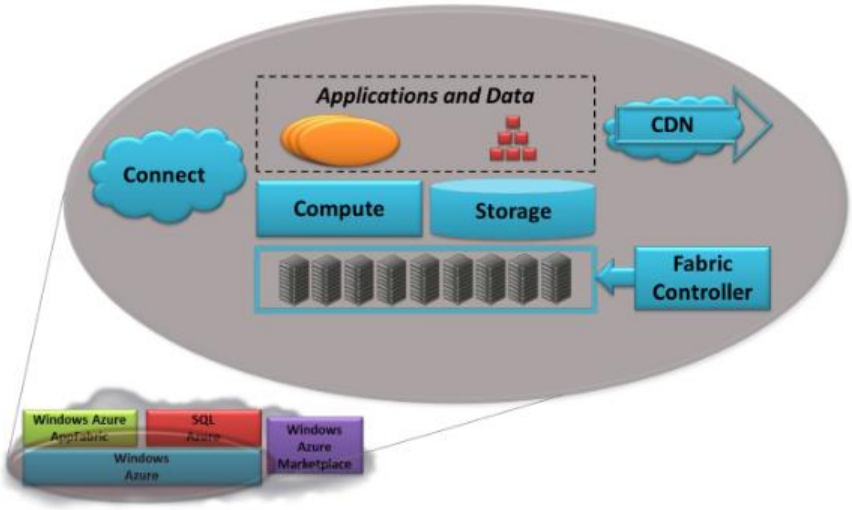

Figura 3-16. Principales componentes de Windows Azure [13].

Windows Azure y sus componentes se presentan en la figura 3-11. ***Compute*** ejecuta aplicaciones en la nube en un entorno Windows Server. Las aplicaciones pueden crearse con lenguajes .NET, C o Java. Los desarrolladores pueden utilizar Visual Studio u otras herramientas de desarrollo y son capaces de decidir si usar ASP.NET, WCF (Windows Communication Foundation) o PHP. ***Storage*** almacena datos binarios y estructurados en la nube. Los elementos de los datos en el almacenamiento de Windows Azure son blobs, colas y tablas. SQL Azure puede usarse si se requiere almacenamiento relacional tradicional. El servicio de almacenamiento de información puede ser usado por las aplicaciones Azure y aplicaciones locales utilizando REST (Representational State Transfer) [17], [18].

El propósito del ***fabric controller*** es implementar, administrar y monitorear aplicaciones, y también gestionar las actualizaciones de software del sistema a lo largo de la plataforma. ***Content Delivery Network*** existe para acelerar el acceso global a los datos binarios almacenados en almacenamiento de Windows Azure manteniendo copias en caché de datos. Esto puede hacerse para blobs. ***Conect*** módulo que permite a las organizaciones interactuar con las aplicaciones de la nube como si fueran dentro del firewall propio de las organizaciones. Permite conexiones de nivel de IP entre máquinas locales y aplicaciones Azure. Por ejemplo, una aplicación Azure puede acceder a las bases de datos situadas en locales [17], [18].

3.2.5.2. Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud proporciona a sus usuarios capacidad redimensionable de computación en la nube. Su objetivo es hacer que la computación a escala web sea más fácil para los desarrolladores [3]. El usuario tiene el control de ampliar la capacidad en función de sus necesidades y el usuario paga sólo por la capacidad que se utiliza. Según Amazon, *"los desarrolladores cuentan con herramientas para construir aplicaciones resistentes a fallos y escenarios aisladores comunes de fallos"* [3]. EC2 es un auténtico entorno informático virtual. Mediante interfaces de servicio Web, un usuario puede iniciar instancias de diversos sistemas operativos, los carga con entornos de aplicaciones personalizadas y administra los permisos de acceso a la red [3]. Un usuario puede seleccionar una imagen pre-configurada a partir de plantillas, o el usuario puede crear una AMI (Amazon Machine Image) que contiene las aplicaciones deseadas, bibliotecas, datos y parámetros de configuración asociados. La seguridad y el acceso de la red para EC2 pueden configurados. Un usuario puede iniciar cualquier número de instancias y mediante la API del servicio Web y herramientas de gestión provistas para que las instancias puedan ser controladas. La siguiente lista contiene los beneficios de Amazon EC2 [3].

- ***Elastic***: Amazon EC2 permite aumentar o reducir la capacidad en cuestión de minutos, sin esperar horas ni días. Puede enviar una, cientos o incluso miles de instancias del servidor simultáneamente [3].

Desde luego, como todo esta operación se controla con API de servicio Web, la aplicación se escalará (aumentará o disminuirá su capacidad) dependiendo de sus necesidades.

- *Totalmente controlado:* Tendrá control total sobre sus instancias. Tiene acceso de usuario raíz a todas ellas, y puede interactuar con ellas como con cualquier otra máquina. Puede detener su instancia y mantener los datos en su partición de arranque, para reiniciar a continuación la misma instancia a través de las API del servicio web. Las instancias se pueden reiniciar de forma remota mediante las API del servicio web. Asimismo, tiene acceso a la emisión de consola de sus instancias [3].
- *Flexible:* Tendrá la posibilidad de elegir entre varios tipos de instancia, sistemas operativos y paquetes de software. Amazon EC2 permite seleccionar una configuración de memoria, CPU y almacenamiento de instancias, así como el tamaño de la partición de arranque óptimo para su sistema operativo y su aplicación. Por ejemplo, entre sus opciones de sistemas operativos se incluyen varias distribuciones de Linux y Microsoft Windows Server [3].
- *Diseño pensado para su uso con otros Amazon Web Services:* Amazon EC2 trabaja con Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon SimpleDB y Amazon Simple Queue Service (Amazon SQS) para proporcionar una solución informática completa, procesamiento de consultas y almacenamiento en una gran gama de aplicaciones [3].
- *Fiable*: Amazon EC2 ofrece un entorno muy fiable en el que las instancias de sustitución se pueden enviar con rapidez y anticipación. El servicio se ejecuta en los centros de datos y la infraestructura de red acreditados de Amazon. El compromiso del contrato a nivel de servicio de Amazon EC2 es de una disponibilidad del 99,95% en cada Región de Amazon EC2 [3].
- *Seguro:* Amazon EC2 funciona junto con Amazon VPC para proporcionar una funcionalidad de la red sólida y segura para sus recursos informáticos. Sus instancias informáticas se ubican en una Virtual Private Cloud (VPC) con el rango de IP que especifique. Usted decide las instancias que se exponen en Internet y las que permanecen privadas [3].
- *Económico:* Amazon EC2 permite disfrutar de las ventajas financieras de Amazon. Pagará una tarifa muy baja por la capacidad informática que realmente utiliza. Consulte las Opciones de compra de instancias de Amazon EC2 para obtener una descripción más detallada [3].
- *On-Demand Instances:* Con On-Demand Instances puede pagar por la capacidad informática por hora, sin compromisos a largo plazo. Esto le liberará de los costes y las complejidades de la planificación, la compra y el mantenimiento del hardware y transformará lo que normalmente son grandes costes fijos en costes variables mucho más reducidos.

Gracias a On-Demand Instances también se elimina la necesidad de comprar una "red de seguridad" de capacidad para gestionar picos de tráfico periódicos [3].

- *Instancias reservadas*: Las instancias reservadas ofrecen la opción de realizar un pago puntual reducido por cada instancia que desee reservar y recibir a cambio un descuento importante en el cargo de uso por horas de dicha instancia. Existen tres tipos de instancias reservadas (instancias reservadas de utilización ligera, media e intensa) que permiten equilibrar el importe del pago anticipado a realizar con su precio por hora efectivo [3].
- *Instancias puntuales*: Con las instancias puntuales, los clientes pueden ofertar la capacidad sin utilizar de Amazon EC2 y ejecutar dichas instancias mientras su oferta supere el precio puntual. El precio puntual cambia periódicamente según la oferta y la demanda, y los clientes cuyas ofertas alcancen o superen dicho precio tendrán acceso a las instancias puntuales disponibles [3].
- *Fácil de empezar*: Puede empezar a utilizar Amazon EC2 con rapidez; para ello, visite AWS Marketplace, donde podrá seleccionar el software pre-configurado en las imágenes de máquina de Amazon (AMI). Puede implementar este software con rapidez en EC2 gracias al lanzamiento de 1-Click o con la consola de EC2 [3].

3.2.5.3. Google App Engine

Google App Engine permite a los usuarios ejecutar aplicaciones Web en la infraestructura de Google. Según Google, *"las aplicaciones de App Engine son fáciles de construir, fáciles de mantener y fácil de escalar de acuerdo al tráfico y almacenamiento de datos que se necesite para crecer"* [33]. No hay ninguna necesidad de mantener servidores o preocuparse por hardware, parches o copias de seguridad al usar Google App Engine. El usuario sólo tiene que cargar aplicaciones en la plataforma. Las aplicaciones pueden ser compartidas con el mundo, o el acceso puede ser limitado a determinados grupos de usuarios [33].

Google App Engine tiene entornos de ejecución Java y Python, y pueden construir aplicaciones con varios lenguajes de programación. Los entornos de ejecución están construidos para asegurar que las aplicaciones se ejecutan de forma rápida, segura y sin interferencia de otras aplicaciones en el sistema. Los entornos proporcionan protocolos y tecnologías comunes para el desarrollo de aplicaciones Web. El método de fijación de precios es el pago por uso, sin costes de instalación ni cargos recurrentes [33].

El almacenamiento y ancho de banda se miden por gigabyte, y la cantidad de recursos que consuman las aplicaciones se pueden controlar. Todas las aplicaciones pueden utilizar 500 MB de almacenamiento y CPU y ancho de banda que permite 5 millones de páginas vistas en un mes gratis. Cuando se necesita más espacio de almacenamiento de la CPU, el usuario paga sólo el monto que exceda los niveles libres. Una consola de administración basada en la Web se proporciona para que el usuario sea capaz de gestionar sus aplicaciones en ejecución [33].

Google App Engine incluye las siguientes características [33]:

- Servidor Web dinámico, totalmente compatible con las tecnologías web más comunes.
- Almacenamiento permanente con funciones de consulta, clasificación y transacciones.
- Escalado automático y distribución de carga.
- API para autenticar usuarios y enviar correo electrónico a través de Google Accounts.
- Un completo entorno de desarrollo local que simula Google App Engine en tu equipo.
- Colas de tareas que realizan trabajos fuera del ámbito de una solicitud web.
- Tareas programadas para activar eventos en momentos determinados y en intervalos regulares.

Las aplicaciones se ejecutan en un entorno seguro que proporciona acceso limitado al sistema operativo subyacente. Estas limitaciones permiten a App Engine distribuir solicitudes Web de la aplicación en varios servidores e iniciar y detener los servidores según las demandas del tráfico. La zona de pruebas aísla la aplicación en su propio entorno seguro de confianza, totalmente independiente del hardware, del sistema operativo y de la ubicación física del servidor web [33].

Algunos ejemplos de las limitaciones del entorno seguro de la zona de pruebas son:

- Una aplicación solo podrá acceder a otros equipos de Internet a través de los servicios de correo electrónico y extracción de URL proporcionados. Otros equipos solo se podrán conectar a la aplicación mediante solicitudes HTTP (o HTTPS) en los puertos estándar [33].
- Una aplicación no podrá escribir en el sistema de archivos. Una aplicación podrá leer archivos, pero solo aquellos subidos con el código de la aplicación. La aplicación deberá utilizar el almacén de datos de App Engine, Memcache u otros servicios para todos los datos que permanezcan entre las solicitudes [33].
- El código de aplicación solo se ejecuta en respuesta a una solicitud web, a una tarea en cola o a una tarea programada y debe devolver datos de respuesta en un periodo de 30 segundos en cualquier caso. Un controlador de solicitudes no podrá generar un subproceso ni ejecutar código después de haber enviado la respuesta [33].

3.3. Ingeniería basada en Modelos (MDE)

La ingeniería basada en modelos (MDE) es una disciplina de la ingeniería del software que se basa en modelos como artefactos de primera clase y que tiene como objetivo desarrollar, mantener y evolucionar software por medio de transformaciones de modelo. El proceso de desarrollo bajo MDE es llamado Desarrollo de Software Dirigido por Modelos (DSDM).

MDE ofrece un enfoque más efectivo; los modelos son partes activas del proceso de desarrollo de software. Los modelos son abstractos y formales, al mismo tiempo. La abstracción no destaca aquí por su vaguedad, pero para la esencia de la compacidad. Los modelos MDE tienen el significado exacto del código del programa, en el sentido de que la mayor parte de la aplicación final, no sólo la clase y los esqueletos de los métodos pueden ser generados a partir de ellos [106]. En este caso los modelos ya no son sólo la documentación, sino partes del software, lo que constituye un factor decisivo para aumentar la velocidad y calidad de desarrollo de software.

Un enfoque basado en modelos requiere lenguajes para la especificación de modelos, definición de transformación y descripción del metamodelo. MDE propone el uso de transformaciones de modelos con el fin de transformar un modelo en otro, y también para producir el producto final.

MDE es aceptado por diversas organizaciones y empresas que incluyen a la OMG, IBM y Microsoft.

Existen cinco cosas que una infraestructura de soporte MDE debe definir:

- Conceptos disponibles para la creación de modelos y reglas claras que rigen su uso.
- La notación a utilizar para describir los modelos.
- Tiene que ser claro, cómo los elementos del modelo representan los elementos del mundo real y los artefactos de software.
- Conceptos para facilitar las extensiones de usuarios dinámicos para modelar conceptos, modelos de notación y los modelos creados a partir de ellos.
- Conceptos para facilitar el intercambio de conceptos de modelos y notación, y los modelos creados a partir de conceptos definidos por el usuario para facilitar las asignaciones de los modelos a otros artefactos.

3.3.1. Desarrollo de Software Dirigido por Modelos (DSDM)

El desarrollo de software dirigido por modelos (DSDM) es un enfoque de desarrollo de software basado en modelos. Un modelo puede definir la funcionalidad, estructura o comportamiento de un sistema. Los modelos permiten trabajar con un nivel de abstracción más cerca a los conceptos de dominio, en lugar de centrarse en conceptos orientados a la plataforma como el desarrollo del software tradicional. El objetivo de esta propuesta es maximizar la productividad, incrementar la interoperabilidad entre sistemas, facilitando la reutilización y adaptación del cambio tecnológico.

Además cualquier artefacto software es considerado un *modelo* o un elemento del modelo. Mientras que los enfoques anteriores utilizan modelos para documentación o la comunicación de ideas, que son entidades de primera clase durante todo el ciclo de vida ingeniería todo basado en modelos [11]. DSDM es una aproximación abierta e integradora no vinculada a una norma especial, por lo tanto, existen diferentes implementaciones.

El DSDM intenta capturar lo que con frecuencia se expresa de manera informal como especificaciones formales basadas en modelos. Los conceptos clave para mitigar el problema actual de la ingeniería de software son *modelos*, *metamodelos*, *espacios tecnológicos*, *lenguajes de modelado de dominio específico* y diferentes *tipos de transformaciones de modelo* como modelo a modelo o modelo a texto.

3.3.2. Model-Driven Architecture (MDA)

Model-Driven Architecture (MDA), y como se ha dicho antes, está incluido en la definición de MDE. El estándar MDA desde el *Object Management Group* (OMG) es una encarnación específica de la ingeniería dirigida por modelos (MDE).

MDA es un joven modelo establecido por la OMG. La OMG se fundó en el año 1989 y es actualmente un consorcio abierto de aproximadamente 800 empresas en el mundo. La OMG crea especificaciones independientes de fabricantes para mejorar la interoperabilidad y portabilidad de sistemas de software.

MDA está relacionado al uso de lenguajes de modelado como lenguajes de programación más que meramente como lenguajes de diseño. La programación con lenguajes de modelado puede mejorar la calidad y la velocidad de desarrollo del software.

El objetivo de MDA es separar la forma en que los sistemas de aplicación se definen a partir de la tecnología en la que se ejecutan.

MDA inicia con la idea bien conocida y establecida desde hace mucho tiempo de separar las especificaciones operacionales del sistema de los detalles de la forma en que los sistema utilizan las capacidades de su plataforma.

La independencia de la plataforma software es análoga a la independencia de la plataforma hardware. Un lenguaje independiente de la plataforma hardware, tales como C o Java, permite la escritura de una especificación que se puede ejecutar en una amplia variedad de plataformas de hardware con ningún cambio. De la misma forma que un lenguaje software independiente de la plataforma permite la escritura de una especificación que puede ejecutarse en una amplia variedad de plataformas de software, o los diseños de la arquitectura software, sin tener que realizar ningún cambio. Por lo tanto, un software independiente de la plataforma podría ser asignado a un multiprocesador o entorno multitarea CORBA, o un ambiente cliente-servidor de base de datos relacional, sin cambio en el modelo.

En general, la organización y procesamiento de datos implicada un modelo conceptual que puede no ser igual a la organización de los datos y el procesamiento de la implementación. Si se consideran dos conceptos, los del "cliente" y "cuenta", modelarlos como clases, utilizando UML se sugiere que la solución de software debe expresarse en términos de clases de software llamados Cliente y Cuenta. Sin embargo, hay muchos diseños de software posibles que pueden cumplir con estos requisitos, muchas de las cuales ni siquiera están orientados a objetos.

Entre el concepto y la aplicación, un atributo puede ser un referente, una clase puede dividirse en conjuntos de instancias de objetos de acuerdo con algunos criterios de clasificación, las clases se pueden combinar o dividir; el estado de gráficos puede ser aplanado, combinado, o separado, y así sucesivamente. Un lenguaje de modelado que permite este tipo de asignaciones es un software independiente de la plataforma.

Elevando el nivel de abstracción cambia la plataforma de la cual depende cada capa de abstracciones. El desarrollo basado en modelos se basa en la construcción de modelos que son independientes de sus plataformas de software, que incluyen entornos cliente servidor de base de datos relacional y la estructura de código final [60].

MDA proporciona un enfoque, y permite que las herramientas sean provistas para:

- La especificación de un sistema independiente de la plataforma que lo soporta.
- La especificación de las plataformas.
- La elección de una plataforma en particular para el sistema.
- Y finalmente, la transformación de la especificación del sistema en una particular plataforma.

Los tres objetivos principales de MDA son portabilidad, interoperabilidad y reutilización mediante separación arquitectónica de diversos puntos de vista.

El ciclo de vida de desarrollo de MDA, no parece muy diferente del ciclo de vida tradicional. Se identifican las mismas fases. Una de las principales diferencias se encuentra en la naturaleza de los artefactos que se crean durante el proceso de desarrollo. Los artefactos son modelos formales, es decir, los modelos que pueden ser entendidos por las computadoras [48].

Como conclusión MDA es una aproximación al desarrollo de sistemas, que aumenta la potencia de los modelos en ese trabajo. Está basado en modelos porque proporciona un medio para que el uso de modelos para dirigir el curso del entendimiento, diseño, construcción, implementación, operación, mantenimiento y modificación.

3.3.2.1. Puntos de Vistas MDA

Un punto de vista sobre un sistema es una técnica para la abstracción utilizando un conjunto seleccionado de conceptos arquitectónicos y reglas de estructuración, con el fin de centrarse en preocupaciones particulares dentro de ese sistema. La arquitectura basada en modelos especifica tres puntos de vista sobre un sistema, un punto de vista independiente de cómputo, un punto de vista independiente de plataforma y un punto de vista específico de plataforma [70].

Punto de Vista Independiente de Cómputo

El punto de vista independiente de cómputo se centra en el entorno del sistema y los requisitos para el sistema; los detalles de la estructura y procesamiento están ocultos o aún indeterminados.

Punto de Vista Independiente de Plataforma

El punto de vista independiente de plataforma se centra en el funcionamiento de un sistema al tiempo que oculta los detalles necesarios para una determinada plataforma. Una vista independiente de plataforma muestra parte de la especificación completa que no cambia de una plataforma a otra. Una vista independiente de la plataforma puede utilizar un lenguaje de modelado de propósito general o un lenguaje específico para el área en la cual se utilizará el sistema.

Punto de Vista Específico de Plataforma

El punto de vista específico de plataforma combina el punto de vista independiente de plataforma con un enfoque adicional en el detalle del uso de una plataforma específica por un sistema.

3.3.2.2. Modelos en MDA

Los artefactos de primera clase en la ingeniería basada en modelos son los modelos. La primera cosa a hacer es definir que es un modelo. Un modelo es una simplificación (o una descripción abstracta) de una parte del mundo llamado sistema, construido con un objetivo previsto en la mente. Un modelo debería ser más fácil de usar y entender que el sistema original, debe ser capaz de responder a preguntas sobre el sistema. La respuesta proporcionada por el modelo debe ser exactamente las mismas que las dadas por el mismo sistema [100], [12].

Los modelos pueden consistir en un conjunto de elementos con una representación gráfica o textual. Mientras que esto sirve como un punto de partida, Kleppe et al. [48] da una definición aún más dirigida al DSDM. Un modelo es una descripción de un (parte de) sistema escrito en un lenguaje bien definido. Un lenguaje bien definido es un lenguaje con una forma bien definida (sintaxis), y significado (semántica), lo que es conveniente para la interpretación automatizada por un computador.

La idea de MDA es la creación de diferentes modelos de un sistema en diferentes niveles de abstracción. Cada modelo representa un aspecto determinado del sistema.

De acuerdo con estas definiciones, el código fuente es un modelo también. El código fuente es una representación simplificada de la estructura inferior de la máquina y las operaciones que son necesarias para automatizar las tareas en el mundo real. Por otra parte, el correcto código fuente es un modelo muy útil, ya que le dice a la máquina qué le indica a la maquina las medidas necesarias para mantener el objetivo del sistema.

Tipos de modelos

El estándar MDA [70] define cuatro tipos de modelos en el ciclo de vida del desarrollo de software dirigido por modelos:

Modelo Independiente de Cómputo (CIM)

Un modelo independiente de cómputo es una vista del punto de vista independiente de cómputo, que se centra en el entorno y en los requisitos del sistema; los detalles estructurales o de procesamiento del sistema son ocultados o todavía no son determinados. Un CIM no muestra los detalles de la estructura del sistema. Un CIM algunas veces es llamado un modelo de dominio y un vocabulario que es familiar para los profesionales del dominio en cuestión y utilizan en su especificación [70]. Se centra en el entorno del sistema y en los requisitos que el usuario tiene en el sistema, la descripción proporciona lo que se espera que el sistema debe hacer.

Modelo Independiente de Plataforma (PIM)

El modelo independiente de plataforma se centra en el funcionamiento de un sistema al mismo tiempo ocultando los detalles necesarios para una determinada plataforma. Un PIM exhibe un determinado grado de independencia de la plataforma con el fin de ser aptos para su uso con un número de diferentes plataformas de tipo similar. También es una representación de la funcionalidad y comportamiento del negocio, sin distorsión por los detalles de la tecnología. Además, muestra la parte de la especificación completa que no cambia desde una plataforma a otra.

El objetivo es posponer el proceso de desarrollo de creación de modelos que tengan en cuenta los aspectos tecnológicos de una plataforma tanto como posible. La principal ventaja es ser capaz de reaccionar eficientemente y con bajos costos a los cambios de tecnología.

Modelo Específico de Plataforma (PSM)

Un modelo específico de plataforma es una combinación de un PIM con el detalle adicional de la plataforma específica del sistema.

Modelo de Plataforma

Finalmente los modelos de plataforma son la representación de conceptos técnicos de partes de la plataforma, los servicios prestados por esa plataforma y para su uso en un posterior PSM, los conceptos que modela el uso de la plataforma por las aplicaciones.

3.3.2.3. Metamodelos

La palabra “meta” es griega y significa “arriba”, por lo tanto el término metamodelo puede interpretarse como un modelo que describe otro modelo. Un lenguaje consiste en palabras cuya combinación es restringida por una gramática. Si una sentencia en un lenguaje es vista como un posible modelo, la definición de su estructura, la gramática puede ser vista como su metamodelo. Anteriormente se dijo que en el DSDM un metamodelo define como un modelo puede ser visto, esto puede ser formulado de manera más precisa como: un metamodelo define los constructores y las reglas que se utilizan para crear una clase de modelos.

Esto es consistente con la siguiente definición:

"Un metamodelo es un modelo de un conjunto de modelos [66]."

"Un metamodelo es un modelo que define el lenguaje para expresar un modelo [95]."

Un metamodelo es un modelo de especificación que describe sus modelos en un cierto lenguaje de modelado. Un metamodelo dice lo que se puede expresar en un modelo válido del lenguaje de modelado.

La interpretación de un metamodelo es el mapeo de elementos del metamodelo a elementos del lenguaje de modelado. El valor de verdad de las declaraciones en el metamodelo puede determinarse por cualquier modelo expresado en el lenguaje de modelado. Puesto que el metamodelo es la especificación de modelos, un modelo en el lenguaje de modelado es válido sólo si ninguna de estas afirmaciones son falsas. Los metamodelos precisos son un requisito previo para la realización de transformación de modelos automáticas y para definir modelos precisos.

Puesto que un metamodelo es también un modelo, podría ser expresado en algunos lenguajes de modelado. Un metamodelo para un lenguaje de modelado podría utilizar el mismo lenguaje de modelado. Las afirmaciones en el metamodelo se expresan en el mismo lenguaje que está siendo descrito por el metamodelo. Esto es llamado metamodelo reflexivo.

3.3.2.4. Meta-Object Facility (MOF)

Meta-Object Facility (MOF) es un estándar propuesto y definido por la OMG [82] para soportar a MDA. Esta norma propone cuatro niveles de arquitectura metamodelos con cuatro meta capas como se muestra en la figura 3-17. Cada meta capa es el metamodelo de los constructores en las capas anteriores. Las capas se describen a continuación:

1. Capa M3 Metametamodelo: En esta capa reside el metametamodelo, un lenguaje para definir los metamodelos del nivel M2. En el estándar de OMG, MOF es el lenguaje definido en esta capa.
2. Capa M2 Metamodelo: Los metamodelos M2 se utilizan para describir los modelos M1. El metamodelo de UML de la OMG describirá los constructores de UML.
3. Capa M1 Modelo: En este nivel es donde se definen los modelos. Un modelo es una instancia de un metamodelo M2. Es decir, si en la capa M2 reside el metamodelo de UML, en M1 podríamos tener uno de sus modelos: diagrama de clases, diagrama de actividad, diagrama de secuencia, y así sucesivamente.
4. Capa M0 Instancia: En esta capa se definen objetos del mundo real.

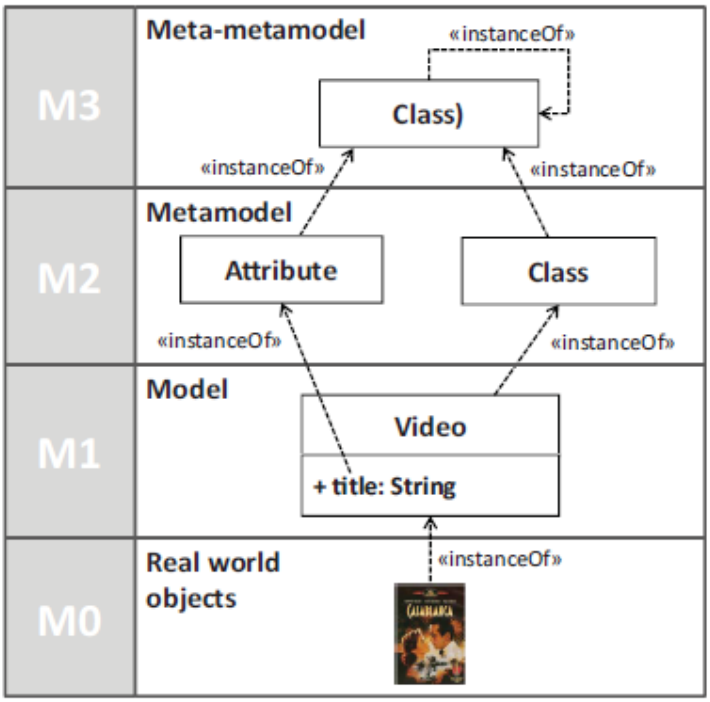

Figura 3-17. Capas de Arquitectura MOF [13].

MOF es una arquitectura de metamodelado cerrada. Esto significa que el nivel M3 podría definirse con instancias de elementos de M3. Esto implica que podemos definir MOF.

Además MOF proporciona conceptos para definir un lenguaje:

- Clases, que son metaobjetos del modelo MOF.
- Tipos de datos (propiedad), que son necesarios para el modelo de datos descriptivos (es decir, los tipos primitivos).
- Asociaciones (propiedad), que son las relaciones binarias entre los metaobjectos del modelo.
- Paquetes, que son modularizan los modelos.

3.3.2.5. Transformación de Modelos

Transformación de modelos: es el proceso de conversión de un modelo a otro modelo del mismo sistema [70]. Un modelo puede ser transformado a varios modelos alternativos que pueden mantener la semántica, pero con diferente sintaxis.

Las asignaciones y las relaciones se definen como las especializaciones de transformaciones. La figura 3-13 muestra los dos diferentes niveles en los que la transformación es definida (nivel metamodelo) y ejecutados (modelo).

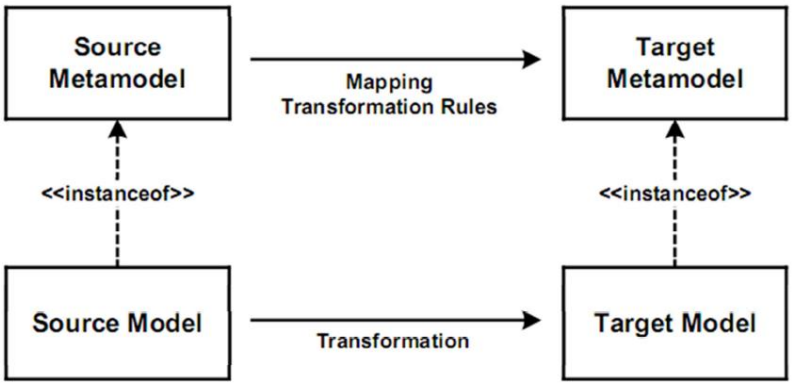

Figura 3-18. Definición de una transformación de modelos.

Una definición de transformación consiste en una colección de reglas de transformación que son especificaciones inequívocas de la manera que (parte de) un modelo puede utilizarse para crear una pieza de otro modelo. Las transformaciones son definidas en términos de los metamodelos involucrados en el proceso de transformación.

Una transformación, una definición de transformación y sus reglas de transformación pueden definirse de la siguiente manera [40]:

- Una transformación es la generación automática de un modelo destino desde un modelo de fuente, según una definición de transformación.
- Una definición de transformación es un conjunto de reglas de transformación que juntas describen cómo un modelo origen puede ser transformado en un modelo destino.
- Una regla de transformación es una descripción de cómo una o más constructores en la lenguaje origen pueden ser transformados en uno o más constructores de un lenguaje destino.

La característica más importante de una transformación es el hecho de que una transformación de modelos debe mantener el significado entre el modelo origen y el modelo destino. En este punto hay que decir que el significado del modelo sólo puede ser preservado y expresado en el modelo origen y destino. Parte de la información pueden perderse si el lenguaje de destino es menos expresivo que el lenguaje de origen.

Un mapeo se define como una transformación unidireccional en contraste a una relación que define una transformación bidireccional.

3.3.2.6. Lenguaje de Restricción de Objetos (OCL)

Lenguaje de restricciones de objetos (OCL) [81] es un lenguaje formal usado para describir las expresiones de modelos UML. Estas expresiones suelen especificar condiciones invariables que se deben mantener en el modelado del sistema o consultas sobre los objetos descritos en un modelo. Tenga en cuenta que cuando OCL evalúa las expresiones, que no tienen efectos secundarios; es decir, su evaluación no puede alterar el estado de ejecución del correspondiente sistema.

Las expresiones OCL pueden ser usadas para especificar operaciones/acciones que, cuando son ejecutadas, no alteraran el estado del sistema. Los modeladores UML pueden utilizar OCL para especificar las restricciones específicas de la aplicación en sus modelos. Los modeladores UML pueden también utilizar OCL para especificar las preguntas sobre el modelo de UML, que son totalmente lenguaje de programación independiente.

OCL no sólo se puede aplicar en modelos UML, pero también puede ser aplicado en UML o metamodelos MOF ya que están expresados en UML o un subconjunto de UML.

Por lo tanto OCL puede ser utilizado para restringir la semántica del metamodelo, por ejemplo, por medio de estereotipos o lenguajes de dominio específicos (LDE).

En el contexto MDA, OCL se puede utilizar de tres formas:

- Modelos precisos en M1 a nivel MOF.
- Definición de lenguajes de modelado.
- Definición de transformaciones.

3.3.2.7. ATL (Atlas Transformation Language)

Es un lenguaje de dominio específico para la especificación de transformaciones de un modelo a otro [7], [46], [47] . ATL está inspirado en los requerimientos del estándar QVT [79] de la OMG, y se basa en el formalismo de OCL [81].

La decisión de utilizar OCL está motivada por su amplia adopción en MDE y el hecho de que es un lenguaje estándar apoyado por OMG y de las principales herramientas de los proveedores.

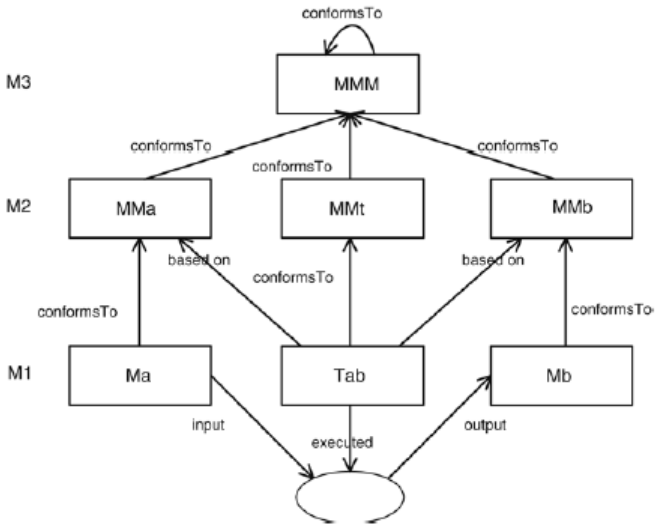

Figura 3-19. Patrón de transformación de modelos.

ATL es aplicado en el contexto del patrón de transformación que se muestra en la figura 3-19. Este patrón es un modelo origen *Ma* que se transforma en un modelo destino *Mb* según una definición de transformación *mma2mmb.atl* escrita en el lenguaje ATL. La definición de transformación es un modelo conforme al metamodelo de ATL [46]. Todos los metamodelos son conformes a MOF.

ATL es un lenguaje híbrido, es decir, que ofrece una mezcla de construcciones declarativas e imperativas. El estilo declarativo de especificación de transformación tiene un número de ventajas. Generalmente está basado en la especificación de las relaciones entre los patrones origen y destino, y por lo tanto tiende a estar más cerca de la forma en que los desarrolladores perciben intuitivamente una transformación [46].

Las transformaciones ATL son unidireccionales, es decir, operan en modelos origen sólo para lectura y en modelos de destino para escritura. Durante la ejecución de una transformación, el modelo origen puede ser navegado, pero los cambios en él no están permitidos, y en un modelo destino es imposible ser navegado [46]. Una transformación bidireccional se implementa como un par de transformaciones: una para cada dirección. Los modelos origen y destino de ATL pueden expresarse en el formato de serialización XMI de la OMG. Los metamodelos origen y destino también pueden expresarse en XMI o en la notación más conveniente [46].

Las definiciones de transformación en ATL se realizan en módulos (*module*). Un módulo contiene una sección obligatoria de cabecera (*header*), una sección de importación (*import*), y un número de ayudantes (*helpers*), y reglas de transformación (transformation rules). La sección de cabecera da el nombre al módulo de transformación y declara los modelos origen y destino. A continuación se detalla un ejemplo de la sección de cabecera:

```
module Class2Relational;
create OUT : Relational from IN : Class;
```

La sección ***header*** comienza con la palabra clave *module* seguida por el nombre del módulo. Luego, los modelos origen y destino se declaran como variables introducidas por sus metamodelos [46]. La palabra clave *create* indica los modelos de destino. La palabra clave *from* indica los modelos origen. En este ejemplo, el modelo destino junto a la variable *OUT* es creado del modelo origen *IN*. Los modelos origen y destino se ajustan a los metamodelos *Class* y *Relational* respectivamente. En general, más de un modelo origen y destino pueden ser enumerados en la sección de cabecera [46].

Un ***helper*** define las operaciones en el contexto de los elementos de un modelo o en el contexto de un módulo. Ellos pueden tener parámetros de entrada y pueden utilizar la recursividad. Un *helper attribute* se utiliza para asociar valores de sólo lectura nombrados en los elementos de un modelo. De igual forma las operaciones tienen un nombre, un contexto y un tipo. La diferencia es que ellas no pueden tener parámetros de entrada. Sus valores se especifican mediante una expresión OCL. Para ilustrar la sintaxis de los *helpers* se presenta el siguiente ejemplo [46]:

```
helper context String def: firstToLower() : String =
self.substring(1, 1).toLower() + self.substring(2, self.size());
```

Las reglas de transformación es la construcción básica en ATL que se utiliza para expresar la lógica de transformación. Las reglas ATL pueden especificarse en un estilo declarativo o en un estilo imperativo [46].

Matched Rules: Las *matched rules* son reglas ATL declarativas. Una matched rule se compone de un patrón de origen y un patrón de destino. La regla del patrón de origen regla especifica un conjunto de tipos de destino (procedentes de los metamodelos de origen y desde el conjunto de tipos disponibles de colecciones en OCL) y un *guard* (una expresión OCL booleana). Un patrón de origen se evalúa a un conjunto de coincidencias en el modelo origen [46].

El patrón de destino se compone de un conjunto de elementos. Cada elemento especifica un tipo de destino (del metamodelo destino) y un conjunto de enlaces. Un enlace se refiere a una característica del tipo de destino (es decir, un atributo, una referencia, o un extremo de la asociación) y especifica una expresión de inicialización para el valor de característica [46]. El siguiente fragmento de código muestra una simple *matched rule*. Esta regla implementa la lógica para la transformación de las clases a las tablas.

```
rule Class2Table {
 from
     c : Class!Class
 to
     out : Relational !Table (
         name <- c.name,
         col <- Sequence {key}->union(c.attr->select(e | not e
         .multiValued)),
         key <- Set {key}
     ),
     key : Relational!Column (
         name <- 'objectId',
         type <- thisModule.objectIdType
     )
}
```

Tipos de *mathed rules*: Hay varios tipos de *matched rules* que difieren en la forma en que se desencadenan [46].

- *Standard rule* son aplicadas una vez para todas las combinaciones que se puedan encontrar en los modelos origen [46].
- *Lazy rule* son provocados por otras reglas. Se aplican en una sola combinación tantas veces como es referida por otras reglas, cada vez que produzca un nuevo conjunto de elementos de destino [46].
- *Unique lazy rule* también son provocados por otras reglas. Se aplican una sola vez por una combinación determinada. Si una *unique lazy rule* se activa más tarde en la misma combinación, esta utiliza los elementos de destino ya creados [46].

Herencia de reglas. En ATL, la herencia de reglas se puede utilizar como un mecanismo de reutilización de código, y también como un mecanismo para especificar reglas polimórficas [46].

Una regla (llamada subregla) puede heredar de otra regla (regla padre). Una subregla coincide con un subconjunto de los combinaciones de la regla padre. Los tipos patrón de origen de la regla padre podrán ser sustituidos por sus subtipos en el patrón de origen de la subregla [46].

ATL tiene una parte imperativa en base a dos conceptos principales:

- *Called rules*: Una *called rule* es básicamente un procedimiento: se invoca por su nombre y puede tomar argumentos [46].
- *Action block*: Un *action block* es una secuencia de sentencias imperativas, y se puede utilizar en lugar de o en una combinación con un patrón de destino en combinación o en *called rules* [46].

Las declaraciones imperativas disponibles en ATL son las construcciones conocidas para la especificación de un flujo de control, tales como las condiciones, bucles, asignaciones, etc [46].

3.3.2.8. Transformaciones de Modelo a Texto

MDA coloca el modelado en el centro del proceso de desarrollo de software. Varios modelos son utilizados para capturar diversos aspectos del sistema de una manera independiente de la plataforma [80]. Luego, se aplican conjuntos de transformaciones a estos modelos independientes de plataforma (PIM) para derivar modelos específicos de la plataforma (PSM). Estos PSMs necesitan ser finalmente transformados en artefactos de software como código, especificaciones de implementación, informes, documentos, etc [80].

El estándar *MOF model to text* (*mof2text*) aborda cómo traducir un modelo a varios artefactos de texto como código, especificaciones de implementación, informes, documentos, etc [80]. En esencia, el estándar *mof2text* debe abordar la manera de transformar un modelo en una representación de texto. Una forma intuitiva de abordar este requerimiento es un enfoque basado en plantillas (*templates*) en la que el texto que se generen a partir de los modelos se especifica como un conjunto de plantillas de texto que se configuran con los elementos del modelo [80].

Un enfoque basado en plantillas es usado donde un *template* específica una plantilla de texto con marcadores de posición para los datos que se extraen de los modelos. Estos marcadores de posición son esencialmente expresiones especificadas sobre las entidades del metamodelo con consultas, siendo los principales mecanismos para la selección y la extracción de los valores de los modelos. Estos valores se convierten en fragmentos de texto usando un lenguaje de expresión aumentado con una biblioteca de manipulación de cadenas [80].

Un *template* puede estar compuesto para abordar requerimientos de complejas transformaciones. Las grandes transformaciones pueden ser estructurados dentro de módulos (*module*) que tienen partes públicas y privadas [80].

Por ejemplo, la figura 3-20 presenta la especificación de un *template* para generar una definición de Java para una clase UML [80].

```
[template public classToJava(c : Class)]
class [c.name/]
{
   // Constructor
   [c.name/]()
   {
   }
}
[/template]
```

Figura 3-20. Plantilla de generación de código [80].

Un *template* puede invocar a otros *templates*. La invocación de un template es equivalente a to *in situ* del texto producido por el template que se invoca [80].

```
[template public classToJava(c : Class)]
class [c.name/]
{
   // Attribute declarations
   [attributeToJava(c.attribute)/]

   // Constructor
   [c.name/]()
   {
   }
}
[/template]

[template public attributeToJava(a : Attribute)]
[a.type.name/] [a.name/];
[/template]
```

Figura 3-21. Plantilla con invocación a otras plantillas de generación de código [75].

La figura 3-21, presenta el *template classToJava* que invoca al *template attributeToJava* para cada atributo de la clase y pone ';' como separador entre los fragmentos de textos producidos por cada invocación del *template attributeToJava* [80]. Un *template* puede iterar sobre una colección mediante el uso de un bloque *for,* por ejemplo *[for]..[/for]* [80].

Las navegaciones sobre modelo complejo pueden ser especificadas mediante *queries*. El siguiente ejemplo en la figura 3-22 muestra el uso de un *query allOperations* para recoger las operaciones de todas las clases padres abstractas de una clase en una jerarquía de clases [80].

```
[query public allOperations(c: Class) : Set ( Operation ) =
c.operation->union( c.superClass->select(sc|sc.isAbstract=true)-
>iterate(ac : Class;
         os:Set(Operation) = Set{}| os->union(allOperations(ac)))) /]

[template public classToJava(c : Class) ? (c.isAbstract = false)]
class [c.name/]
{
// Attribute declarations
   [attributeToJava(c.attribute)/]
   // Constructor
   [c.name/]()
   {
   }
   [operationToJava(allOperations(c))/]
}

[/template]

[template public operationToJava(o : Operation)]
[o.type.name/] [o.name/] ([for(p:Parameter | o.parameter)
separator(',')] [p.type/] [p.name/] [/for]);
[/template]
```

Figura 3-22. *Query* en la generación de texto [80].

3.3.2.9. Espacios tecnológicos

En esta sección se describen las áreas tecnológicas relacionadas con el trabajo.

Eclipse

La comunidad Eclipse [107] es una comunidad de código abierto cuyos proyectos se centran en la creación de una plataforma de desarrollo abierta compuesta por marcos extensibles, herramientas para la creación, implementación y gestión de software a través del ciclo de vida.

Uno de sus proyectos más importantes es Eclipse. Eclipse es un proyecto cogido abierto, robusto, con múltiples características, plataforma industrial de calidad comercial para el desarrollo de herramientas altamente integradas.

Entorno de desarrollo integrado (IDE) utiliza diferentes módulos para mejorar su funcionalidad; Esto es una ventaja sobre otros ambientes monolíticos donde la funcionalidad no es configurable.

En definitiva, la naturaleza de esta herramienta es un IDE abierto y ampliable para múltiples propósitos. Este trabajo será de particular interés el Eclipse Modeling Framework (EMF), un marco para la gestión de modelos y generación de código de modelos que se describen en XMI. EMF es descrito en detalle en el apartado siguiente.

Eclipse Modeling Framework (EMF)

El proyecto EMF es un marco de modelado y generación de código para la creación de herramientas y otras aplicaciones basadas en modelos de datos estructurados. Todo inicia con una especificación del modelo descrito en XMI. EMF proporciona las herramientas necesarias para producir un conjunto de clases Java para el modelo.

Los modelos se pueden especificar utilizando Java anotado, documentos XML, o las herramientas de modelado como Rational Rose a través del cual pueden ser importados a EMF. Lo más importante es que EMF proporciona la base para establecer la interoperabilidad con otras herramientas y aplicaciones basadas en EMF. Con respecto a la relación de EMF a la OMG y MOF, EMF empezó como una implementación de la especificación madura MOF de la experiencia adquirida en el desarrollo de herramientas por los desarrolladores de Eclipse. EMF puede verse como una implementación eficiente de utilización conjunta de la API de MOF. Sin embargo, para evitar confusión, el metamodelo de EMF está basado en el núcleo de MOF y es llamado Ecore [107].

ATL - una tecnología de transformación de modelos

ATL es un componente del proyecto MMT [71] que tiene como objetivo proporcionar un conjunto de herramientas de transformación de modelo a modelo. Estos incluyen algunas transformaciones ATL, que muestran un motor de transformación ATL y un IDE para ATL [7].

ATL (Atlas Transformación Language) es un lenguaje de transformación de modelos y kit de herramientas [7]. En el campo de la Ingeniería Dirigida por Modelos (MDE), ATL proporciona formas para producir un conjunto de modelos destino de un conjunto de modelos origen. Desarrollado sobre la plataforma Eclipse, proporciona una serie de herramientas de desarrollo estándar (resaltado de sintaxis, depurador, etc.) que tiene como objetivo facilitar el desarrollo de las transformaciones ATL [7].

Acceleo

Acceleo es una implementación pragmática del estándar *MOF Modelo to Text Language (MTL)* del *Object Management Group (OMG)*. Acceleo es el resultado de varios años de investigación y desarrollo. Se inició en la empresa francesa Obeo [78], la unión entre el estándar MTL OMG, y la experiencia de su equipo con la generación de código industrial y los últimos avances en el campo de investigación M2T, ofrecen ventajas excepcionales: alta capacidad personalización, interoperabilidad, gestión de la trazabilidad, etc [75].

Visual Studio

Visual Studio es una colección completa de herramientas y servicios que permiten crear una gran variedad de aplicaciones, tanto para plataformas de Microsoft como para otras plataformas [64].

Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Phyton, Ruby, php; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual se puede sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Mónaco [119].

Capítulo 4. Generación de aplicaciones Cloud desde SoaML y SLA

Este capítulo presenta el enfoque basado en modelos que se ha definido para generar de forma automática aplicaciones Cloud desde modelos SoaML y SLA, implementadas en Windows Azure. Para llevar a cabo esta generación se ha definido un proceso de migración, que está basado en la notación SPEM [84].

En la Sección 4.1 se introduce los metamodelos de la especificación SoaML, del Acuerdo de Nivel de Servicio SLA, Cloud y Windows Azure.

En la Sección 4.2 se presenta el proceso de migración llevado a cabo con el fin de migrar aplicaciones SOA a Cloud Computing.

En la Sección 4.3 se presentan las correspondencias definidas entre SoaML, SLA, Cloud y Windows Azure.

En la Sección 4.4 se presentan las transformaciones definidas para cada una de las correspondencias.

4.1. Metamodelos

4.1.1. SoaML

El metamodelo de SoaML extiende el metamodelo UML2 para apoyar un modelado de servicios explícito en entornos distribuidos. Esta extensión tiene como objetivo apoyar a los diferentes escenarios de modelado de servicios tales como la descripción de un solo servicio, modelado de arquitectura orientada al servicio, o la definición de contrato de servicios [83].

El metamodelo UML2 se extiende en cinco áreas principales: Los participantes, servicios, interfaces, contrato de servicios, y datos de servicio. Los participantes permiten definir los proveedores de servicios y los consumidores en un sistema. Las ServiceInterfaces hacen posible modelar explícitamente las operaciones proporcionadas y necesarias para completar la funcionalidad de un servicio. Los ServiceContracts se utilizan para describir los patrones de interacción entre las entidades de servicio. Por último, el metamodelo también proporciona elementos para el modelo mensajes de servicio de manera explícita y también para modelar datos adjuntos de mensajes [83].

La figura 4-1 ilustra los elementos que soportan los participantes y el modelado de una ServiceInterface. Un participante puede jugar el papel de proveedor de servicios, consumidor, o ambas cosas. Cuando un participante trabaja como proveedor contiene puntos de servicio (*service points*). Por otro lado, cuando un participante trabaja como un consumidor contiene puertos de solicitud (*request port*) [83].

Un ServiceInterface puede ser usado como el protocolo de un servicio o un puerto de solicitud [83].

- Si es usado en un puerto de servicio, significa que el participante que posee los puertos es capaz de implementar la ServiceInterface.
- Si es usado en un puerto de solicitud, significa que el participante utiliza esa ServiceInterface.

La relación "Makes" se deriva de "ownedPort", donde el puerto es una solicitud. La relación "ofertas" se deriva de "ownedPort", donde el puerto es un servicio [83].

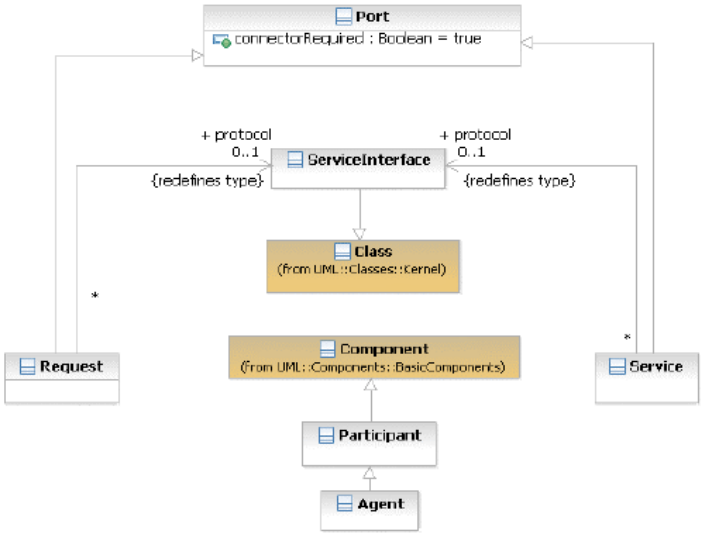

Figura 4-1. Participantes y ServiceInterfaces [83].

La figura 4-2 presenta los elementos que apoyan el modelado del *ServiceContract*. Más adelante estos contratos pueden ser realizados por los elementos del servicio, tales como participantes, *ServiceInterfaces* o *ConnectableElements* (*Request* o *Service Port*). Otro de los elementos incluidos en esta figura es la *ServiceArchitecture*. La ServiceArchitecture se utiliza para modelar las arquitecturas de servicios y sus comportamientos [83].

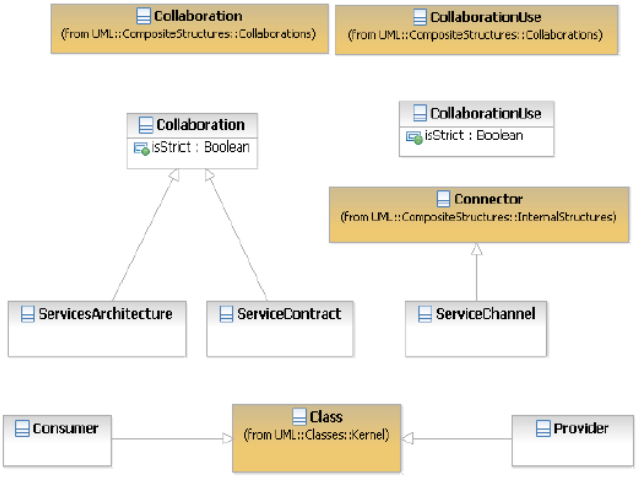

Figura 4-2. ServiceContract y ServiceArchitecture [83].

La figura 4-3 presenta los elementos que dan soporte al modelado de datos. Los *attachment* se utilizan para modelar los elementos que tienen su propia identidad cuando se toman fuera del sistema.

Por ejemplo, se puede definir un *attachment* para enviar un documento de texto que puede utilizarse con otras aplicaciones externas al sistema.

El *MessageType* se utiliza para identificar explícitamente los elementos de datos que viajan entre los participantes de la interacción del servicio.

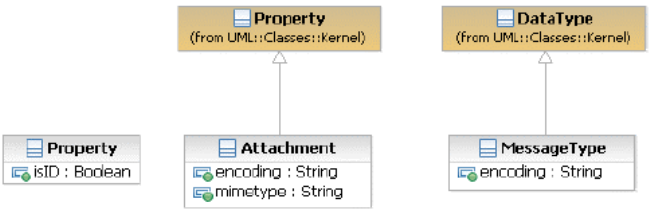

Figura 4-3. Service Data [83].

Nuestro metamodelo ha sido construido basándose en la especificación propuesta por la OMG [83], y también por los aportes del proyecto Minerva [23] y del Irish Software Engineering Centre [2].

Elementos del metamodelo de SoaML

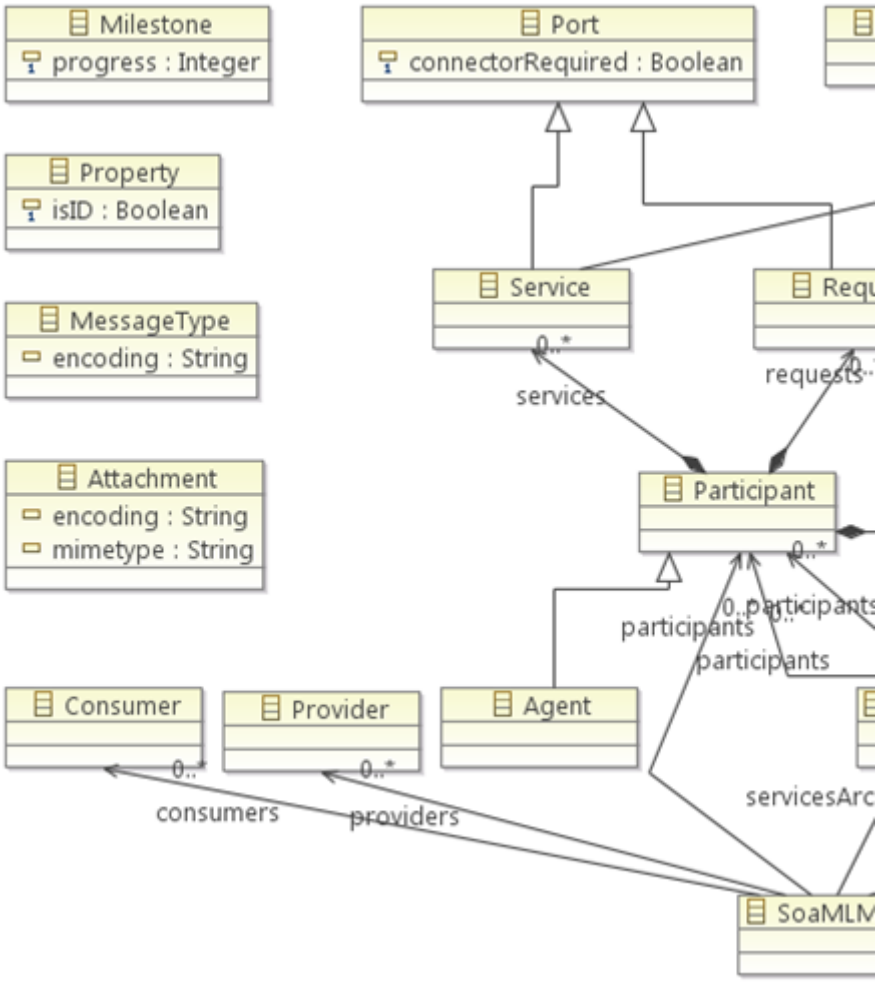

Figura 4-4. Metamodelo de SoaML (Parte A)

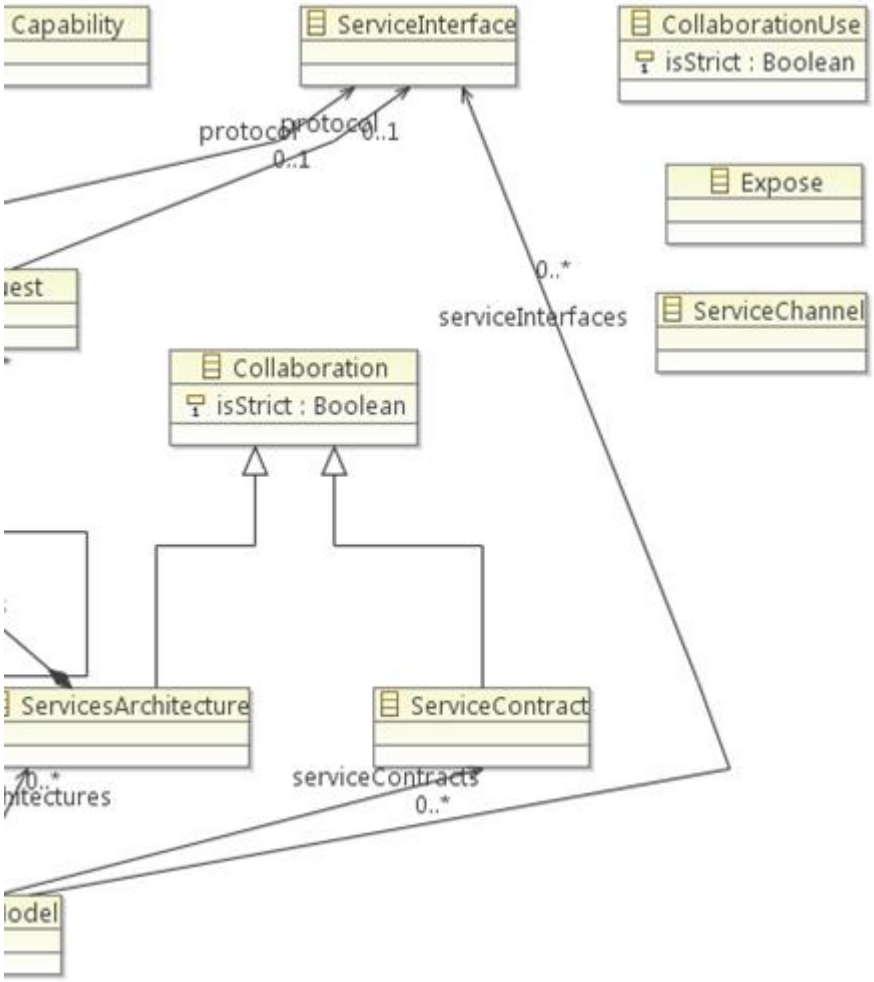

Figura 4-5. Metamodelo de SoaML (Parte B).

Los elementos del metamodelo de la especificación SoaML utilizados en este trabajo de investigación, son presentados en las figuras 4-4 y 4-5, y a continuación se detalla cada uno de los elementos:

Agent

Un *agent* es una clasificación de las entidades autónomas que se pueden adaptar e interactuar con su entorno. Describe un conjunto de instancias de agentes que tienen en común las características, restricciones y semántica. Los agentes en SoaML también son participantes, proveen y usan servicios. Generaliza la metaclase *Participant* [83].

Attachement

Una parte de un mensaje que está adjuntada o más bien contenida en el mensaje. Extiende la metaclase *Property* de UML [83].

Capability

Un *capability* es la capacidad para actuar y producir un resultado que logra un resultado. Puede especificar una capacidad general de un participante, así como la capacidad específica para prestar un servicio. Extiende la metaclase *Class* de UML [83].

Consumer

Un *consumer* se usa como el tipo de un rol en un contrato de servicio, y el tipo de puerto de un participante. Extiende la metaclase *Class* (en el caso de la composición de un contrato de servicio), o la metaclase *Interface* (en el caso de la no composición de un contrato de servicio) ambas de UML [83].

Collaboration

Un *collaboration* es extendido para indicar sí el rol a la parte de enlaces de *CollaborationUses* escritos por un *collaboration* son estrictamente forzados o no. Extiende la metaclase *Collaboration* de UML [83].

CollaborationUse

Un *collaborationUse* es extendido para indicar sí el rol a la parte de los enlaces son estrictamente forzados o sueltos. Extiende la metaclase *CollaborationUse* de UML [83].

Expose

Una dependencia *expose* se utiliza para indicar una funcionalidad expuesta a través de una ServiceInterface. El origen de un *expose* es la ServiceInterface, y el destino es la *capability* expuesta. Extiende la metaclase *Dependency* de UML [83].

MessageType

Un *messageType* es utilizado para la especificación de la información intercambiada entre proveedores y consumidores del servicio. Extiende las metaclases *DataType, Class y Signal* de UML [83].

Milestone

Un *milestone* es un medio para representar el progreso en determinados comportamientos con el fin de analizar el *liveness*. Los *milestones* son particularmente útiles para los comportamientos que son de larga duración o incluso infinitos. Extiende la metaclase *Comment* de UML [83].

Participant

Un *participant* es el tipo de proveedor o consumidor de servicios. En el dominio del negocio un participante puede ser una persona, una organización o un sistema.

En el dominio de los sistemas un participante puede ser un sistema, aplicación o componente. Extiende las metaclases *Class y Component* de UML [83].

Port

Un *port* extiende la metaclase *Port* de UML como medio para indicar si se requiere una conexión en este puerto o no [83].

Property

Un *property* aumenta el estándar *Property* de UML con la capacidad de ser distinguido como una propiedad identificativa y significa que la propiedad se puede utilizar para distinguir las instancias del clasificador que contiene. También es conocido como la "clave principal." En el contexto de SoaML el identificador se utiliza para distinguir el identificador de correlación en un mensaje. [83].

Provider

Un *provider* se utiliza como el tipo de rol en un contrato de servicio, y el tipo de puerto en un participante. Extiende la metaclase *Class* (en el caso de la composición de un contrato de servicio), o la metaclase *Interface* (en el caso de la no composición de un contrato de servicio) ambas de UML [83].

Request

Un *request* representa una característica de un participante que es el consumo de un servicio por un participante proporcionados por terceros usando términos, condiciones e interfaces bien definidas. Un *request* designa los puertos que definen el punto de conexión a través del cual un participante cumple sus necesidades a través del consumo de los servicios prestados por terceros. Extiende la metaclase *Port* de UML [83].

ServiceChannel

Un *serviceChannel* representa una ruta de comunicación entre servicios y solicitudes dentro de una arquitectura. Extiende la metaclase *Connector* de UML [83].

ServiceContract

Un *serviceContract* es la formalización de un intercambio de información, bienes o las obligaciones entre las partes que definen un servicio. Extiende la metaclase *Collaboration* de UML [83].

ServiceInterface

Un *serviceInterface* proporciona la definición de un servicio. Define la especificación de la interacción de un servicio, como el tipo de puerto «Servicio» o «Solicitud». Extiende la metaclase *Class* de UML [83].

Service

Un *service* representa una característica de un participante que es la oferta de un servicio por parte de un participante a otros usando interfaces, condiciones y términos bien definidos. Un servicio designa un puerto que define el punto de conexión a través del cual un participante ofrece sus capacidades y provee un servicio a los clientes. Extiende la metaclase *Port* de UML [83].

ServiceArchitecture

La visión de alto nivel de un *serviceArchitecture* define como un conjunto de participantes trabajan juntos, formando una comunidad, para algún propósito de proporcionar y usar servicios. Extiende la metaclase *Collaboration* de UML [83].

SoaMLModel

Un *SoaMLModel* permite el modelado de aplicaciones usando la especificación SoaML. En la que pueden trabajar participantes, contratos de servicios, interfaces de servicios, consumidores, proveedores, datos de servicios, entre otros. Extiende la metaclase *Package* de UML.

4.1.2. Service Level Agreement

SLA o Service Level Agreement, traducido como Acuerdo de Nivel de Servicio, es un documento habitualmente anexo al Contrato de Prestación de Servicios. En el SLA se estipulan las condiciones y parámetros que comprometen al prestador del servicio (habitualmente el proveedor) a cumplir con unos niveles de calidad de servicio frente al contratante de los mismos (habitualmente el cliente) [1].

A diferencia de los productos tangibles que se pueden ver, tocar o manipular, los servicios se basan en la "confianza" que deposita el cliente frente al proveedor por diferentes motivos como la empatía, el conocimiento o el prestigio. La confianza es un término subjetivo. La fórmula que permite definir una serie de medidas objetivas que comprometen al proveedor a ofrecer determinado nivel de calidad es mediante el SLA.

Es importante que las condiciones de calidad afecten a todos los elementos implicados en el servicio y que en el SLA se especifiquen los términos y parámetros sobre los que se adquiere el compromiso en el servicio, se indique el modo de cálculo (métrica e intervalos) del índice de cumplimiento, cuál es el objetivo pactado; indicando el valor o márgenes de referencia, cuáles las posibles compensaciones por incumplimiento y por último las exclusiones o limitaciones en dichos cálculos.

La creación de acuerdos de nivel de servicio proporciona ciertos requisitos de los clientes y proveedores. Necesidad de los clientes ser capaz de cumplir con ciertos requisitos para poder con éxito definir los SLAs, que aparecen brevemente aquí [32]. Un cliente debe:

- Comprender las funciones y responsabilidades que están reguladas por el SLA.
- Ser capaz de describir específica y precisamente el servicio a ser controlado por el SLA.
- Conocer los requerimientos de los servicios controlados, y definir las figuras claves que asimila.
- Especificar los niveles de servicio basados en el rendimiento críticos de las características del servicio.
- Comprender el proceso y los procedimientos de regulación del servicio.

4.1.2.1. Estructura del SLA

La estructura se basa en los elementos presentados en una estructura ejemplar que puede crearse en un SLA. Esto puede verse en la figura 4-6. Está claro que la descripción de los servicios, o nivel de servicio, los objetivos son el aspecto central de cada SLA [32].

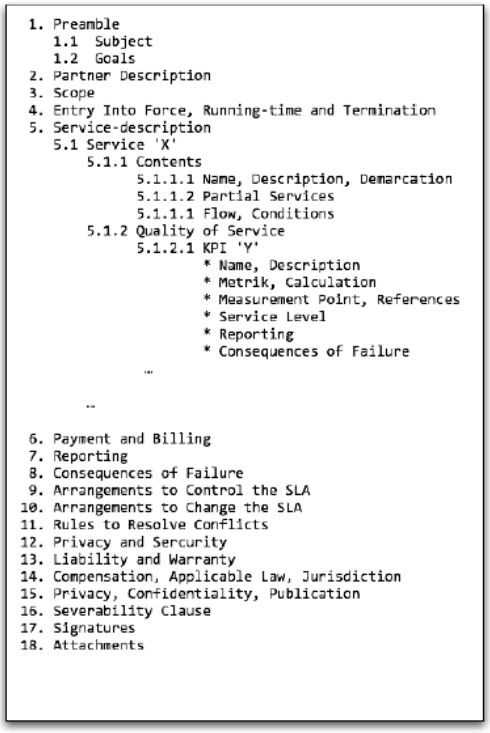

Figura 4-6. Estructura de un SLA [22].

Elementos del metamodelo de SLA

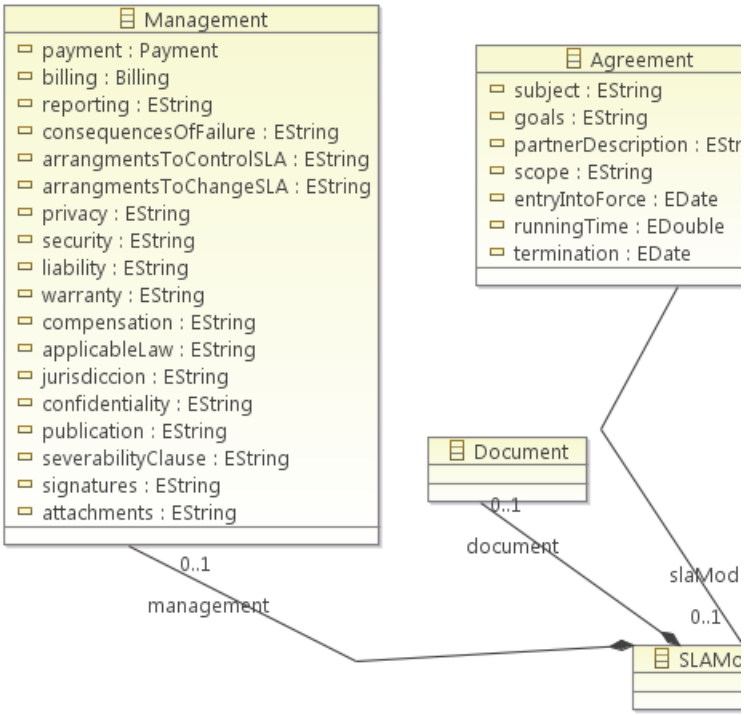

Figura 4-7. Metamodelo de SLA (Parte A).

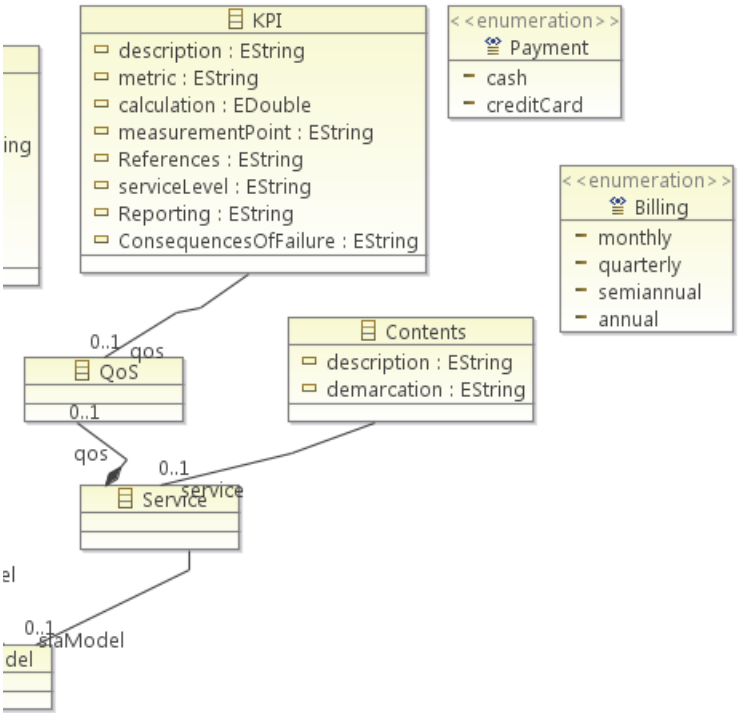

Figura 4-8. Metamodelo de SLA (Parte B).

La estructura de los acuerdos de nivel de servicio es generalmente un escenario específico y puede ser muy mal generalizado. Sin embargo, hay algunos elementos básicos que deben estar presentes en cada SLA. Las figuras 4-7 y 4-8 muestran los elementos del metamodelo de SLA de forma genérico construido para este trabajo de investigación, a continuación se detalla cada uno de los elementos:

Agreement

Un *agreement* permite modelar los elementos relacionados con el acuerdo contienen las reglas básicas del acuerdo e incluyen, entre otros, el tema de los SLAs, objetivos, socios, así como el alcance, la entrada en vigor, duración y terminación de los SLAs.

Service

Un *service* permite el modelado de los elementos relacionados con el servicio representan aquellos elementos que describen la regulación de un servicio.

KeyPerfomaceIndicator (KPI)

Un *KPI* permite modelar los indicadores claves de rendimiento relacionados con el servicio. Estos deben ser especificados individualmente para cada servicio.

Contents

Un *contents* permite el modelado de los elementos relacionados con los documentos incluyen elementos administrativos y editoriales, que juegan un papel de menor importancia en el SLA y existen principalmente para mejorar el manejo, comprensión y legibilidad.

Management

Un *management* permite modelar los elementos relacionados con la gestión incluyen los aspectos que tienen que ver con la administración y control de los SLAs.

SLAModel

Un *SLAModel* permite el modelado de un acuerdo de servicio. En el que se estipula el acuerdo, el servicio, las condiciones, indicadores de calidad, y la gestión del mismo.

4.3.1. Cloud

SOA es un paraguas que describe cualquier tipo de servicio. Una aplicación Cloud es un servicio. Una metamodelo de una aplicación Cloud es un modelo SOA conforme al metamodelo de SOA. Esto hace que las aplicaciones Cloud también sean aplicaciones SOA. Sin embargo, las aplicaciones SOA no son necesarias en las aplicaciones Cloud [38]. Una aplicación Cloud es una aplicación SOA que se ejecuta bajo un específico entorno, que es el entorno (plataforma) de la computación en la nube o Cloud Computing. Este entorno se caracteriza por la escalabilidad horizontal, rápido aprovisionamiento, facilidad de acceso y precios flexibles [38].

Elementos del metamodelo

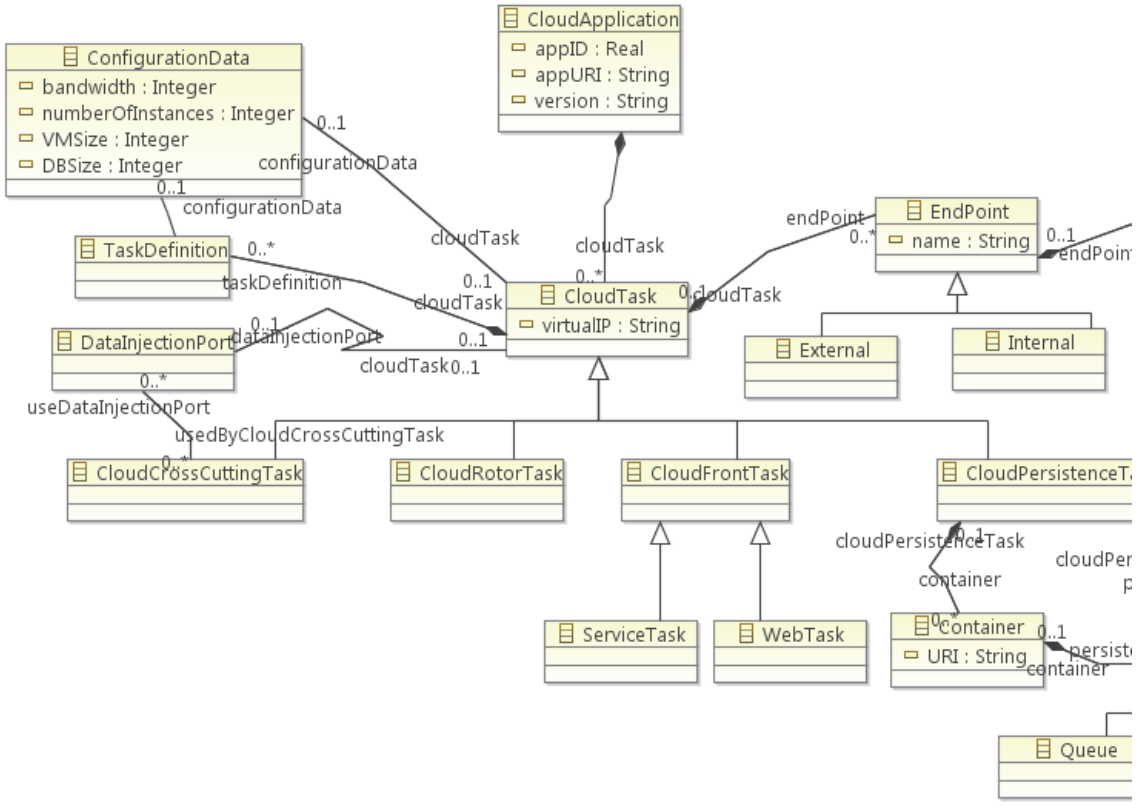

Figura 4-9. Metamodelo de Cloud (Parte A) [38].

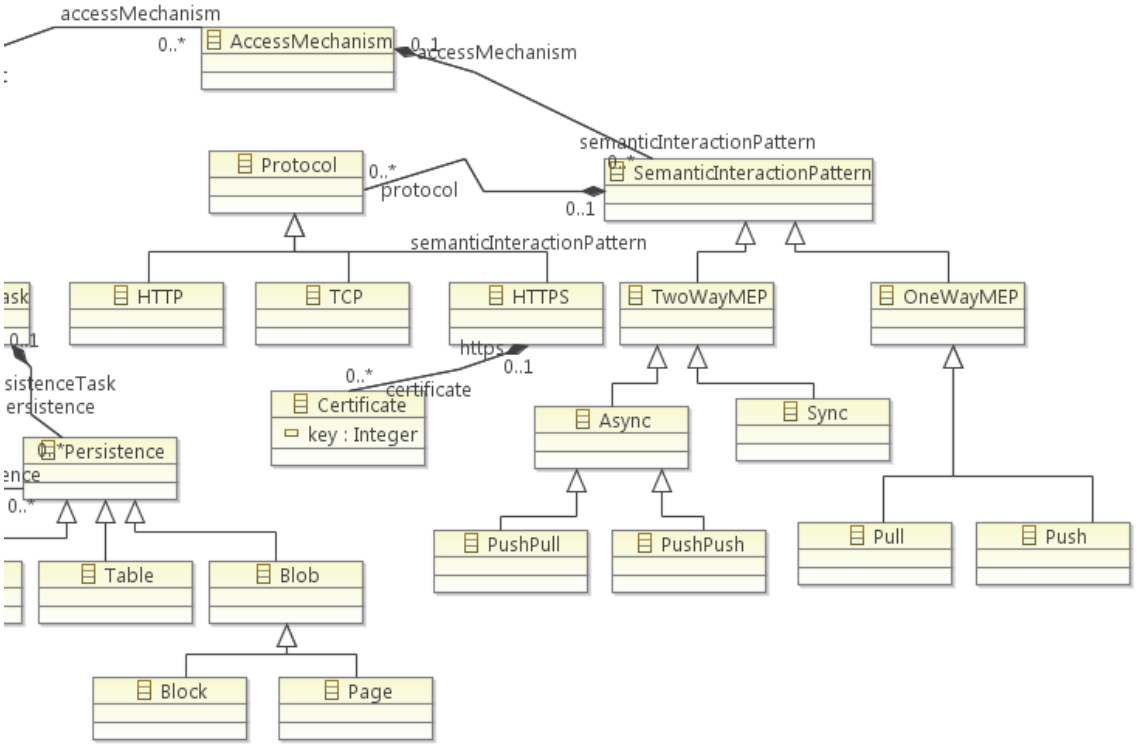

Figura 4-10. Metamodelo de Cloud (Parte B) [38].

Las figuras 4-9 y 4-10 muestran los elementos del metamodelo de Cloud, a continuación se detalla cada uno de los elementos:

CloudApplication

Una *CloudApplication* no es ni orientada a servicios ni una aplicación web estándar [38]. Las aplicaciones cloud combinan el software tradicional con servicios remotos proporcionando alta disponibilidad, escalabilidad y de alto rendimiento del software, además de la tolerancia a fallos, almacenamiento escalable que puede escalar a petabytes.

Una aplicación en la nube apoya al paradigma del Software más Servicios (S + S) [101], donde cada aplicación consiste en un número de tareas.

CloudTask

Una *CloudTask* es una unidad componible, que consiste en un conjunto de acciones que utilizan los servicios proporcionando una funcionalidad específica para resolver un problema [38]. Además, es una unidad mutable que se puede copiar a otra máquina virtual con el fin de permitir la escalabilidad horizontal y vertical.

TaskDefinition

Una *TaskDefinition* contiene información sobre las tareas de la aplicación en la nube, que se determinan en tiempo de diseño [38]. Una *TaskDefinition* ofrece la estructura de la aplicación de nube, en términos de tareas proporcionadas, sus tipos y relaciones. También proporciona el conjunto de interfaces de tareas y sus contratos.

ConfigurationData

Las aplicaciones en la nube tienen la capacidad de escalar hacia dentro y hacia afuera. Esto puede ser logrado mediante la clonación de las tareas en las máquinas virtuales en tiempo de ejecución para satisfacer la demanda laboral cambiante. Un *configurationData,* es donde, los aspectos dinámicos de la aplicación en el cloud se determinan en tiempo de ejecución [38].

DataInjectionPort

Las propiedades de las tareas se pueden modificar en tiempo de ejecución. Esto se puede lograr a través de un *DataInjectionPort* [38]. Un *DataInjectionPort* modifica las propiedades de las tareas transversales, tales como, los relacionados con la calidad de servicio (QoS).

CloudFrontTask

Una *CloudFrontTask* se encarga de las peticiones del usuario, que son distribuidas por un balanceador de carga [38].

CloudRotorTask

Una ***CloudRotorTask:*** Ejecuta en segundo plano el *CloudFrontTask* en el centro de datos en la nube [38].

CloudCrossCuttingTask

Una ***CloudCrossCuttingTask:*** Es responsable de la gestión de aspectos transversales como los relacionados con los recursos de monitoreo de la nube, que incluye instancias de almacenamiento y cómputo, y una carga balanceador para garantizar la utilización de los recursos y el rendimiento [38].

CloudPersistenceTask

Una ***CloudPersistenceTask:*** Es el responsable de la gestión de las cuentas de almacenamiento. Además, gestiona el control de acceso e inicio de sesión en el almacenamiento de la nube. Una nube de almacenamiento (por ejemplo, blob, tabla, cola) no tiene ningún mecanismo de control de acceso [38].

Containers

Las CloudPersistenceTasks crean *containers*, que son análogos, pero sin fusionar. Los *containers* son accesibles a través de un identificador de recursos uniforme único (URI). Las *CloudPersistenceTasks* asignan persistencia a los *containers* y les da un URI único que es accesible público o privadamente. La *CloudPersistenceTask* admite tres principales tipos de almacenamiento en la nube que son fiables, pueden escalar, simples, no costosos y tienen un mejor rendimiento en el entorno de la nube. Estos tipos son: los datos no estructurados (*blobs*), los datos estructurados (tablas), y la mensajería asíncrona (colas) [38].

Blobs

Los *blobs* son archivos grandes de datos no estructurados y sus metadatos. Se pueden almacenar como una secuencia de bloques o páginas.

Tables

Las *tables* son archivos de datos estructurados, que son más complejos que los blobs, pero diferente a las tablas de las bases de datos relacionales (RDB).

Queue

Un almacenamiento de mensajes escalable, que soporta el modelo basado en el sondeo es usado en el paso de mensajes entre tareas. Un mensaje puede ser almacenado durante largos periodos (por ejemplo días) antes de que se lea y luego sea removido de la cola (***queue***) [38].

EndPoint

Las relaciones entre las tareas se pueden determinar por los *EndPoints* [38]. Los *EndPoints* son puertos a través del cual una *CloudTask* puede conectarse a otras tareas o al entorno.

Cada *EndPoints* utiliza un ***AccessMechanism***, que utiliza un ***SemanticInteractionPattern*** para la coordinación de intercambio de mensajes. Estos patrones se basan en protocolos específicos que determinan la sintaxis y la semántica de los mensajes que se intercambian entre de dos partes de la comunicación [38].

Estos patrones se basan en protocolos específicos que determinan la sintaxis y la semántica de los mensajes que se intercambian entre las dos partes de la comunicación. Los ***MessageExchangePatterns*** (MEP) se pueden clasificar en dos categorías principales, unidireccionales o bidireccionales [38].

4.3.2. Windows Azure

Como se mencionó el capítulo 3 Windows Azure es un conjunto de tecnologías que proporciona un conjunto específico de servicios a desarrolladores de aplicaciones [17]. Además ofrece una plataforma donde el ISV (proveedor independiente de software) y las empresas pueden hospedar sus aplicaciones y datos.

Para este trabajo de investigación seleccionó esta plataforma de servicio para el hospedaje y despliegue de las aplicaciones obtenidas después de realizar la migración.

Para tal efecto, se creó un metamodelo genérico de la plataforma azure, el mismo que tiene características del entorno de desarrollo y la plataforma.

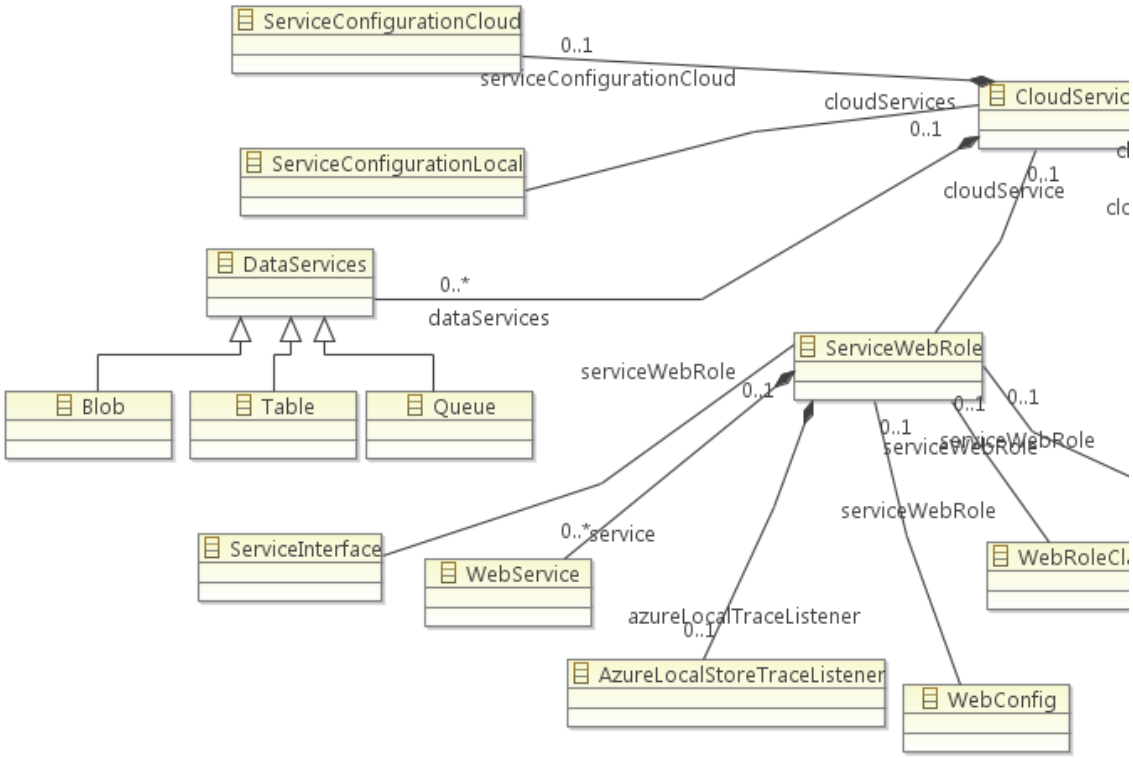

Figura 4-11. Metamodelo genérico de Windows Azure (Parte A)

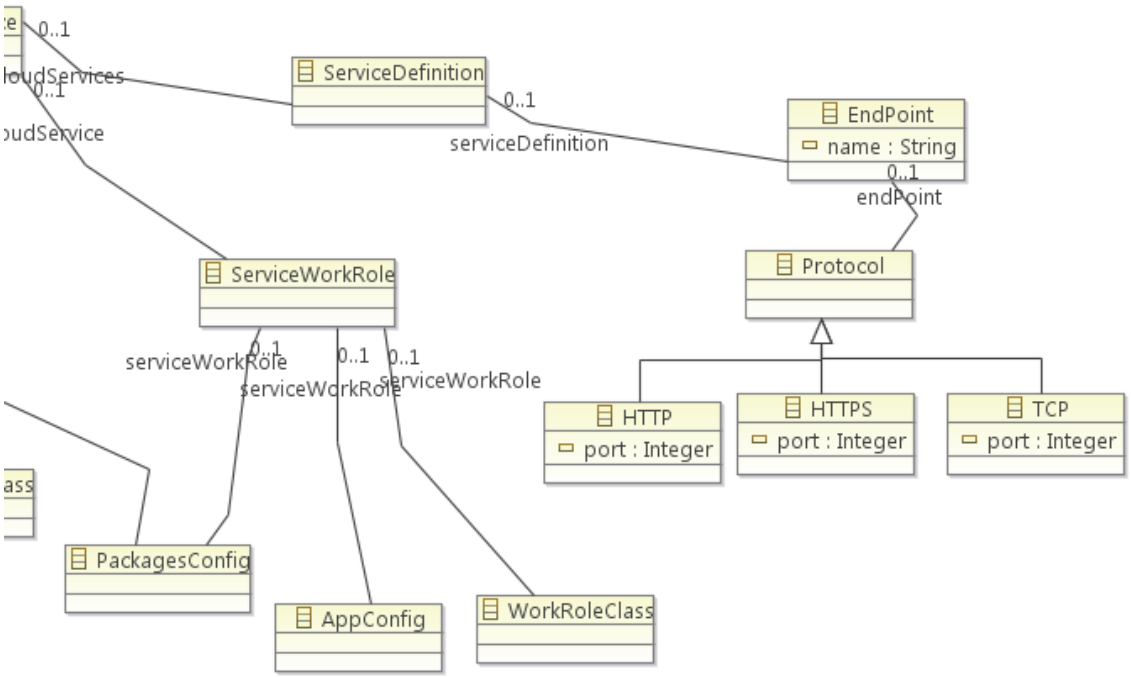

Figura 4-12. Metamodelo genérico de Windows Azure (Parte B)

Las figuras 4-11 y 4-12 muestran los elementos del metamodelo genérico de Azure, a continuación se detalla cada uno de los elementos:

CloudServices

Una *cloudServices* permite crear una solución con las dos diferentes tipos de instancias que proporciona la plataforma de Windows Azure para los desarrolladores, *Web Roles* y *Work Roles*.

ServiceWebRole

Un *WebRole* es un rol que se personaliza para la programación de aplicaciones web como el apoyo de IIS 7, como ASP.NET, PHP, Windows Communication Foundation y FastCGI. Toda la información de un WebRole se almacena en un archivo de definición y la extensión predeterminada para el archivo de definición de servicio es .csdef [65].

ServiceWorkRole

Un *WorkRole* es una función que es útil para el desarrollo generalizado, y puede realizar el procesamiento de fondo para un rol web. Toda la información de un WorkRole se almacena en un archivo de definición y la extensión predeterminada para el archivo de definición de servicio es .csdef [68].

ServiceDefinition

El archivo de definición de servicio Windows Azure define el modelo de servicio para una aplicación. El archivo contiene las definiciones de las funciones disponibles en un servicio, especifica los extremos de servicio, y establece los valores de configuración para el servicio. Valores de los ajustes de configuración se encuentran en el archivo de configuración de servicio, según lo descrito por el esquema de configuración de servicio de Windows Azure (. File cscfg) [67].

ServiceConfiguration

El archivo de configuración de servicio especifica el número de instancias a implementar para cada rol en el servicio, los valores de los parámetros de configuración y las huellas digitales de todos los certificados asociados a un rol.

Si el servicio incluye una máquina virtual de Windows Azure, el archivo de configuración de servicio especifica el disco duro virtual del cual se crea la máquina virtual. Si el servicio es parte de una red virtual, la información de configuración para la red virtual debe ser proporcionada en el archivo de configuración del servicio, así como en el archivo de configuración de red virtual [66].

4.2. Proceso de Migración

En esta sección se presenta el proceso de migración que representa la forma de migrar aplicaciones basadas en SOA a entornos Cloud Computing.

Uno de los aspectos claves de este proceso es el uso de dos modelos como artefactos de entrada, y ejecución de transformaciones de modelos: un modelo de la aplicación SOA basado en la especificación SoaML, y un modelo de acuerdo de servicios basado en SLA.

A diferencia de otros procesos de transformación, que sólo utilizan un modelo de origen como entrada para aplicar las transformaciones.

El proceso de migración se divide en tres fases: análisis, transformación, y despliegue. La figura 4-13 presenta el proceso de migración que permite a los desarrolladores poner en práctica la metodología a seguir.

Este proceso es completamente independiente, ya que permite a los desarrolladores empezar la migración desde cero, es decir, modelando aplicaciones SOA a través de la especificación SoaML, o a partir de aplicaciones existentes modeladas en SoaML.

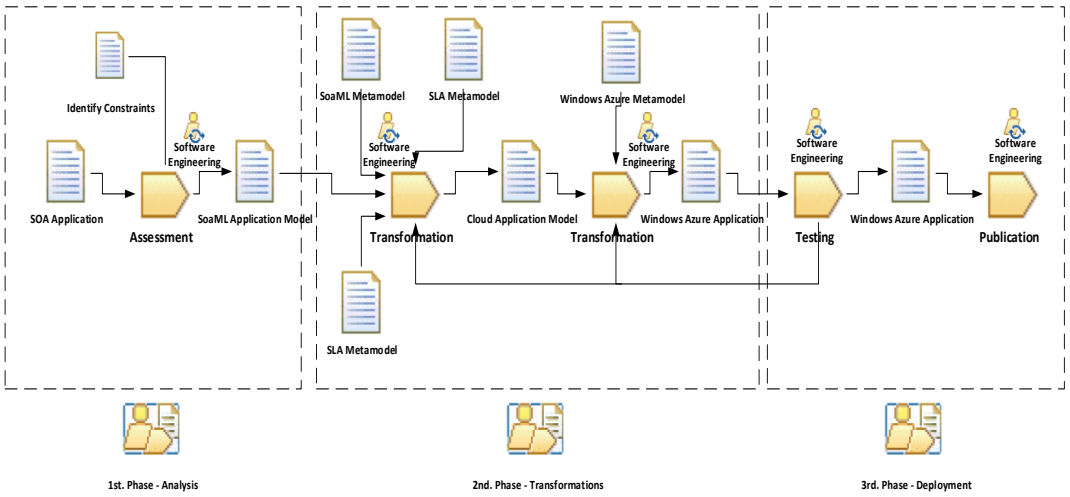

Figura 4-13. Proceso de migración.

En la fase de análisis, el desarrollador debe realizar una valoración de la aplicación SOA a migrar, la misma que estará modelada en la especificación SoaML, identificando las restricciones de rendimiento y de calidad que puede tener el modelo.

También es importante identificar en esta primera etapa que una aplicación SOA también podrá ser código fuente escrito en un lenguaje de programación, para lo cual, es necesario que el desarrollador utilice herramientas que permitan obtener el modelo de la aplicación basado en la especificación SoaML, a través de la ingeniería inversa. Una vez concluida esta fase se entrega un modelo listo a ser migrado.

La fase de transformación, se divide en dos etapas: la primera es la transformación, la misma que utiliza dos modelos como artefactos de entrada, el primer modelo es la aplicación SOA conforme al metamodelo de la especificación SoaML, y el segundo modelo del acuerdo de servicios basado al metamodelo SLA, y se obtiene un modelo de la aplicación conforme al metamodelo de referencia Cloud. A continuación este modelo es transformado a un modelo conforme al metamodelo genérico de la plataforma Windows Azure que ha sido seleccionada para este trabajo de investigación. Finalmente, una vez que obtiene el modelo de la aplicación conforme a la plataforma seleccionada se procede a realizar una transformación de modelo a texto para obtener el código fuente conforme a la tecnología utilizada, para este caso C# de Windows Azure.

La segunda etapa de la fase es la validación del modelo obtenido tras la transformación, en ella se comprueba que todas los mapeos especificados en cada transformación, ya sea de SoaML, SLA a Cloud, y de Cloud a Windows Azure sean correspondidos entre los metamodelos correspondientes.

La última fase es el despliegue, la aplicación obtenida de las fases anteriores será desplegada en un entorno Cloud, para el propósito de este trabajo de investigación este entorno de servicio es Windows Azure.

4.3. Correspondencias

Basado en las definiciones presentadas en el capítulo 3 se identificaron los elementos claves en los metamodelos de SoaML, SLA, Cloud y Windows Azure que se quieren relacionar para generar aplicaciones Cloud a partir de modelos de servicios.

4.2.1. Elementos claves de SoaML en la transformación

Tal como se presentó en el capítulo 3, los elementos claves para el modelado de servicios en este enfoque se basan en:

Un ***participant*** representa alguna parte o componente que proporciona y/o consume servicios (los participantes podrán representar a las personas, organizaciones o sistemas que proveen y / o utilizan los servicios). Un participante es un proveedor de servicios, siempre que ofrezca un servicio. Un participante es un consumidor de servicios si se utiliza un servicio. Un participante puede proporcionar o consumir cualquier cantidad de servicios. Los puertos Service y Request se definen para hacer eso, que luego se identifican mediante las correspondientes Interfaces.

Un ***consumer*** será normalmente el que inicia la interacción de servicios. Las interfaces de consumo se utilizan como el tipo de un *ServiceContract* y están sujetos a los términos y condiciones de dicho contrato de servicio. El *consumer* está destinado a ser utilizado como el tipo de puerto de un participante que utiliza un servicio.

Un ***provider*** entrega los resultados de la interacción de servicio. Normalmente, el proveedor será el que responde a la interacción del servicio. Las interfaces de proveedor se utilizan como el tipo de un *ServiceContract* y están sujetas a los términos y condiciones de dicho contrato de servicio. La interfaz *provider* está destinada a ser utilizada como el tipo de puerto de un participante que proporciona un servicio.

Un ***MessageType*** es un tipo de objeto de valor que representa la información intercambiada entre las solicitudes y servicios de los participantes. Esta información consiste en datos pasados a, y/o retornados de, la invocación de una operación o evento definido en una interfaz de servicio. Un *MessageType* está en el contenido de dominio o de servicio específico y no incluye cabecera u otra aplicación o información específica del protocolo.

4.2.2. Elementos claves de SLA en la transformación

Un *agreement* es en donde se especifica la calidad con la que deben prestarse los servicios contratados.

Un *KPI* incluye indicadores o métricas que pueden incluir: que porcentaje del tiempo de servicio estará disponible, número de instancias, ancho de banda, entre otras.

4.2.3. Elementos claves de Cloud en la transformación

Una *CloudApplication* será normalmente donde se inicia la aplicación desde un entorno Cloud, la misma que incluye su identificador, URI, versión.

Una *CloudTask* será normalmente donde se ejecuten todas las acciones en una aplicación en un entorno Cloud. Una CloudTask están semánticamente conectadas a otras tareas a través de roles (EndPoint), que juegan un papel para satisfacer los requerimientos del negocio. Puede incluir un nombre, y su URI.

Un *ServiceTask* también podrá ser visto como un servicio web, que es proporcionado por un tercero. Un *ServiceTask* utiliza el Enterprise Service Bus (EBS) para descubrir y acceder a los servicios remotos o empresariales.

Un *Queue* será utilizado para almacenar los mensajes que son intercambiados entre las tareas.

4.2.4. Elementos claves de Windows Azure en la transformación

Un *CloudServices* será utilizado para crear las aplicaciones basadas en la plataforma de azure.

Un *ServiceWebRole* será utilizado para crear los servicios web de la aplicación dentro de la plataforma seleccionada.

Un *ServiceWorkRole* será utilizado para crear los roles de trabajo, y el encargado del procesamiento de los web roles.

4.2.5. Definición de correspondencias entre SoaML, SLA vs Cloud

Para la especificación de las transformaciones de modelos ATL se definieron las correspondencias entre elementos de SoaML, SLA y Cloud. Estas correspondencias especifican que elementos del modelo origen conforme al metamodelo origen, se transformarán en los elementos del modelo de destino conforme al metamodelo destino, cuando se ejecute la transformación.

Estas correspondencias entre los elementos en los metamodelos fueron definidas previamente de la definición de las transformaciones de modelo ATL, las mismas que se muestran en la tabla 4-1 y 4-2.

SoaML (Elementos)	Cloud (Elementos)
SoaMLModel	CloudApplication
Participant; Agent	CloudTask
Consumer Provider ServiceInterface	TaskDefinition
MessageType	Queue

Tabla 4-1. Correspondencias entre elementos de SoaML y Cloud.

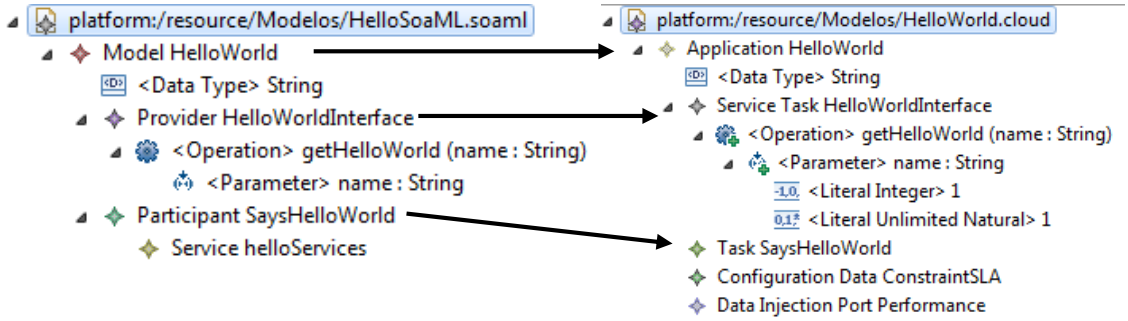

Figura 4-14. Correspondencias entre SoaML vs Cloud.

ServiceLevelAgreement (Elementos)	Cloud (Elementos)
Agreements	ConfigurationData
KPIs	DataInjectionPort

Tabla 4-2. Correspondencias entre elementos de SLA y Cloud.

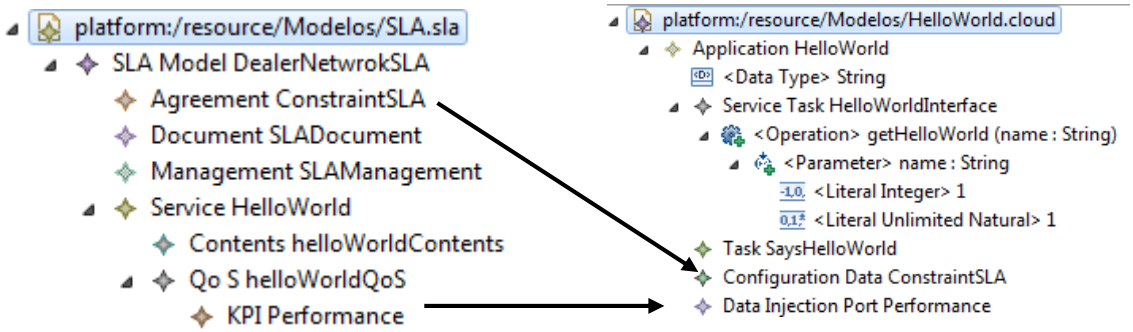

Figura 4-15. Correspondencias entre SLA vs Cloud.

En primer lugar se presentan las correspondencias entre los metamodelos SoaML y Cloud, las mismas que se detallan a continuación:

- El elemento *SoaMLModel,* y *ServiceArchitecture* (si existe en el modelo) corresponde al elemento *CloudApplication* que incluye al resto de elementos del modelo.
- Los elementos *Participant* y *Agent* corresponden a *CloudTask* que es donde se ejecutan las acciones de una aplicación cloud.
- Los elementos *Consumer, Provider, y ServiceInterface* corresponden a *TaskDefinition* que en cloud provee la estructura de la aplicación y el conjunto de interfaces y sus contratos.

- El elemento MessageType corresponde a Queue que en Cloud es el encargado de la coordinación del intercambio de mensajes.

En segundo lugar se presentan las correspondencias entre los metamodelos SLA y Cloud, las mismas que se describen a continuación:

- El elemento *SLA* corresponde al elemento *ConfigurationData* que es donde los aspectos relacionados a los acuerdos de servicio de la aplicación Cloud son determinados.
- El elemento *KPI* corresponde al elemento *DataInjectionPort* que es donde se modifican tareas relacionadas a la calidad del servicio (QoS).

4.2.6. Definición de correspondencias entre Cloud vs Azure

Para la especificación de las transformaciones de modelos ATL se definieron las correspondencias entre elementos de Cloud y Azure. Estas correspondencias especifican que elementos del modelo origen conforme al metamodelo origen, se transformarán en los elementos del modelo de destino conforme al metamodelo destino, cuando se ejecute la transformación.

Estas correspondencias entre los elementos en los metamodelos fueron definidas previamente de la definición de las transformaciones de modelos ATL, las mismas que se muestran en la tabla 4-3.

Cloud	WindowsAzure
CloudApplication	CloudServices
WebTask **+ ServiceTask**	ServiceWebRole
CloudRotorTask	ServiceWorkerRole
Table	SQL Azure Table
Queue	SQL Azure Queue
Blob	SQL Azure Blob
EndPoint	EndPoint
Protocol	Protocol
ConfigurationData	ServiceConfigurationLocal + ServiceConfigurationCloud

Tabla 4-3. Correspondencias entre elementos de Cloud y Azure.

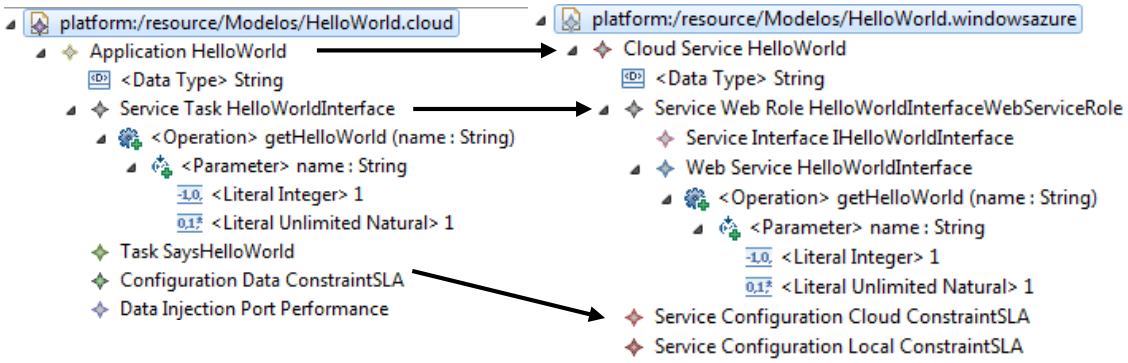

Figura 4-16. Correspondencias entre Cloud vs Azure.

Las correspondencias entre los metamodelos Cloud y Windows Azure, se detallan a continuación:

- El elemento *CloudApplication* corresponde al elemento *CloudServices* que incluye al resto de elementos del modelo.
- Los elementos *WebTask* y *ServiceTask* corresponden a *ServiceWebRole* que es los servicios de la aplicación serán especificados.
- El elemento *CloudRotorTask* corresponden a *ServiceWorkRole* que en es el encargado del procesamiento de los web roles en una aplicación en la plataforma de azure.
- El elemento *Table* corresponde a *SQL Azure Table* que en azure es el encargado del almacenamiento de información estructurada.
- El elemento *Queue* corresponde a *SQL Azure Queue* que en azure es el encargado del almacenamiento de mensajes escalables.
- El elemento *Blob* corresponde a *SQL Azure Blob* que en azure es el encargado del almacenamiento de datos no estructurados.
- El elemento *EndPoint* corresponde a *EndPoint* que en azure es el encargado del intercambio de información entre web roles y work roles.
- El elemento *Protocol* corresponde a *Protocol* que en azure es el puerto utilizado por los web roles.
- El elemento *ConfigurationData* corresponde a *ServiceConfiguration* que en azure almacena datos de configuración de los web roles, y work roles.

4.4. Transformaciones

Las transformaciones de modelos se definieron a través del lenguaje de transformación de modelos ATL, tal como se presenta en el capítulo 3, definiendo las plantillas de objetos del dominio de origen SoaML, SLA al dominio de destino Cloud. Como los metamodelos de SoaML, SLA y Cloud hacen referencia a elementos del metamodelo de UML, se incluye también elementos UML, los mismos que son accesibles desde las reglas.

Las reglas definidas permiten que las transformaciones ATL generen de forma automática los elementos del modelo de destino Cloud como se define en las correspondencias, así como como las relaciones entre los elementos generados en el modelo resultante.

Las transformaciones entre los metamodelos SoaML, SLA, Cloud y Windows Azure, se realizaron en dos fases:

En la primera fase se realizaron las transformaciones de modelos entre SoaML, SLA a Cloud, y desde Cloud a Azure; donde todos los elementos claves identificados de SoaML previamente identificados en la sección 4.2.1.

Y los elementos claves de SLA también identificados en la sección 4.2.2, se correspondieron a cada elemento clave identificado en el metamodelo de Cloud tal como lo indica la sección 4.2.3. A continuación se presenta un extracto de estas transformaciones:

```
-- @nsURI Cloud=http://Cloud.ecore

-- @nsURI SoaML=http://SoaML.ecore
-- @nsURI SLA=http://sla.ecore

module soaml2cloud;
create OUT: Cloud from IN: SoaML, IN1: SLA;

-- SoaMLModel To CloudApplication
-- In this part was mandatory to include SLA Elements.
rule SoaMLModelToCloudApplication {
    from
        sm: SoaML!SoaMLModel,
        sl: SLA!SLAModel
    to
        ca: Cloud!CloudApplication (
            clientDependency <- sm.clientDependency,
            name <- sm.name.toString(),
            templateParameter <- sm.templateParameter,
            URI <- sm.URI,
            visibility <- sm.visibility,
            packagedElement <- sm.packagedElement,
            packagedElement <- sl.packagedElement
        )
}
```

Esta regla de transformación permite que un elemento *SoaMLModel* se ha convertido en una *CloudApplication*. También es muy importante destacar, que se debe incluir el elemento *SLAModel* para poder ejecutar la transformación entre SoaML, SLA, y Cloud.

```
-- @nsURI Azure=htt://WindowsAzure.ecore
-- @nsURI Cloud=http://Cloud.ecore

module cloud2azure;
create OUT: Azure from IN: Cloud;

-- CloudApplication To CloudServices
rule CloudApplicationToCloudServices {
    from
        ca: Cloud!CloudApplication,
        cd: Cloud!ConfigurationData
    to
        wa: Azure!CloudService (
            clientDependency <- ca.clientDependency,
            name <- ca.name.toString(),
            templateParameter <- ca.templateParameter,
            URI <- ca.URI,
            visibility <- ca.visibility,
            packagedElement <- ca.packagedElement,
            packagedElement <- configCloud,
            packagedElement <- configLocal
        ),
```

```
        configCloud: Azure!ServiceConfigurationCloud (
            owner <- cd.owner,
            clientDependency <- cd.clientDependency,
            isAbstract <- cd.isAbstract,
            isFinalSpecialization <- cd.isFinalSpecialization,
            isLeaf <- cd.isLeaf,
            name <- cd.name.toString(),
            powertypeExtent <- cd.powertypeExtent,
            redefinedClassifier <- cd.redefinedClassifier,
            representation <- cd.representation,
            templateParameter <- cd.templateParameter,
            useCase <- cd.useCase,
            visibility <- cd.visibility,
            ownedAttribute <- cd.ownedAttribute,
            ownedOperation <- cd.ownedOperation
        ),
        configLocal: Azure!ServiceConfigurationLocal (
            clientDependency <- cd.clientDependency,
            isAbstract <- cd.isAbstract,
            isFinalSpecialization <- cd.isFinalSpecialization,
            isLeaf <- cd.isLeaf,
            name <- cd.name.toString(),
            powertypeExtent <- cd.powertypeExtent,
            redefinedClassifier <- cd.redefinedClassifier,
            representation <- cd.representation,
            templateParameter <- cd.templateParameter,
            useCase <- cd.useCase,
            visibility <- cd.visibility,
            ownedAttribute <- cd.ownedAttribute,
            ownedOperation <- cd.ownedOperation
        )
}
```

Esta regla de transformación permite que un elemento *CloudApplication* se ha convertido en un *CloudServices*. También es muy importante destacar, que se debe incluir el elemento *ConfigurationData* para poder de esta forma poder crear los elementos correspondientes a los ServiceConfiguration.Local y ServiceConfiguration.Local del metamodelo genérico de la plataforma Azure.

En la segunda fase se realizaron las transformaciones de modelos a texto desde el metamodelo genérico de la plataforma Azure a código fuente correspondiente al lenguaje de programación C# utilizado en la herramienta Visual Studio.

Capítulo 5. Ejemplo de Aplicación

Nuestro escenario de ejemplo de negocio (ver figura 5-1) es una comunidad de distribuidores independientes, fabricantes y transportistas que quieren ser capaces de trabajar juntos de forma coherente y no volver a diseñar los procesos de negocio o sistemas cuando se trabaja con otras partes de la comunidad.

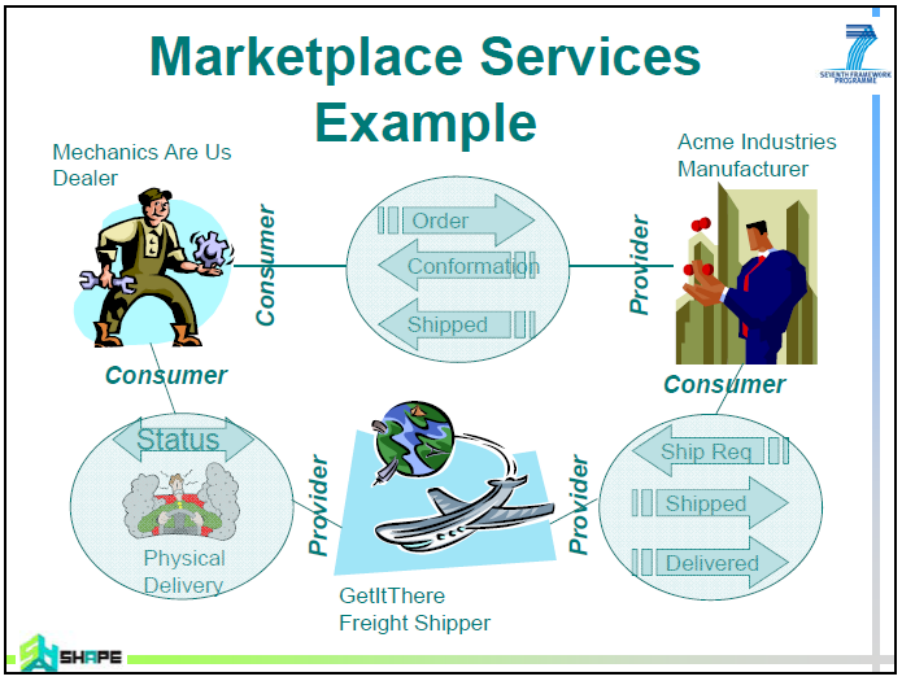

Figura 5-1. Comunidad Dealers Network [27].

Por otro lado, quieren ser capaces de tener sus propios procesos de negocio, reglas e información. La comunidad ha decidido definir una arquitectura orientada al servicio de la comunidad para permitir este entorno de negocio abierto y ágil. Este ejemplo ha sido definido para representar la especificación de SoaML hecha por la OMG [83].

5.1. Definición de la comunidad

La red de distribuidores se define como una comunidad de "colaboración" que implica tres papeles principales para los participantes en esta comunidad: *dealer*, *manufacturer* y *shipper*. Del mismo modo, los participantes participan en los tres servicios "B2B", un servicio de compras, un servicio de envío y un servicio del estado del servicio. La figura 5-2 ilustra estas funciones y servicios en la arquitectura de la red del distribuidor.

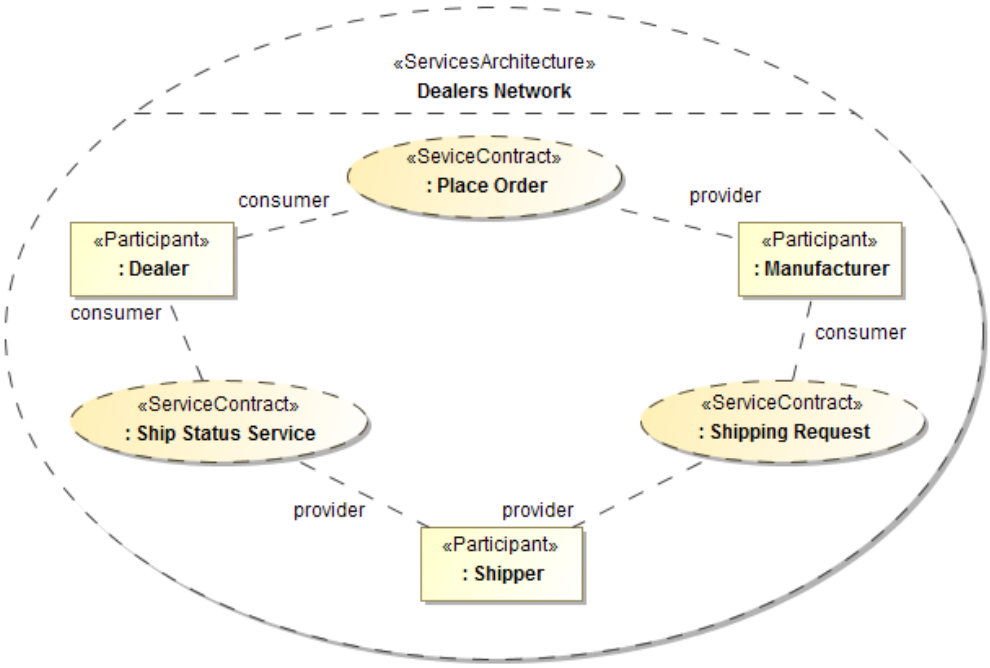

Figura 5-2. Arquitectura DealerNetwork.

Existe una correspondencia directa entre las funciones y servicios a la comunidad de distribuidores, y SOA define a la "ServicesArchitecture" como una comunidad de colaboración de SoaML. La arquitectura de servicios proporciona una vista de alto nivel y contextual de los roles de los participantes y los servicios sin mostrar excesivo detalle. Note que el detalle está totalmente integrado con la vista de alto nivel que proporciona trazabilidad desde la arquitectura a través de la aplicación.

Un detalle adicional que puede verse en este diagrama es el papel que desempeñan los participantes con respecto a cada uno de los servicios. Tenga en cuenta que el *manufacturer* desempeña el papel de *provider* con respecto a *place order* y el *dealer* desempeña el papel de *consumer* en relación con el mismo servicio. Tenga en cuenta también que el *manufacturer* desempeña el papel *consumer* en relación con el servicio *shipping request*.

Es común que los participantes juegan papeles múltiples en múltiples servicios dentro de una arquitectura. El mismo participante puede ser proveedor de algunos servicios y un consumidor de otros.

5.2. Servicios que soporta la comunidad

La arquitectura de servicios no define el servicio, utiliza la especificación del servicio. El servicio es definido de forma independiente y luego se suelta en la arquitectura de servicio para que pueda ser utilizado y reutilizado.

El servicio *place order service* es un servicio simple. El diagrama de la figura 5-3 diagrama identifica el *service contract,* los términos y condiciones del servicio, así como también define los dos roles: *order placer* y *order taker*.

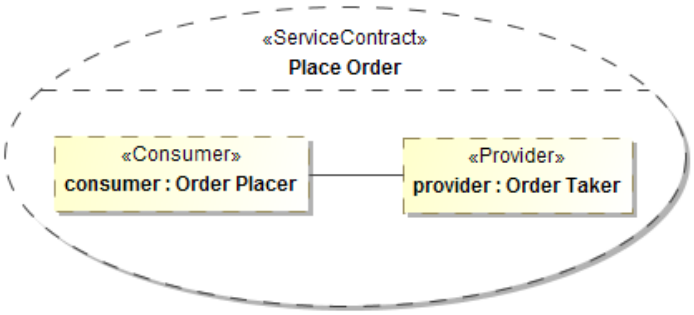

Figura 5-3. Place Order Service.

El servicio *shipping request service* (figura 5-6) es un servicio simple. El diagrama de la figura 5-4 identifica el *service contract,* los términos y condiciones del servicio, así como también define los dos roles: *shipping consumer* y *shipping provider*.

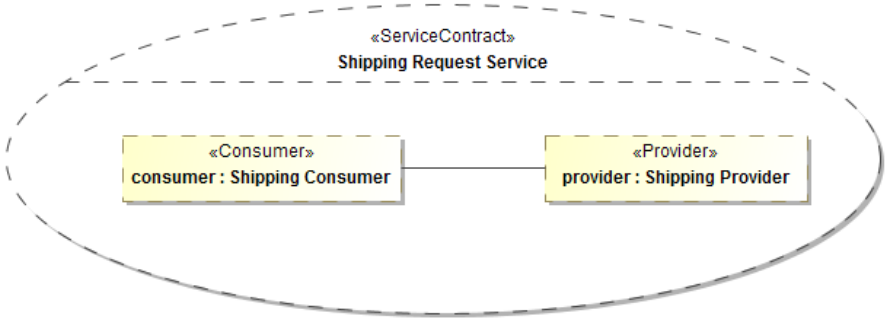

Figura 5-4. Shipping Request Service.

El servicio *ship status service* es un servicio simple. El diagrama de la figura 5-5 identifica el *service contract,* los términos y condiciones del servicio, así como también define los dos roles: *receiver* y *shipper*.

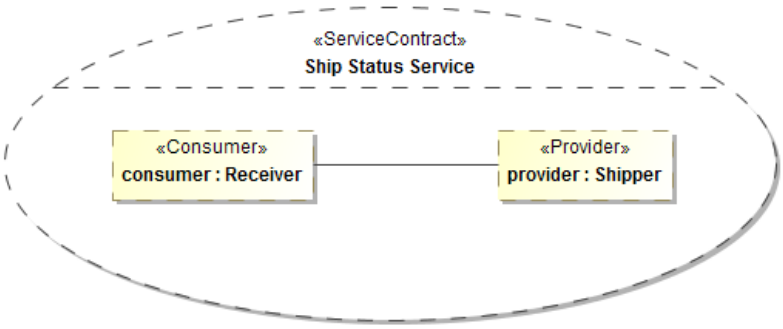

Figura 5-5. Ship Status Service.

5.3. Aplicación del procedimiento de generación

En el entorno de desarrollo Eclipse – SOA2Cloud se creó el modelo SoaML y se guardó en formato *.soaml*. Además también se creó el modelo para especificar los acuerdos del servicio entre el consumidor y el proveedor de servicios SLA en formato *.sla*. Este entorno utiliza como entrada para las transformaciones los dos modelos *.soaml* (figura 5-6) y *.sla* (figura 5-7) respectivamente.

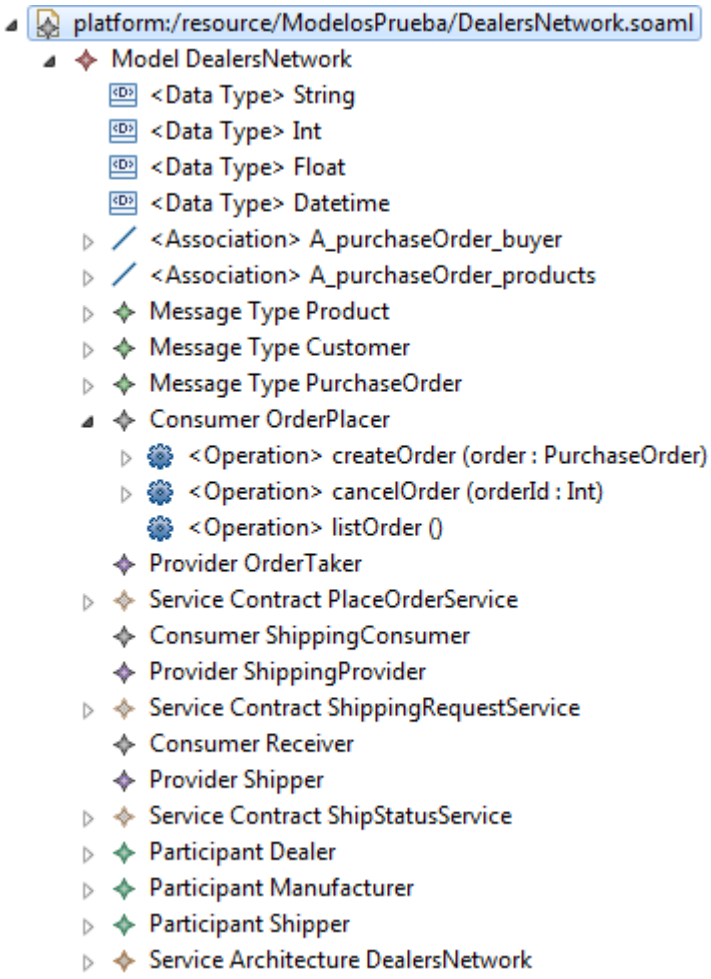

Figura 5-6. Modelo DealersNetwork en SoaML.

El modelo de la figura 5-6 representa la comunidad DealersNetwork modelada a través de la especificación SoaML, usada en el presente trabajo de investigación. Este modelo consta de un *service architecture*, tres *participants*, tres *service constracts*, tres *consumers*, tres *providers, y tres mesagges types.* Los mismos que interactúan entre sí para formar la comunidad de servicios. También es importante mencionar que para el modelado en SoaML es prescindible utilizar varios elementos de UML, por ejemplo: *data type*, *asociation*, *operation*, entre otros.

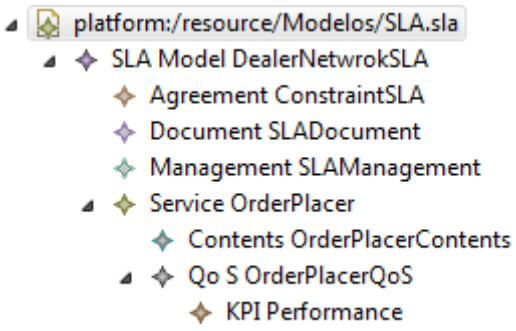

Figura 5-7. Modelo SLA para DealersNetwork.

El modelo de la figura 5-7 representa los acuerdos de nivel de servico para la comunidad DealersNetwork modelada a través del metamodelo genérico de SLA, usado en el presente trabajo de investigación. Este modelo consta de un *agreement*, un *document*, un *management*, un *service*, un *QoS,* y un *kpi.* Los mismos que interactúan entre sí para especificar los acuerdos de nivel de servicio en la comunidad de servicios.

Una vez que ambos modelos han sido creados/modelados, se ejecutaron las transformaciones de modelos a través del lenguaje de transformación de modelos ATL, la misma que nos dió como resultado un nuevo modelo conforme al metamodelo Cloud (figura 5-9).

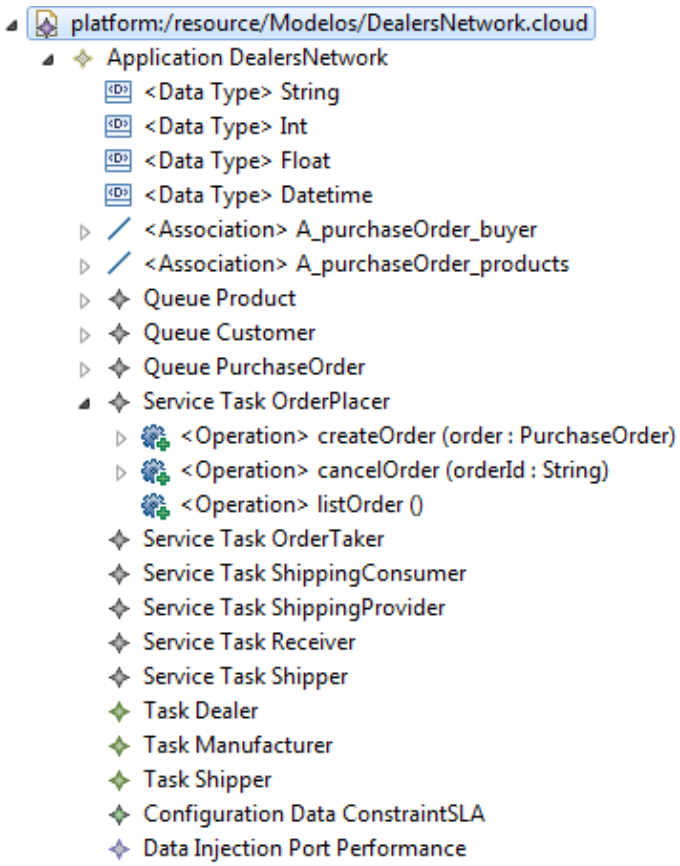

Figura 5-8. Modelo Cloud.

Luego de obtener el modelo de la aplicación Cloud, fue necesario realizar una nueva transformación de modelos, pero esta vez solo se necesito como entrada el modelo Cloud para obtener el modelo para la plataforma Azure (figura 5-10).

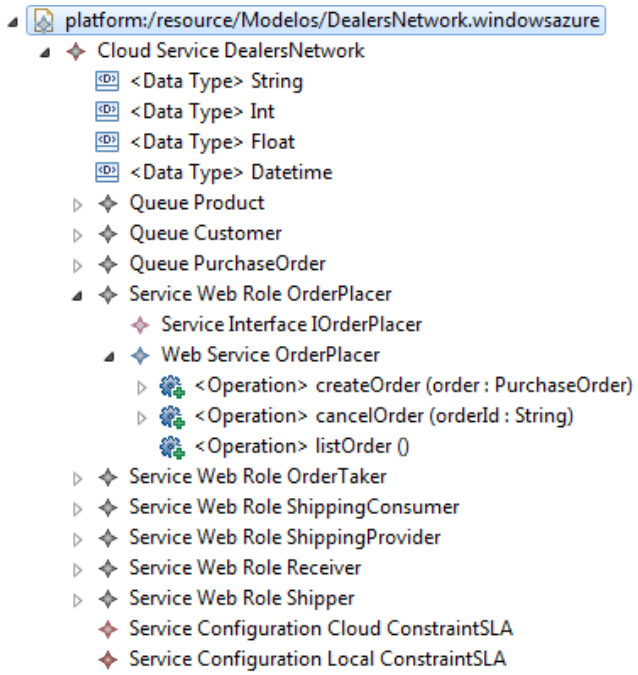

Figura 5-9. Modelo Windows Azure.

Por último, a partir del modelo de la aplicación en la plataforma Windows Azure se procedió a generar la aplicación (figura 5-10), la misma que fue implementada en Visual Studio para su compilación, testeo, y despliegue o publicación (figura 5-11) de los servicios en la plataforma de Microsoft Windows Azure.

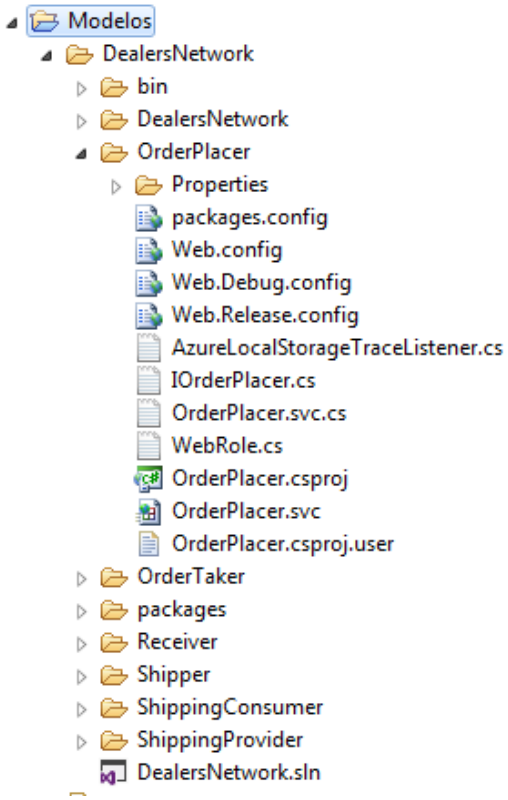

Figura 5-10. Aplicación Windows Azure.

Cabe indicar que una vez que ha sido implementado en Visual Studio, fué necesario realizar algunas modificaciones en el código fuente, respecto a operaciones en las interfaces y servicios. También muy importante el uso de la librería System.Runtime.Serialization, ya que muchas veces ocasionó problemas al momento del testeo, esto se debe a que la compilación y la generación de las librerías dinámicas de los servicios no son creados correctamente, por lo que se debe realizar la compilación varias veces, y si es el caso eliminar la referencia y volver a insertarla.

Además, fué necesario eliminar servicios que no tengan ninguna funcionalidad, para tal efecto en el modelo DealersNetwork, se contaba también con *OrderTaker, ShippingConsumer, ShippingProvider*, *Receiver* y *Shipper.*

Para este ejemplo de aplicación, los *data services* fueron incluidos en la interfaz del servicio como *[Datacontract]* y *[Datamember],* que son necasarios para el intercambio de la información entre los servicios.

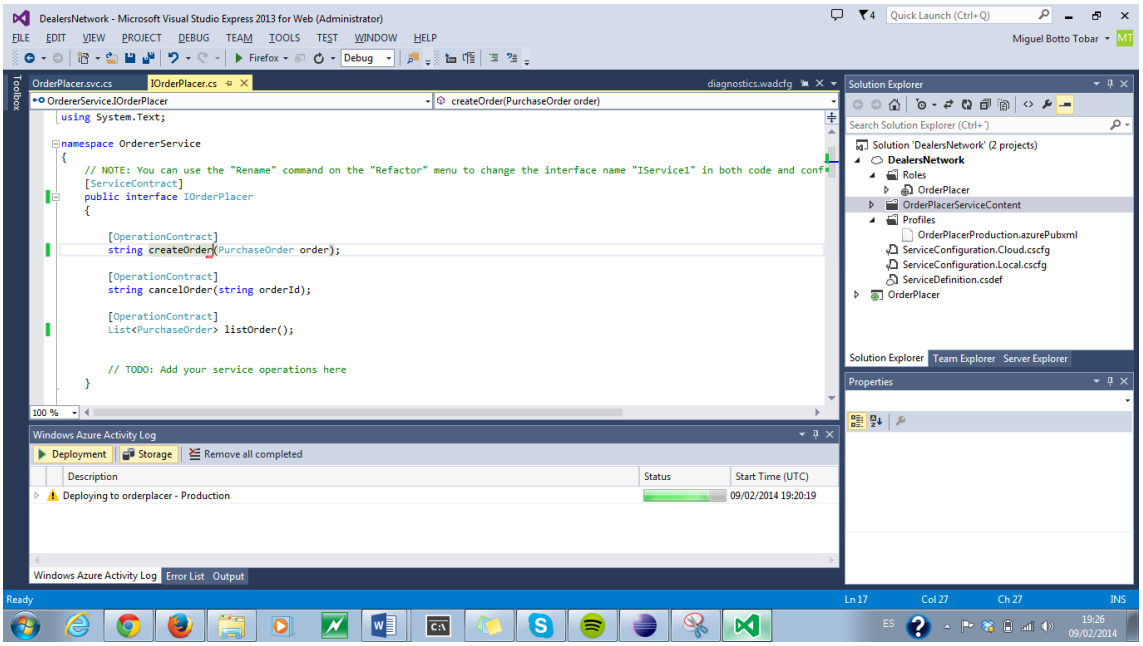

Figura 5-11. Publicación del Servicio a través de Visual Studio

Figura 5-12. Mensaje de Windows Azure

La figura 5-12 indica el mensaje que proporciona la herramienta Visual Studio al confirmar que el servicio ha sido completado correctamente, para tal efecto se lo puede comprobar a través del *manage console* de la plataforma Windows Azure (figura 5-13).

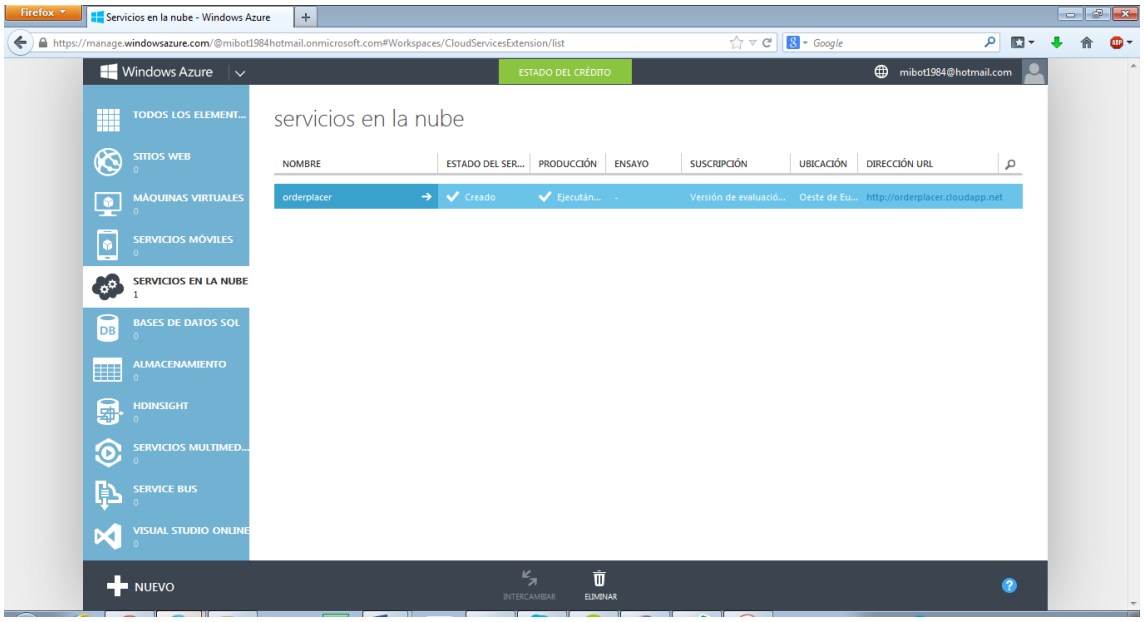

Figura 5-13. Manage Console de Windows Azure.

Para comprobar el correcto funcionamiento del servicio se procedió a ingresar al enlace que proporciona el servicio, el mismo que es: http://orderplacer.cloudapp.net/, y nos abre la siguiente ventana con el servicio publicado (figura 5-14 y 5-15).

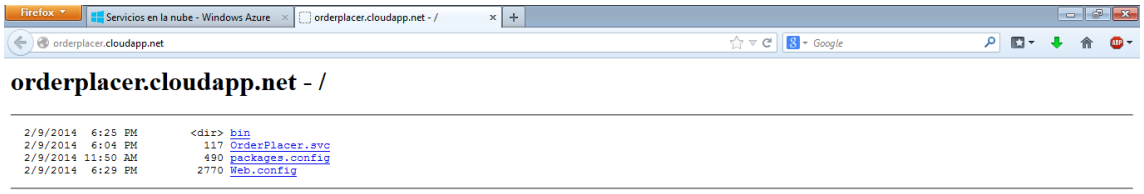

Figura 5-14. Servicio OrderPlacer.

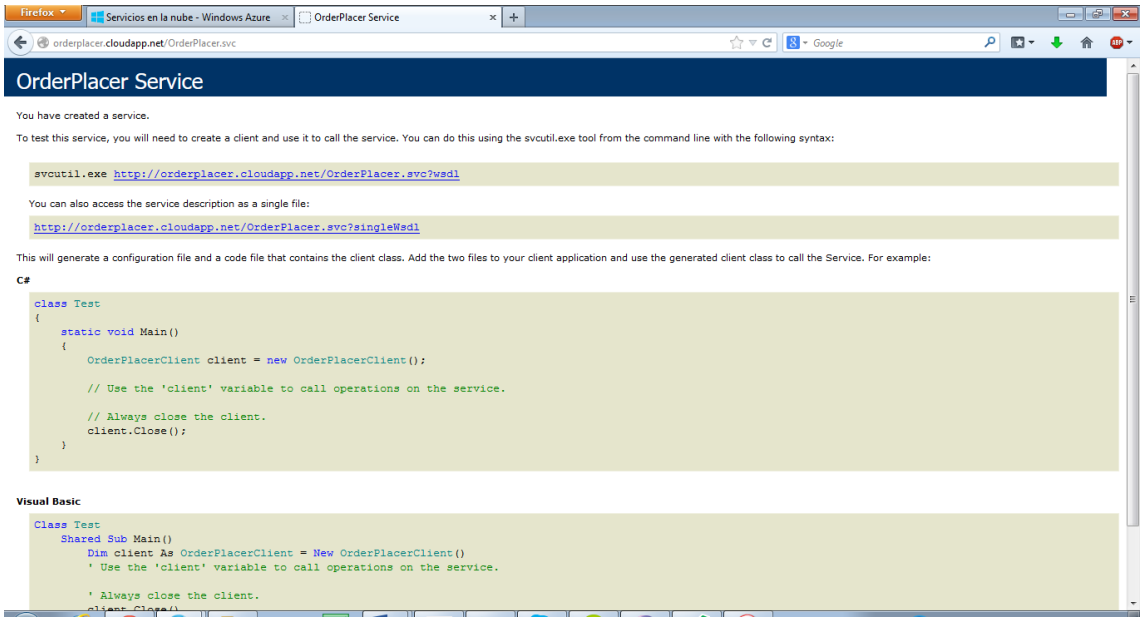

Figura 5-15. Servicio OrderPlacer.

Una vez realizada esta acción se comprobó (figura 5-16) que las operaciones del servicio funcionen correctamente, para tal efecto se utilizó la herramienta proporcionada por Visual Studio Tool, llamada Cliente de Prueba WCF (*wcfTestClient*) [62].

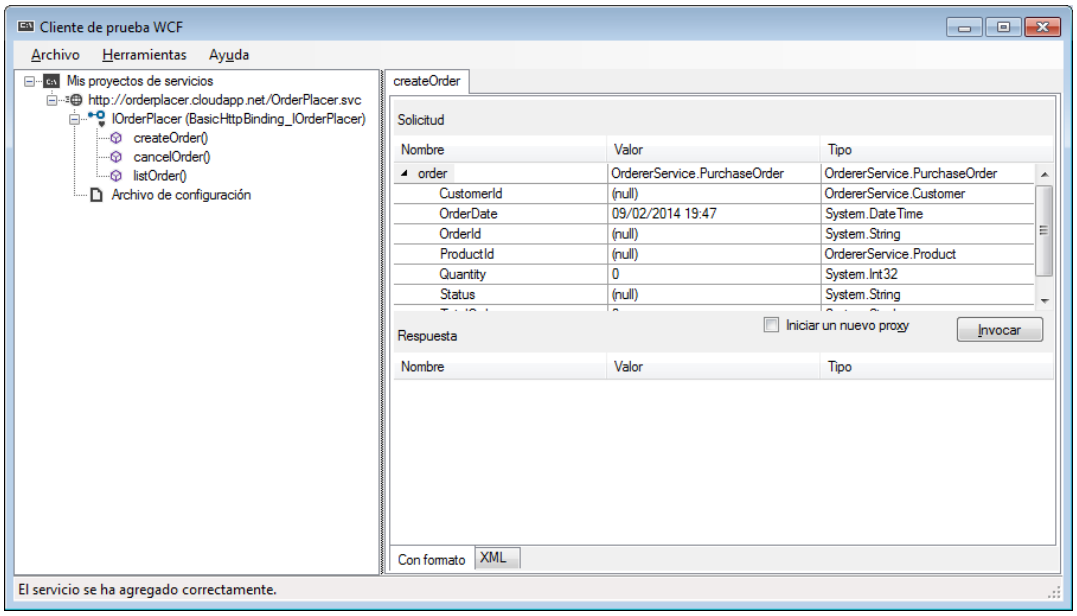

Figura 5-16. Cliente de Prueba WCF.

Es necesario indicar que la comprobación del servicio también se la puede realizar a través del desarrollo de una aplicación ya sea esta de tipo web o de escritorio, para la cual solo es necesario referenciar la URL del servicio.

5.4. Conclusión del procedimiento de generación

La aplicación del marco de trabajo propuesto, nos ha permitido obtener los resultados deseados, ya que mediante este marco se consiguió migrar una aplicación SOA modelada a través de la especificación SoaML, con acuerdos de nivel de servicio a un entorno Cloud, siendo para tal efecto Windows Azure.

Este capítulo ha mostrado, cómo se aplica el marco de trabajo para la migración de aplicaciones SOA a Cloud Computing propuesto en este trabajo. El uso del paradigma del Desarrollo de Software Dirigido por Modelos, fue esencial, ya que a través de los beneficios que aporta este enfoque y con el marco de trabajo propuesto se puede conseguir una migración de una aplicación SOA modelada en SoaML en poco tiempo, que al desarrollarla con otros herramientas.

El proceso de migración propuesto en la sección 4.2, ha servido de base para el marco de trabajo.

Aquí, no solamente hace falta ver la aplicación en conjunto, sino descomponerla y modelarla de acuerdo a la especificación SoaML, para que de esta forma el marco de trabajo propuesta pueda tener éxito.

Por medio de la implementación del ejemplo de aplicación, se ha podido dar una retroalimentación al modelo, mejorarlo y además descubrir aspectos que a simple vista son difíciles de divisar.

Capítulo 6. Conclusiones y Trabajos Futuro

En este capítulo se recogen las conclusiones generales de este trabajo, los trabajos futuros propuestos con miras a la continuación de esta investigación y las publicaciones realizadas como resultados de esta tesina.

6.1. Conclusiones

En los últimos años, Cloud Computing se ha convertido en una opción para utilizar los recursos informáticos como una solución de la crisis económica mundial, es decir, una forma más barata de tener los recursos de TI. Por lo tanto, muchas empresas han comenzado a migrar sus sistemas para infraestructuras de nube, y sin el apoyo suficiente para llevar a cabo este proceso.

La motivación principal de este trabajo de investigación, es el contar con un marco de trabajo que permita la migración de aplicaciones SOA a Cloud, siguiendo una aproximación dirigida por modelos, ya que existen propuestas que realizan este proceso, pero que no abarcan aspectos del desarrollo de software dirigido por modelos.

Teniendo esto en consideración, era necesario el contar con un marco de trabajo que permita tener en cuenta cada una de las características propias de las aplicaciones SOA a través de la especificación SoaML, permitiendo al usuario y al desarrollador, tener una herramienta para realizar una migración de sus aplicaciones SOA a entornos Cloud.

Mediante el desarrollo del trabajo de investigación, se consideró también que las aplicaciones SOA puedan también ser código fuente, por lo que es necesario la utilización de una herramienta de ingeniera inversa, la misma que proporcione su modelo basándose en la especificación SoaML.

Otro aspecto a tener en consideración fue el Acuerdo de Nivel de Servicios (SLA), el mismo que es un aspecto clave en la prestación de servicios Cloud entre los consumidores y proveedores.

También, se ha tenido en cuenta el producto final, en el cual se ha dirigido a través del ejemplo de aplicación, y en el que se ve globalmente a la aplicación, su funcionalidad y su rendimiento/despliegue en un proveedor de servicios Cloud.

Según los objetivos planteados en el primer capítulo:

1. Analizar en profundidad los métodos y técnicas propuestos para migrar aplicaciones basadas en SOA hacia entornos Cloud Computing, con particular énfasis en aquellos que siguen el paradigma del desarrollo de software dirigido por modelos.
2. Analizar la estructura, componentes y servicios de las aplicaciones basadas en SOA y Cloud Computing que los investigadores y desarrolladores consideran relevantes para su desarrollo.

3. Definir un metamodelo para SOA, y un metamodelo para Cloud Computing respectivamente que recoja las características existentes de las aplicaciones de cada una, y cubra carencias encontradas en el estado del arte.
4. Definir un proceso genérico de transformaciones de modelos que dé cobertura a la migración de aplicaciones basadas en SOA a entornos Cloud Computing.
5. Evaluar el rendimiento y escalabilidad de las aplicaciones obtenidas a través del proceso de migración en su ejecución dentro de un entorno Cloud Computing. Tareas de Investigación

Se puede decir que todos, y cada uno de ellos han sido abordados y llevados a cabo con éxito:

Con respecto al primer objetivo, se realizó un mapeo sistemático, sobre las estrategias de migración de aplicaciones SOA a Cloud, a través del paradigma de desarrollo de software dirigido por modelos.

Tras este mapeo se llegó a la conclusión de que existen pocos trabajos que abordan la migración de aplicaciones SOA a entornos Cloud, haciendo uso del paradigma de desarrollo de software dirigido por modelos.

Con respecto al segundo objetivo, se analizaron las diversas estructuras, componentes y servicios de aplicaciones SOA y Cloud, a través de la especificación SoaML, y Cloud. Esto nos permitió conocer los diversos artefactos y componentes que se encuentran presentes en una aplicación SOA y Cloud.

Con respecto al tercer objetivo, se definieron dos principales metamodelos; el primero basado en la especificación SoaML de la OMG que fue nuestro metamodelo origen, y Cloud que fue nuestro metamodelo destino. Adicional a estos dos principales metamodelos, también se definieron dos metamodelos genéricos, el primero para representar el Acuerdo de Nivel de Servicios, y el segundo de la plataforma de servicios Cloud seleccionada Windows Azure.

Con respecto al cuarto objetivo, se definió un proceso genérico de transformaciones de modelos, donde los modelos de origen fueron SoaML y SLA, para la obtención de un modelo destino Cloud, al mismo que se le aplica una transformación modelo a texto para la obtención de la aplicación resultante y su posterior despliegue en la plataforma de servicios seleccionada.

Con respecto al quinto objetivo, se cumplió aplicando el marco de trabajo en la migración de un ejemplo de aplicación que muestra una aplicación modelada en SoaML, y a través de transformaciones de modelos es migrada y desplegada en una plataforma de servicios Cloud, para este caso Windows Azure.

Como se puede observar los objetivos de este trabajo de investigación fueron abarcados en su totalidad y vale la pena destacar que a nivel académico se ha logrado un claro entendimiento del tema, de cómo se puede contribuir al proceso de migración de aplicaciones SOA a entornos Cloud, y cómo este resulta una forma clara al estar definido con una notación conocida como es SPEM puede contribuir claramente y sin ambigüedades al proceso de migración de aplicaciones SOA, pero más allá de ello, se puede extender a aplicaciones de cualquier tipo, ayudando a desarrolladores y personas involucradas a obtener productos de calidad.

6.2. Trabajos Futuros

El trabajo presentado, constituye una primera aproximación de un marco de trabajo que permita la migración de aplicaciones SOA a Cloud, siguiendo una aproximación dirigida por modelos, este puede ser aplicado posteriormente a otros tipos de aplicaciones que según los avances tecnológicos puedan aparecer, como son aplicaciones SOA en diferentes superficies o dispositivos que tengan diferentes niveles de complejidad.

Se plantea también la necesidad de una contribución en la cual también sea posible migración de aplicaciones SOA a otros proveedores Cloud, como son Google App Engine y Amazon Web Services.

Otro aspecto necesario para realizarlo en un futuro es el refinamiento del modelo de Cloud mediante la evaluación de un conjunto más amplio de artefactos basándose en todas las plataformas existentes.

Se realizará en lo posterior una validación empírica del marco de trabajo propuesto a través de experimentos controlados, donde el marco de trabajo sea evaluado objetivamente acorde a su efectividad y eficiencia; y subjetivamente a su facilidad de uso y satisfacción.

Se ve también necesaria para futuros trabajos crear una herramienta específica en eclipse para dar soporte al marco de trabajo propuesto, y que pueda apoyar a la mayoría de proveedores de servicios Cloud. También es necesario indicar que en la herramienta se incluirán restricciones a través del lenguaje OCL para asegurar la calidad del modelado de las aplicaciones.

Otro trabajo futuro sería la propuesta de guías del proceso de migración desde su modelado hasta su despliegue en cualquiera plataforma de servicios Cloud.

6.3. Publicaciones

Durante el desarrollo de este trabajo de investigación de máster, se realizó una contribución en modo de publicación que fue sometida a un proceso de revisión por pares y fue aceptada en una conferencia internacional.

- Botto, M.; Insfrán, E.: “Are Model-Driven Techniques Used As A Means To Migrate SOA Applications To Cloud Computing?”, 10th International Conference on Web Information Systems and Technology, Barcelona, Spain (WEBIST 2014).

En este artículo presentó un mapeo sistemático sobre las estrategias de migración de aplicaciones SOA a Cloud, (Sección 2.2), a través del cual se definió el estado actual sobre la migración de aplicaciones SOA a Cloud.

Apéndice A: Bibliografía del Mapeo Sistemático

[S01] Amoretti M, Laghi MC, Tassoni F, Zanichelli F. Service migration within the cloud: Code mobility in SP2A, in 2010 International Conference on High Performance Computing & Simulation, 2010, 196–202, DOI:10.1109/HPCS.2010.5547130.

[S02] Andrikopoulos V, Binz T, Leymann F, Strauch S. How to adapt applications for the Cloud environment. Computing 2012; 95: 493–535, DOI: 10.1007/s00607-012-0248-2.

[S03] Azeemi IK, Lewis M, Tryfonas T. Migrating To The Cloud: Lessons And Limitations Of 'Traditional' IS Success Models. Procedia Comput. Sci. 2013; 16: 737–746, DOI:10.1016/j.procs.2013.01.077.

[S04] Babar MA, Chauhan MA. A tale of migration to cloud computing for sharing experiences and observations, in Proceeding of the 2nd international workshop on Software engineering for cloud computing - SECLOUD '11, 2011, 50, DOI:10.1145/1985500.1985509.

[S05] Beserra P V., Camara A, Ximenes R, Albuquerque AB, Mendonca NC. Cloudstep: A step-by-step decision process to support legacy application migration to the cloud, in 2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012, 7–16, DOI:10.1109/MESOCA.2012.6392602.

[S06] Biro L, Bacu V, Rodila D, Barabas L, Gorgan D. Grid to cloud migration of scientific applications, using dynamically created cloud clusters, in 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing, 2012, 335–340, DOI:10.1109/ICCP.2012.6356210.

[S07] Cai B, Xu F, Ye F, Zhou W. Research and application of migrating legacy systems to the private cloud platform with cloudstack. Autom. Logist. (ICAL), 2012; 400–404.

[S08] Chauhan MA, Babar MA. Migrating Service-Oriented System to Cloud Computing: An Experience Report. 2011 IEEE 4th Int. Conf. Cloud Comput. 2011; 404–411, DOI:10.1109/CLOUD.2011.46.

[S09] Chauhan MA, Babar MA. Towards Process Support for Migrating Applications to Cloud Computing, in 2012 International Conference on Cloud and Service Computing, 2012, 80–87, DOI:10.1109/CSC.2012.20.

[S10] Chee Y-M, Zhou N, Meng FJ, Bagheri S, Zhong P. A Pattern-Based Approach to Cloud Transformation, in 2011 IEEE 4th International Conference on Cloud Computing, 2011, 388–395, DOI:10.1109/CLOUD.2011.86.

[S11] Chen Y, Shen Q, Sun P, Li Y, Chen Z, Qing S. Reliable Migration Module in Trusted Cloud Based on Security Level - Design and Implementation, in 2012 IEEE 26th

International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012, 2230–2236, DOI:10.1109/IPDPSW.2012.275.

[S12] Costa PJP Da, Cruz AMR Da. Migration to Windows Azure – Analysis and Comparison. Procedia Technol. 2012; 5: 93–102, DOI:10.1016/j.protcy.2012.09.011.

[S13] Fan C-T, Wang W-J, Chang Y-S. Agent-Based Service Migration Framework in Hybrid Cloud, in 2011 IEEE International Conference on High Performance Computing and Communications, 2011, 887–892, DOI:10.1109/HPCC.2011.127.

[S14] Gabner R, Schwefel H-P, Hummel KA, Haring G. Optimal Model-Based Policies for Component Migration of Mobile Cloud Services, in 2011 IEEE 10th International Symposium on Network Computing and Applications, 2011, 195–202, DOI:10.1109/NCA.2011.33.

[S15] Gerhards M, Sander V, Belloum A. About the flexible Migration of Workflow Tasks to Clouds Combining on- and off-premise Executions of Applications, in CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization, 2012, 82–87.

[S16] Gkatzikis L, Koutsopoulos I. Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems. IEEE Wirel. Commun. 2013; 20: 24–32, DOI:10.1109/MWC.2013.6549280.

[S17] Guillén J, Miranda J, Murillo JM, Canal C. A service-oriented framework for developing cross cloud migratable software. J. Syst. Softw. 2013; 86: 2294–2308, DOI:10.1016/j.jss.2012.12.033.

[S18] Guillén J, Miranda J, Murillo JM, Canal C. Developing migratable multicloud applications based on MDE and adaptation techniques, in Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies - NordiCloud '13, 2013, 30–37, DOI:10.1145/2513534.2513541.

[S19] Hajjat M, Sun X, Sung Y-WE, Maltz D, Rao S, Sripanidkulchai K, Tawarmalani M. Cloudward bound, in Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10, 2010, 243, DOI:10.1145/1851182.1851212.

[S20] Hao W, Yen I-L, Thuraisingham B. Dynamic Service and Data Migration in the Clouds, in 2009 33rd Annual IEEE International Computer Software and Applications Conference, 2009, 134–139, DOI:10.1109/COMPSAC.2009.127.

[S21] Hung S-H, Shih C-S, Shieh J-P, Lee C-P, Huang Y-H. An Online Migration Environment for Executing Mobile Applications on the Cloud, in 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2011, 20–27, DOI:10.1109/IMIS.2011.77.

[S22] Jiang L, Cao J, Li P, Zhu Q. A Mixed Multi-tenancy Data Model and Its Migration Approach for the SaaS Application, in 2012 IEEE Asia-Pacific Services Computing Conference, 2012, 295–300, DOI:10.1109/APSCC.2012.16.

[S23] Juan-Verdejo A, Baars H. Decision support for partially moving applications to the cloud, in Proceedings of the 2013 international workshop on Hot topics in cloud services - HotTopiCS '13, 2013, 35, DOI:10.1145/2462307.2462316.

[S24] Kaisler S, Money WH. Service Migration in a Cloud Architecture, in 2011 44th Hawaii International Conference on System Sciences, 2011, 1–10, DOI:10.1109/HICSS.2011.371.

[S25] Kempf J, Johansson B, Pettersson S, Luning H, Nilsson T. Moving the mobile Evolved Packet Core to the cloud, in 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2012, 784–791, DOI:10.1109/WiMOB.2012.6379165.

[S26] Khajeh-Hosseini A, Greenwood D, Sommerville I. Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS, in 2010 IEEE 3rd International Conference on Cloud Computing, 2010, 450–457, DOI:10.1109/CLOUD.2010.37.

[S27] Khajeh-Hosseini A, Sommerville I, Bogaerts J, Teregowda P. Decision Support Tools for Cloud Migration in the Enterprise, in 2011 IEEE 4th International Conference on Cloud Computing, 2011, 541–548, DOI:10.1109/CLOUD.2011.59.

[S28] King TM, Ganti AS. Migrating Autonomic Self-Testing to the Cloud, in 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, 2010, 438–443, DOI:10.1109/ICSTW.2010.66.

[S29] Lamberti F, Sanna A, Demartini C. How to move your own applications into the cloud by exploiting interfaces automation and accessibility features, in 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, 368–372, DOI:10.1109/CCIS.2011.6045092.

[S30] Li J, Jia Y, Liu L, Wo T. CyberLiveApp: A secure sharing and migration approach for live virtual desktop applications in a cloud environment. Futur. Gener. Comput. Syst. 2013; 29: 330–340, DOI:10.1016/j.future.2011.08.001.

[S31] Lloyd W, Pallickara S, David O, Lyon J, Arabi M, Rojas K. Migration of Multi-tier Applications to Infrastructure-as-a-Service Clouds: An Investigation Using Kernel-Based Virtual Machines, in 2011 IEEE/ACM 12th International Conference on Grid Computing, 2011, 137–144, DOI:10.1109/Grid.2011.26.

[S32] Menzel M, Ranjan R. CloudGenius: decision support for web server cloud migration. Proc. 21st Int. Conf. … 2012; 979, DOI:10.1145/2187836.2187967.

[S33] Mohagheghi P, Sæther T. Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project, in 2011 IEEE World Congress on Services, 2011, 507–514, DOI:10.1109/SERVICES.2011.26.

[S33] Nguyen D, Thoai N. EBC: Application-level migration on multi-site cloud, in 2012 International Conference on Systems and Informatics (ICSAI2012), 2012, 876–880, DOI:10.1109/ICSAI.2012.6223147.

[S34] Nussbaumer N, Liu X. Cloud Migration for SMEs in a Service Oriented Approach, in 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops, 2013, 457–462, DOI:10.1109/COMPSACW.2013.71.

[S35] Pfitzmann B, Joukov N. Migration to Multi-image Cloud Templates, in 2011 IEEE International Conference on Services Computing, 2011, 80–87, DOI:10.1109/SCC.2011.89.

[S36] Qin X, Zhang W, Wang W, Wei J, Zhao X, Huang T. Optimizing data migration for cloud-based key-value stores, in Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12, 2012, 2204, DOI:10.1145/2396761.2398602.

[S37] Qiu W, Zheng Z, Wang X, Yang X, Lyu MR. Reliability-Based Design Optimization for Cloud Migration. IEEE Trans. Serv. Comput. 2013; 1–1, DOI:10.1109/TSC.2013.38.

[S38] Song J, Han F, Yan Z, Liu G, Zhu Z. A SaaSify Tool for Converting Traditional Web-Based Applications to SaaS Application, in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, 396–403, DOI:10.1109/CLOUD.2011.34.

[S39] Suen C-H, Kirchberg M, Lee BS. Efficient Migration of Virtual Machines between Public and Private Cloud, in 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011, 549–553, DOI:10.1109/CloudCom.2011.83.

[S40] Tak B, Tang C. PseudoApp: Performance prediction for application migration to cloud. … Netw. Manag. (IM … 2013; 303–310,.

[S41] Tankovic N. Model Driven Development Approaches: Comparison and Opportunities. 161.53.72.120.

[S42] Tran V, Keung J, Liu A, Fekete A. Application migration to cloud, in Proceeding of the 2nd international workshop on Software engineering for cloud computing - SECLOUD '11, 2011, 22, DOI:10.1145/1985500.1985505.

[S43] Vashishtha H, Smit M, Stroulia E. Moving Text Analysis Tools to the Cloud, in 2010 6th World Congress on Services, 2010, 107–114, DOI:10.1109/SERVICES.2010.91.

[S44] Venugopal S, Desikan S, Ganesan K. Effective Migration of Enterprise Applications in Multicore Cloud, in 2011 Fourth IEEE International Conference on Utility and Cloud Computing, 2011, 463–468, DOI:10.1109/UCC.2011.76.

[S45] Vu QH, Asal R. Legacy Application Migration to the Cloud: Practicability and Methodology, in 2012 IEEE Eighth World Congress on Services, 2012, 270–277, DOI:10.1109/SERVICES.2012.47.

[S46] Wu W-W. Developing an explorative model for SaaS adoption. Expert Syst. Appl. 2011; 38: 15057–15064, DOI:10.1016/j.eswa.2011.05.039.

[S47] Zhang Q, Lin Y, Wang Z. Cost-effective capacity migration of Peer-to-Peer social media to clouds. Peer-to-Peer Netw. Appl. 2012; 6: 247–256, DOI: 10.1007/s12083-012-0148-4.

[S48] Zhou L. CloudFTP: A Case Study of Migrating Traditional Applications to the Cloud, in 2013 Third International Conference on Intelligent System Design and Engineering Applications, 2013, 436–440, DOI:10.1109/ISDEA.2012.108.

Apéndice B: Atributos de Calidad de Servicio de SOA

Calidad de ejecución	
Disponibilidad	La ausencia de los tiempos de paradas de servicio. La disponibilidad es la probabilidad de que el servicio está disponible. Cuanto mayor sea el valor, más accesible es el servicio. Un valor TTR (tiempo de reparación) se representa generalmente con disponibilidad. Representa el tiempo que tarda para reparar un servicio roto.
Accesibilidad	El grado en que una petición de servicio es servida. Es posible que la probabilidad describa el cambio de utilización de los servicios. Un alto grado de accesibilidad significa que los clientes pueden utilizar el servicio con facilidad.
Integridad	El grado con que un servicio Web realiza sus tareas según su WSDL y Acuerdo de Nivel de Servicio. Cuanto mayor es la integridad, más cercana es la funcionalidad del servicio a su WSDL o SLA.
Rendimiento	El rendimiento se mide en términos de procesamiento y latencia. El rendimiento es el número de solicitudes de servicio que se sirven en un período de tiempo dado. La latencia es el tiempo empleado en el envío de la solicitud y la recepción de la respuesta. Alto rendimiento y baja latencia significan un buen desempeño del servicio.
Fiabilidad	La capacidad de un servicio para funcionar de forma correcta y sistemática. Mide la capacidad de operación del servicio a ser ejecutado dentro del plazo máximo previsto. Un servicio fiable proporciona la misma calidad de servicio a pesar de fallos del sistema o de la red. La confiabilidad de un servicio Web es representada generalmente como un número de fallos de transacciones en un mes o año.
Escalabilidad	La capacidad del servicio para atender las peticiones a pesar de las variaciones en el volumen de las solicitudes. Un alto grado de accesibilidad se puede lograr mediante la creación servicios escalables.
Seguridad	Cuestiones tales como la autenticación, autorización, integridad de mensajes y confidencialidad se incluyen en la seguridad.

	La seguridad es importante, porque los servicios web operan a través de Internet. El nivel deseado de seguridad para el servicio que se utiliza se define en el SLA. El proveedor de servicios debe mantener este nivel de seguridad.
Transaccionabilidad	A veces los servicios Web necesitan comportamiento transaccional. Si un determinado servicio requiere este comportamiento, se describe en el SLA.
Calidad de Negocios	
Costo	Las medidas de las unidades de dinero que el cliente de servicios tiene que pagar por el uso del servicio.
Reputación	Los clientes de los servicios pueden evaluar el servicio después de utilizarlo.
Regulatoria	Mide la conformidad de reglas, leyes y cumplimiento de los servicios Web con las normas y contratos establecidos en el acuerdo de nivel de servicio. Es importante conocer los solicitantes del servicio, la versión de la norma específica que está conformando el servicio (por ejemplo, WSDL versión 2.0). Los proveedores de los servicios deben cumplir con las normas especificadas en el acuerdo de nivel de servicio (SLA) entre los proveedores de servicio y solicitantes.

Apéndice C: Los posibles contenidos de un SLA.

Propósito	Las razones por las que el SLA se ha creado.
Partes	Partes que intervienen en el SLA y sus funciones de estas partes (proveedor de servicios, consumidor de servicios o el cliente).
Periodo de validez	El período de tiempo cuando el SLA es válido.
Ámbito	Define los servicios que están cubiertos en el acuerdo.
Restricciones	Define los pasos necesarios que deben adoptarse con el fin de proporcionar los niveles de servicio solicitados.
Objetivos de nivel de servicio	Los niveles de los objetivos que tanto el proveedor y el cliente estén de acuerdo. Incluyen generalmente un conjunto de indicadores de nivel de servicio, como la disponibilidad, el rendimiento y la fiabilidad que cada uno tiene un nivel objetivo que se pretende alcanzar.
Multas	Define las acciones que se toman si fallaron los objetivos definidos en el SLA.
Servicios opcionales	Especifica los servicios que podrían ser necesarios en caso de una falla.
Términos de exclusión	Especifica lo que ahora se trata en el SLA.
Administración	Define los procesos y objetivos medibles en un SLA y define la autoridad para la organización de la supervisión de ellos.

Apéndice D: Transformaciones entre Modelos

```
-- @nsURI Cloud=http://Cloud.ecore
-- @nsURI SoaML=http://SoaML.ecore
-- @nsURI SLA=http://sla.ecore

module soaml2cloud;
create OUT: Cloud from IN: SoaML, IN1: SLA;

-- SoaMLModel To CloudApplication
-- In this part was mandatory to include SLA Elements.
rule SoaMLModelToCloudApplication {
    from
        sm: SoaML!SoaMLModel,
        sl: SLA!SLAModel
    to
        ca: Cloud!CloudApplication (
            clientDependency <- sm.clientDependency,
            name <- sm.name.toString(),
            templateParameter <- sm.templateParameter,
            URI <- sm.URI,
            visibility <- sm.visibility,
            packagedElement <- sm.packagedElement,
            packagedElement <- sl.packagedElement
        )
}

-- Participant To CloudTask
rule ParticipantToCloudTask {
    from
        pa : SoaML!Participant
    to
        ca : Cloud!CloudTask (
            packagedElement <- pa.packagedElement,
            classifierBehavior <- pa.classifierBehavior,
            clientDependency <- pa.clientDependency,
            extension <- pa.extension,
            isAbstract <- pa.isAbstract,
            isActive <- pa.isActive,
            isFinalSpecialization <- pa.isFinalSpecialization,
            isIndirectlyInstantiated <- pa.isIndirectlyInstantiated,
            isLeaf <- pa.isLeaf,
            name <- pa.name.toString(),
            powertypeExtent <- pa.powertypeExtent,
            redefinedClassifier <- pa.redefinedClassifier,
            representation <- pa.representation,
            templateParameter <- pa.templateParameter,
            useCase <- pa.useCase,
            visibility <- pa.visibility,
            ownedAttribute <- pa.ownedAttribute,
            ownedOperation <- pa.ownedOperation
        )
}

-- Consumer To ServiceTask
rule ConsumerToServiceTask {
    from
        co : SoaML!Consumer
    to
```

```
        td : Cloud!ServiceTask (
            cloudApplication <- co.soamlModel,
            classifierBehavior <- co.classifierBehavior,
            clientDependency <- co.clientDependency,
            extension <- co.extension,
            isAbstract <- co.isAbstract,
            isActive <- co.isActive,
            isFinalSpecialization <- co.isFinalSpecialization,
            isLeaf <- co.isLeaf,
            name <- co.name.toString(),
            powertypeExtent <- co.powertypeExtent,
            redefinedClassifier <- co.redefinedClassifier,
            representation <- co.representation,
            templateParameter <- co.templateParameter,
            useCase <- co.useCase,
            visibility <- co.visibility,
            ownedAttribute <- co.ownedAttribute,
            ownedOperation <- co.ownedOperation,
            ownedReception <- co.ownedReception
        )
}

-- Provider To ServiceTask
rule ProviderToServiceTask {
    from
        pr : SoaML!Provider
    to
        st : Cloud!ServiceTask(
            classifierBehavior <- pr.classifierBehavior,
            clientDependency <- pr.clientDependency,
            extension <- pr.extension,
            isAbstract <- pr.isAbstract,
            isActive <- pr.isActive,
            isFinalSpecialization <- pr.isFinalSpecialization,
            isLeaf <- pr.isLeaf,
            name <- pr.name.toString(),
            powertypeExtent <- pr.powertypeExtent,
            redefinedClassifier <- pr.redefinedClassifier,
            representation <- pr.representation,
            templateParameter <- pr.templateParameter,
            useCase <- pr.useCase,
            visibility <- pr.visibility,
            ownedAttribute <- pr.ownedAttribute,
            ownedOperation <- pr.ownedOperation
        )
}


-- ServiceInterface To ServiceTask
rule ServiceInterfaceToServiceTask {
    from
        si : SoaML!ServiceInterface
    to
        td : Cloud!ServiceTask (
            classifierBehavior <- si.classifierBehavior,
            clientDependency <- si.clientDependency,
            extension <- si.extension,
            isAbstract <- si.isAbstract,
            isActive <- si.isActive,
```

```
            isFinalSpecialization <- si.isFinalSpecialization,
            isLeaf <- si.isLeaf,
            name <- si.name.toString(),
            powertypeExtent <- si.powertypeExtent,
            redefinedClassifier <- si.redefinedClassifier,
            representation <- si.representation,
            templateParameter <- si.templateParameter,
            useCase <- si.useCase,
            visibility <- si.visibility,
            ownedAttribute <- si.ownedAttribute,
            ownedOperation <- si.ownedOperation

        )
}

-- MessageType To Queue
rule MessageTypeToQueue {
    from
        mt : SoaML!MessageType
    to
        ct : Cloud!Queue (
            clientDependency <- mt.clientDependency,
            isAbstract <- mt.isAbstract,
            isFinalSpecialization <- mt.isFinalSpecialization,
            isLeaf <- mt.isLeaf,
            name <- mt.name,
            powertypeExtent <- mt.powertypeExtent,
            redefinedClassifier <- mt.redefinedClassifier,
            representation <- mt.representation,
            templateParameter <- mt.templateParameter,
            useCase <- mt.useCase,
            visibility <- mt.visibility,
            ownedAttribute <- mt.ownedAttribute
        )
}


-- SoaMLAssociationToCloudAssociation
rule Association {
    from
        s_as : SoaML!Association
    to
        c_as : Cloud!Association (
            clientDependency <- s_as.clientDependency,
            isAbstract <- s_as.isAbstract,
            isDerived <- s_as.isDerived,
            isFinalSpecialization <- s_as.isFinalSpecialization,
            isLeaf <- s_as.isLeaf,
            memberEnd <- s_as.memberEnd,
            name <- s_as.name.toString(),
            navigableOwnedEnd <- s_as.navigableOwnedEnd,
            powertypeExtent <- s_as.powertypeExtent,
            redefinedClassifier <- s_as.redefinedClassifier,
            representation <- s_as.representation,
            templateParameter <- s_as.templateParameter,
            useCase <- s_as.useCase,
            visibility <- s_as.visibility,
            ownedEnd <- s_as.ownedEnd
```

```
        )
}


-- SoaMLDataTypeToCloudDataType
rule DataType {
    from
        s_dt : SoaML!DataType
    to
        c_dt : Cloud!DataType (
            clientDependency <- s_dt.clientDependency,
            isAbstract <- s_dt.isAbstract,
            isFinalSpecialization <- s_dt.isFinalSpecialization,
            isLeaf <- s_dt.isLeaf,
            name <- s_dt.name.toString(),
            powertypeExtent <- s_dt.powertypeExtent,
            redefinedClassifier <- s_dt.redefinedClassifier,
            representation <- s_dt.representation,
            templateParameter <- s_dt.templateParameter,
            useCase <- s_dt.useCase,
            visibility <- s_dt.visibility
        )
}

-- SoaMLOperationToCloudOperation
rule Operation {
    from
        s_op : SoaML!Operation
    to
        c_op : Cloud!Operation (
            --bodyCondition <- s_op.bodyCondition,
            clientDependency <- s_op.clientDependency,
            concurrency <- s_op.concurrency,
            isAbstract <- s_op.isAbstract,
            isLeaf <- s_op.isLeaf,
            --isOrdered <- s_op.isOrdered,
            --isQuery <- s_op.isQuery,
            --isStatic <- s_op.isStatic,
            --isUnique <- s_op.isUnique,
            --lower <- s_op.lower,
            --method <- s_op.method,
            name <- s_op.name.toString(),
            --postcondition <- s_op.postcondition,
            --precondition <- s_op.precondition,
            --raisedException <- s_op.raisedException,
            redefinedOperation <- s_op.redefinedOperation,
            templateParameter <- s_op.templateParameter,
            --type <- s_op.type,
            --upper <- s_op.upper
            visibility <- s_op.visibility,
            ownedParameter <- s_op.ownedParameter
        )
}

-- SoaMLReceptionToCloudOperation
rule Reception {
    from
        s_re : SoaML!Reception
    to
```

```
        c_re : Cloud!Reception (
            clientDependency <- s_re.clientDependency,
            concurrency <- s_re.concurrency,
            isAbstract <- s_re.isAbstract,
            isLeaf <- s_re.isLeaf,
            isStatic <- s_re.isStatic,
            method <- s_re.method,
            name <- s_re.name.toString(),
            raisedException <- s_re.raisedException,
            signal <- s_re.raisedException,
            visibility <- s_re.visibility,
            ownedParameter <- s_re.ownedParameter
        )
}

-- SoaMLParameterToCloudParameter
rule Parameter {
    from
        s_pa : SoaML!Parameter
    to
        c_pa : Cloud!Parameter (
            clientDependency <- s_pa.clientDependency,
            default <- s_pa.default,
            direction <- s_pa.direction,
            effect <- s_pa.effect,
            isException <- s_pa.isException,
            isOrdered <- s_pa.isOrdered,
            isStream <- s_pa.isStream,
            isUnique <- s_pa.isUnique,
            lower <- s_pa.lower,
            name <- s_pa.name.toString(),
            operation <- s_pa.operation,
            parameterSet <- s_pa.parameterSet,
            templateParameter <- s_pa.templateParameter,
            type <- s_pa.type,
            upper <- s_pa.upper,
            visibility <- s_pa.visibility
        )
}

-- SoaMLPropertyToCloudProperty
rule Property {
    from
        s_pr : SoaML!Property
    to
        c_pr : Cloud!Property (
            aggregation <- s_pr.aggregation,
            association <- s_pr.association,
            clientDependency <- s_pr.clientDependency,
            default <- s_pr.default,
            isDerived <- s_pr.isDerived,
            isDerivedUnion <- s_pr.isDerivedUnion,
            isID <- s_pr.isID,
            isLeaf <- s_pr.isLeaf,
            isOrdered <- s_pr.isOrdered,
            isReadOnly <- s_pr.isReadOnly,
            isStatic <- s_pr.isStatic,
            isUnique <- s_pr.isUnique,
            lower <- s_pr.lower,
```

```
            name <- s_pr.name.toString(),
            redefinedProperty <- s_pr.redefinedProperty,
            subsettedProperty <- s_pr.subsettedProperty,
            templateParameter <- s_pr.templateParameter,
            type <- s_pr.type,
            upper <- s_pr.upper,
            visibility <- s_pr.visibility
        )
}

-- 2nd Part SLA To Cloud
-- Agreements To ConfigurationData
rule AgreementsToConfigurationData {
    from
        ag: SLA!Agreement
    to
        cd: Cloud!ConfigurationData (
            clientDependency <- ag.clientDependency,
            isAbstract <- ag.isAbstract,
            isFinalSpecialization <- ag.isFinalSpecialization,
            isLeaf <- ag.isLeaf,
            name <- ag.name.toString(),
            powertypeExtent <- ag.powertypeExtent,
            redefinedClassifier <- ag.redefinedClassifier,
            representation <- ag.representation,
            templateParameter <- ag.templateParameter,
            useCase <- ag.useCase,
            visibility <- ag.visibility
        )
}

-- KPIsToDataInjectionPort
rule KPIsToDataInjectionPort {
    from
        kp: SLA!KPI
    to
        di: Cloud!DataInjectionPort (
            clientDependency <- kp.clientDependency,
            isAbstract <- kp.isAbstract,
            isFinalSpecialization <- kp.isFinalSpecialization,
            isLeaf <- kp.isLeaf,
            name <- kp.name.toString(),
            powertypeExtent <- kp.powertypeExtent,
            redefinedClassifier <- kp.redefinedClassifier,
            representation <- kp.representation,
            templateParameter <- kp.templateParameter,
            useCase <- kp.useCase,
            visibility <- kp.visibility
        )
}

-- @nsURI Azure=htt://WindowsAzure.ecore
-- @nsURI Cloud=http://Cloud.ecore

module cloud2azure;
create OUT: Azure from IN: Cloud;

-- CloudApplication To CloudServices
rule CloudApplicationToCloudServices {
```

```
    from
        ca: Cloud!CloudApplication,
        cd: Cloud!ConfigurationData
    to
        wa: Azure!CloudService (
            clientDependency <- ca.clientDependency,
            name <- ca.name.toString(),
            templateParameter <- ca.templateParameter,
            URI <- ca.URI,
            visibility <- ca.visibility,
            packagedElement <- ca.packagedElement,
            packagedElement <- configCloud,
            packagedElement <- configLocal
        ),
        configCloud: Azure!ServiceConfigurationCloud (
            owner <- cd.owner,
            clientDependency <- cd.clientDependency,
            isAbstract <- cd.isAbstract,
            isFinalSpecialization <- cd.isFinalSpecialization,
            isLeaf <- cd.isLeaf,
            name <- cd.name.toString(),
            powertypeExtent <- cd.powertypeExtent,
            redefinedClassifier <- cd.redefinedClassifier,
            representation <- cd.representation,
            templateParameter <- cd.templateParameter,
            useCase <- cd.useCase,
            visibility <- cd.visibility,
            ownedAttribute <- cd.ownedAttribute,
            ownedOperation <- cd.ownedOperation
        ),
        configLocal: Azure!ServiceConfigurationLocal (
            clientDependency <- cd.clientDependency,
            isAbstract <- cd.isAbstract,
            isFinalSpecialization <- cd.isFinalSpecialization,
            isLeaf <- cd.isLeaf,
            name <- cd.name.toString(),
            powertypeExtent <- cd.powertypeExtent,
            redefinedClassifier <- cd.redefinedClassifier,
            representation <- cd.representation,
            templateParameter <- cd.templateParameter,
            useCase <- cd.useCase,
            visibility <- cd.visibility,
            ownedAttribute <- cd.ownedAttribute,
            ownedOperation <- cd.ownedOperation
        )
}

-- ServiceTask To ServiceWebRole
rule ServiceTaskToServiceWebRole {
    from
        st: Cloud!ServiceTask
    to
        wr: Azure!ServiceWebRole (
            cloudService <- st.cloudApplication,
            classifierBehavior <- st.classifierBehavior,
            clientDependency <- st.clientDependency,
            isAbstract <- st.isAbstract,
            isFinalSpecialization <- st.isFinalSpecialization,
            isLeaf <- st.isLeaf,
```

```
            name <- st.name + 'WebServiceRole'.toString(),
            powertypeExtent <- st.powertypeExtent,
            redefinedClassifier <- st.redefinedClassifier,
            representation <- st.representation,
            templateParameter <- st.templateParameter,
            useCase <- st.useCase,
            visibility <- st.visibility,
            ownedAttribute <- st.ownedAttribute,
            ownedOperation <- st.ownedOperation,
            interface <- Operation,
            service <- Service
        ),
        Operation: Azure!ServiceInterface (
            packagedElement <- st.packagedElement,
            classifierBehavior <- st.classifierBehavior,
            clientDependency <- st.clientDependency,
            extension <- st.extension,
            isAbstract <- st.isAbstract,
            isActive <- st.isActive,
            isFinalSpecialization <- st.isFinalSpecialization,
            isIndirectlyInstantiated <- st.isIndirectlyInstantiated,
            isLeaf <- st.isLeaf,
            name <- 'I' + st.name.toString(),
            powertypeExtent <- st.powertypeExtent,
            redefinedClassifier <- st.redefinedClassifier,
            representation <- st.representation,
            templateParameter <- st.templateParameter,
            useCase <- st.useCase,
            visibility <- st.visibility,
            ownedAttribute <- st.ownedAttribute,
            ownedOperation <- st.ownedOperation
        ),
        Service: Azure!WebService (
            packagedElement <- st.packagedElement,
            classifierBehavior <- st.classifierBehavior,
            clientDependency <- st.clientDependency,
            extension <- st.extension,
            isAbstract <- st.isAbstract,
            isActive <- st.isActive,
            isFinalSpecialization <- st.isFinalSpecialization,
            isIndirectlyInstantiated <- st.isIndirectlyInstantiated,
            isLeaf <- st.isLeaf,
            name <- st.name.toString(),
            powertypeExtent <- st.powertypeExtent,
            redefinedClassifier <- st.redefinedClassifier,
            representation <- st.representation,
            templateParameter <- st.templateParameter,
            useCase <- st.useCase,
            visibility <- st.visibility,
            ownedAttribute <- st.ownedAttribute,
            ownedOperation <- st.ownedOperation
        )
}

-- WebTask To ServiceWebRole
rule WebTaskToServiceWebRole {
    from
        wt: Cloud!WebTask
    to
```

```
wr: Azure!ServiceWebRole (
    cloudService <- wt.cloudApplication,
    packagedElement <- wt.packagedElement,
    classifierBehavior <- wt.classifierBehavior,
    clientDependency <- wt.clientDependency,
    extension <- wt.extension,
    isAbstract <- wt.isAbstract,
    isActive <- wt.isActive,
    isFinalSpecialization <- wt.isFinalSpecialization,
    isLeaf <- wt.isLeaf,
    name <- wt.name + 'WebServiceRole'.toString(),
    powertypeExtent <- wt.powertypeExtent,
    redefinedClassifier <- wt.redefinedClassifier,
    representation <- wt.representation,
    templateParameter <- wt.templateParameter,
    useCase <- wt.useCase,
    visibility <- wt.visibility,
    ownedAttribute <- wt.ownedAttribute,
    ownedOperation <- wt.ownedOperation,
    ownedReception <- wt.ownedReception,
    interface <- Operation,
    service <- Service
),
Operation: Azure!ServiceInterface (
    packagedElement <- wt.packagedElement,
    classifierBehavior <- wt.classifierBehavior,
    clientDependency <- wt.clientDependency,
    extension <- wt.extension,
    isAbstract <- wt.isAbstract,
    isActive <- wt.isActive,
    isFinalSpecialization <- wt.isFinalSpecialization,
    isIndirectlyInstantiated <- wt.isIndirectlyInstantiated,
    isLeaf <- wt.isLeaf,
    name <- 'I' + wt.name.toString(),
    powertypeExtent <- wt.powertypeExtent,
    redefinedClassifier <- wt.redefinedClassifier,
    representation <- wt.representation,
    templateParameter <- wt.templateParameter,
    useCase <- wt.useCase,
    visibility <- wt.visibility,
    ownedAttribute <- wt.ownedAttribute,
    ownedOperation <- wt.ownedOperation
),
Service: Azure!WebService (
    packagedElement <- wt.packagedElement,
    classifierBehavior <- wt.classifierBehavior,
    clientDependency <- wt.clientDependency,
    extension <- wt.extension,
    isAbstract <- wt.isAbstract,
    isActive <- wt.isActive,
    isFinalSpecialization <- wt.isFinalSpecialization,
    isIndirectlyInstantiated <- wt.isIndirectlyInstantiated,
    isLeaf <- wt.isLeaf,
    name <- wt.name.toString(),
    powertypeExtent <- wt.powertypeExtent,
    redefinedClassifier <- wt.redefinedClassifier,
    representation <- wt.representation,
    templateParameter <- wt.templateParameter,
    useCase <- wt.useCase,
```

```
            visibility <- wt.visibility,
            ownedAttribute <- wt.ownedAttribute,
            ownedOperation <- wt.ownedOperation
        )
}

-- CloudRotorTask To ServiceWorkerRole
rule CloudRotorTaskToServiceWorkerRole {
    from
        cr: Cloud!CloudRotorTask
    to
        wl: Azure!ServiceWorkRole (
            cloudService <- cr.cloudApplication,
            classifierBehavior <- cr.classifierBehavior,
            clientDependency <- cr.clientDependency,
            isAbstract <- cr.isAbstract,
            isFinalSpecialization <- cr.isFinalSpecialization,
            isLeaf <- cr.isLeaf,
            name <- cr.name + 'ServiceWorkerRole'.toString(),
            powertypeExtent <- cr.powertypeExtent,
            redefinedClassifier <- cr.redefinedClassifier,
            representation <- cr.representation,
            templateParameter <- cr.templateParameter,
            useCase <- cr.useCase,
            visibility <- cr.visibility,
            ownedAttribute <- cr.ownedAttribute,
            ownedOperation <- cr.ownedOperation
        )
}

-- Table To AzureTable
rule TableToAzureTable {
    from
        ct: Cloud!Table
    to
        at: Azure!Table (
            clientDependency <- ct.clientDependency,
            isAbstract <- ct.isAbstract,
            isFinalSpecialization <- ct.isFinalSpecialization,
            isLeaf <- ct.isLeaf,
            name <- ct.name.toString(),
            powertypeExtent <- ct.powertypeExtent,
            redefinedClassifier <- ct.redefinedClassifier,
            representation <- ct.representation,
            templateParameter <- ct.templateParameter,
            useCase <- ct.useCase,
            visibility <- ct.visibility,
            ownedAttribute <- ct.ownedAttribute
        )
}

-- Blob To AzureBlob
rule BlobToAzureBlob {
    from
        cb: Cloud!Blob
    to
        ab: Azure!Blob (
            clientDependency <- cb.clientDependency,
            isAbstract <- cb.isAbstract,
```

```
            isFinalSpecialization <- cb.isFinalSpecialization,
            isLeaf <- cb.isLeaf,
            name <- cb.name.toString(),
            powertypeExtent <- cb.powertypeExtent,
            redefinedClassifier <- cb.redefinedClassifier,
            representation <- cb.representation,
            templateParameter <- cb.templateParameter,
            useCase <- cb.useCase,
            visibility <- cb.visibility,
            ownedAttribute <- cb.ownedAttribute
        )
}

-- Queue To AzureQueue
rule QueueToAzureQueue {
    from
        cq: Cloud!Queue
    to
        aq: Azure!Queue (
            clientDependency <- cq.clientDependency,
            isAbstract <- cq.isAbstract,
            isFinalSpecialization <- cq.isFinalSpecialization,
            isLeaf <- cq.isLeaf,
            name <- cq.name.toString(),
            powertypeExtent <- cq.powertypeExtent,
            redefinedClassifier <- cq.redefinedClassifier,
            representation <- cq.representation,
            templateParameter <- cq.templateParameter,
            useCase <- cq.useCase,
            visibility <- cq.visibility,
            ownedAttribute <- cq.ownedAttribute
        )
}

-- CloudProtocoloHTTP To AzureProtocoloHTTP
rule ProtocoloHTTP {
    from
        c_ph: Cloud!HTTP
    to
        a_ph: Azure!HTTP (
            clientDependency <- c_ph.clientDependency,
            isAbstract <- c_ph.isAbstract,
            isFinalSpecialization <- c_ph.isFinalSpecialization,
            isLeaf <- c_ph.isLeaf,
            name <- c_ph.name.toString(),
            powertypeExtent <- c_ph.powertypeExtent,
            redefinedClassifier <- c_ph.redefinedClassifier,
            representation <- c_ph.representation,
            templateParameter <- c_ph.templateParameter,
            useCase <- c_ph.useCase,
            visibility <- c_ph.visibility,
            ownedAttribute <- c_ph.ownedAttribute
        )
}

-- CloudProtocoloHTTPS To AzureProtocoloHTTPS
rule ProtocoloHTTPS {
    from
        c_ps: Cloud!HTTPS
```

```
    to
        a_ps: Azure!HTTPS (
            clientDependency <- c_ps.clientDependency,
            isAbstract <- c_ps.isAbstract,
            isFinalSpecialization <- c_ps.isFinalSpecialization,
            isLeaf <- c_ps.isLeaf,
            name <- c_ps.name.toString(),
            powertypeExtent <- c_ps.powertypeExtent,
            redefinedClassifier <- c_ps.redefinedClassifier,
            representation <- c_ps.representation,
            templateParameter <- c_ps.templateParameter,
            useCase <- c_ps.useCase,
            visibility <- c_ps.visibility,
            ownedAttribute <- c_ps.ownedAttribute
        )
}

-- CloudProtocoloTCP To AzureProtocoloTCP
rule ProtocoloTCP {
    from
        c_pt: Cloud!TCP
    to
        a_pt: Azure!TCP (
            clientDependency <- c_pt.clientDependency,
            isAbstract <- c_pt.isAbstract,
            isFinalSpecialization <- c_pt.isFinalSpecialization,
            isLeaf <- c_pt.isLeaf,
            name <- c_pt.name.toString(),
            powertypeExtent <- c_pt.powertypeExtent,
            redefinedClassifier <- c_pt.redefinedClassifier,
            representation <- c_pt.representation,
            templateParameter <- c_pt.templateParameter,
            useCase <- c_pt.useCase,
            visibility <- c_pt.visibility,
            ownedAttribute <- c_pt.ownedAttribute
        )
}

-- CloudEndPoint To AzureEndPoint
rule EndPoint {
    from
        c_ep: Cloud!EndPoint
    to
        a_ep: Azure!EndPoint (
            name <- c_ep.name.toString()
        )
}

-- CloudAssociation To AzureAzociation
rule Association {
    from
        c_as: Cloud!Association
    to
        a_as: Azure!Association (
            clientDependency <- c_as.clientDependency,
            isAbstract <- c_as.isAbstract,
            isDerived <- c_as.isDerived,
            isFinalSpecialization <- c_as.isFinalSpecialization,
            isLeaf <- c_as.isLeaf,
```

```
            memberEnd <- c_as.memberEnd,
            name <- c_as.name.toString(),
            navigableOwnedEnd <- c_as.navigableOwnedEnd,
            powertypeExtent <- c_as.powertypeExtent,
            redefinedClassifier <- c_as.redefinedClassifier,
            representation <- c_as.representation,
            templateParameter <- c_as.templateParameter,
            useCase <- c_as.useCase,
            visibility <- c_as.visibility,
            ownedEnd <- c_as.ownedEnd
        )
}

-- CloudDataType To AzureDataType
rule DataType {
    from
        c_dt: Cloud!DataType
    to
        a_dt: Azure!DataType (
            clientDependency <- c_dt.clientDependency,
            isAbstract <- c_dt.isAbstract,
            isFinalSpecialization <- c_dt.isFinalSpecialization,
            isLeaf <- c_dt.isLeaf,
            name <- c_dt.name.toString(),
            powertypeExtent <- c_dt.powertypeExtent,
            redefinedClassifier <- c_dt.redefinedClassifier,
            representation <- c_dt.representation,
            templateParameter <- c_dt.templateParameter,
            useCase <- c_dt.useCase,
            visibility <- c_dt.visibility
        )
}

-- CloudOperation To AzureOperation
rule Operation {
    from
        c_op: Cloud!Operation
    to
        a_op: Azure!Operation (
            clientDependency <- c_op.clientDependency,
            concurrency <- c_op.concurrency,
            isAbstract <- c_op.isAbstract,
            isLeaf <- c_op.isLeaf,
            name <- c_op.name.toString(),
            redefinedOperation <- c_op.redefinedOperation,
            templateParameter <- c_op.templateParameter,
            visibility <- c_op.visibility,
            ownedParameter <- c_op.ownedParameter
        )
}

-- CloudReception To AzureReception
rule Reception {
    from
        c_re: Cloud!Reception
    to
        a_re: Azure!Reception (
            clientDependency <- c_re.clientDependency,
            concurrency <- c_re.concurrency,
```

```
            isAbstract <- c_re.isAbstract,
            isLeaf <- c_re.isLeaf,
            isStatic <- c_re.isStatic,
            method <- c_re.method,
            name <- c_re.name.toString(),
            raisedException <- c_re.raisedException,
            signal <- c_re.raisedException,
            visibility <- c_re.visibility,
            ownedParameter <- c_re.ownedParameter
        )
}

-- CloudParameter To AzureParameter
rule Parameter {
    from
        c_pa: Cloud!Parameter
    to
        a_pa: Azure!Parameter (
            clientDependency <- c_pa.clientDependency,
            default <- c_pa.default,
            direction <- c_pa.direction,
            effect <- c_pa.effect,
            isException <- c_pa.isException,
            isOrdered <- c_pa.isOrdered,
            isStream <- c_pa.isStream,
            isUnique <- c_pa.isUnique,
            lower <- c_pa.lower,
            name <- c_pa.name.toString(),
            operation <- c_pa.operation,
            parameterSet <- c_pa.parameterSet,
            templateParameter <- c_pa.templateParameter,
            type <- c_pa.type,
            upper <- c_pa.upper,
            visibility <- c_pa.visibility
        )
}

-- CloudProperty To AzureProperty
rule Property {
    from
        c_pr: Cloud!Property
    to
        a_pr: Azure!Property (
            aggregation <- c_pr.aggregation,
            association <- c_pr.association,
            clientDependency <- c_pr.clientDependency,
            default <- c_pr.default,
            isDerived <- c_pr.isDerived,
            isDerivedUnion <- c_pr.isDerivedUnion,
            isID <- c_pr.isID,
            isLeaf <- c_pr.isLeaf,
            isOrdered <- c_pr.isOrdered,
            isReadOnly <- c_pr.isReadOnly,
            isStatic <- c_pr.isStatic,
            isUnique <- c_pr.isUnique,
            lower <- c_pr.lower,
            name <- c_pr.name.toString(),
            redefinedProperty <- c_pr.redefinedProperty,
            subsettedProperty <- c_pr.subsettedProperty,
```

```
            templateParameter <- c_pr.templateParameter,
            type <- c_pr.type,
            upper <- c_pr.upper,
            visibility <- c_pr.visibility
        )
}
```

Apéndice E: Transformaciones de Modelo a Texto

```
[comment encoding = UTF-8 /]
[module generate('htt://WindowsAzure.ecore')]

[template public generateProject(aWindowsAzureApplication : CloudService)]
[comment @main/]
[file
(aWindowsAzureApplication.name.concat('/'.concat(aWindowsAzureApplication.nam
e.concat('.sln'))), false, 'UTF-8')]

Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio Express 2013 for Web
VisualStudioVersion = 12.0.21005.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{CC5FD16D-436D-48AD-A40C-5A424C6E3E79}") = "[name.toUpperFirst()/]",
"[name.toUpperFirst()/]\[name.toUpperFirst()/].ccproj", "{A8A4F897-71AE-4EBB-
9436-4A8FC377A86E}"
EndProject
[for (wr : ServiceWebRole | aWindowsAzureApplication.serviceWebRole)]
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") =
"[wr.name.toUpperFirst()/]",
"[wr.name.toUpperFirst()/]\[wr.name.toUpperFirst()/].ccproj", "{FAE04EC0-
301F-11D3-BF4B-00C04F79EFBC}"
EndProject
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") =
"[wl.name.toUpperFirst()/]",
"[wl.name.toUpperFirst()/]\[wl.name.toUpperFirst()/].ccproj", "{FAE04EC0-
301F-11D3-BF4B-00C04F79EFBC}"
EndProject
[/for]
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Any CPU = Debug|Any CPU
        Release|Any CPU = Release|Any CPU
    EndGlobalSection
    GlobalSection(ProjectConfigurationPlatforms) = postSolution
        {A8A4F897-71AE-4EBB-9436-4A8FC377A86E}.Debug|Any CPU.ActiveCfg =
Debug|Any CPU
        {A8A4F897-71AE-4EBB-9436-4A8FC377A86E}.Debug|Any CPU.Build.0 =
Debug|Any CPU
        {A8A4F897-71AE-4EBB-9436-4A8FC377A86E}.Release|Any CPU.ActiveCfg =
Release|Any CPU
        {A8A4F897-71AE-4EBB-9436-4A8FC377A86E}.Release|Any CPU.Build.0 =
Release|Any CPU
        {595387A5-5CE3-457A-97CB-14D4EB53FEAC}.Debug|Any CPU.ActiveCfg =
Debug|Any CPU
        {595387A5-5CE3-457A-97CB-14D4EB53FEAC}.Debug|Any CPU.Build.0 =
Debug|Any CPU
        {595387A5-5CE3-457A-97CB-14D4EB53FEAC}.Release|Any CPU.ActiveCfg =
Release|Any CPU
        {595387A5-5CE3-457A-97CB-14D4EB53FEAC}.Release|Any CPU.Build.0 =
Release|Any CPU
    EndGlobalSection
    GlobalSection(SolutionProperties) = preSolution
        HideSolutionNode = FALSE
    EndGlobalSection
```

```
EndGlobal
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat(aWindowsAzureApplication.nam
e.concat('/'.concat('ServiceConfiguration.Cloud.cscfg')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfiguration serviceName="[name.toUpperFirst()/]"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfigurati
on" osFamily="3" osVersion="*" schemaVersion="2013-10.2.2">
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
  <Role name="[wr.name.toUpperFirst()/]">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
  <Role name="[wl.name.toUpperFirst()/]">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
[/for]
</ServiceConfiguration>
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat(aWindowsAzureApplication.nam
e.concat('/'.concat('ServiceConfiguration.Local.cscfg')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfiguration serviceName="[name.toUpperFirst()/]"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfigurati
on" osFamily="3" osVersion="*" schemaVersion="2013-10.2.2">
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
  <Role name="[wr.name.toUpperFirst()/]">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
  <Role name="[wl.name.toUpperFirst()/]">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
```

```
[/for]
</ServiceConfiguration>
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat(aWindowsAzureApplication.nam
e.concat('/'.concat('ServiceDefinition.csdef')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="[name.toUpperFirst()/]"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition"
schemaVersion="2013-10.2.2">
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
  <WebRole name="[wr.name.toUpperFirst()/]" vmsize="Small">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <LocalResources>
      <LocalStorage name="[name.toUpperFirst()/].svclog" sizeInMB="1000"
cleanOnRoleRecycle="false" />
    </LocalResources>
  </WebRole>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
  <WorkerRole name="[wl.name.toUpperFirst()/]" vmsize="Small">
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
  </WorkerRole>
[/for]
</ServiceDefinition>
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat(aWindowsAzureApplication.nam
e.concat('/'.concat(aWindowsAzureApplication.name.concat('.ccproj'))))),
false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import
Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.pro
ps"
Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.
Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == ''
">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProductVersion>2.2</ProductVersion>
```

```
    <ProjectGuid>56137298-1fd3-4f23-ad77-3430cc137fff</ProjectGuid>
    <OutputType>Library</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>[name.toUpperFirst()/]</RootNamespace>
    <AssemblyName>[name.toUpperFirst()/]</AssemblyName>
    <StartDevelopmentStorage>True</StartDevelopmentStorage>
    <PackageEnableRemoteDebugger>False</PackageEnableRemoteDebugger>
    <Name>[name.toUpperFirst()/]</Name>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU'
">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' ==
'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <!-- Items for the project -->
  <ItemGroup>
    <ServiceDefinition Include="ServiceDefinition.csdef" />
    <ServiceConfiguration Include="ServiceConfiguration.Local.cscfg" />
    <ServiceConfiguration Include="ServiceConfiguration.Cloud.cscfg" />
  </ItemGroup>
  <ItemGroup>
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
    <ProjectReference
Include="..\[wr.name.toUpperFirst()/]\[wr.name.toUpperFirst()/].csproj">
      <Name>[wr.name.toUpperFirst()/]</Name>
      <Project>{ffbb9f19-8b30-487e-897c-093ce02f1a79}</Project>
      <Private>True</Private>
      <RoleType>Web</RoleType>
      <RoleName>[wr.name.toUpperFirst()/]</RoleName>

<UpdateDiagnosticsConnectionStringOnPublish>True</UpdateDiagnosticsConnection
StringOnPublish>
    </ProjectReference>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
    <ProjectReference
Include="..\[wl.name.toUpperFirst()/]\[wl.name.toUpperFirst()/].csproj">
      <Name>[wl.name.toUpperFirst()/]</Name>
      <Project>{3e02ac1a-e567-45e8-8a31-bb8fb9011916}</Project>
      <Private>True</Private>
      <RoleType>Worker</RoleType>
      <RoleName>[wl.name.toUpperFirst()/]</RoleName>

<UpdateDiagnosticsConnectionStringOnPublish>True</UpdateDiagnosticsConnection
StringOnPublish>
```

```
    </ProjectReference>
[/for]
  </ItemGroup>
  <ItemGroup>
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
    <Folder Include="[wr.name.toUpperFirst()/]Content\" />
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
    <Folder Include="[wl.name.toUpperFirst()/]Content\" />
[/for]
  </ItemGroup>
  <ItemGroup>
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
    <Content Include="[wr.name.toUpperFirst()/]Content\diagnostics.wadcfg">
      <SubType>Content</SubType>
    </Content>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
    <Content Include="[wl.name.toUpperFirst()/]Content\diagnostics.wadcfg">
      <SubType>Content</SubType>
    </Content>
[/for]
  </ItemGroup>
  <!-- Import the target files for this project template -->
  <PropertyGroup>
    <VisualStudioVersion Condition=" '$(VisualStudioVersion)' == ''
">10.0</VisualStudioVersion>
    <CloudExtensionsDir Condition=" '$(CloudExtensionsDir)' == ''
">$(MSBuildExtensionsPath)\Microsoft\VisualStudio\v$(VisualStudioVersion)\Win
dows Azure Tools\2.2\</CloudExtensionsDir>
  </PropertyGroup>
  <Import Project="$(CloudExtensionsDir)Microsoft.WindowsAzure.targets" />
</Project>
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat(aWindowsAzureApplication.nam
e.concat('/'.concat(aWindowsAzureApplication.name.concat('.ccproj.user'))))),
false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
</Project>
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat('packages/'.concat('reposito
ries.config')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<repositories>
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
  <repository path="..\[wr.name.toUpperFirst()/]\packages.config" />
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
  <repository path="..\[wl.name.toUpperFirst()/]\packages.config" />
[/for]
</repositories>
[/file]
```

```
[file
(aWindowsAzureApplication.name.concat('/'.concat('bin/'.concat('Debug/'.conca
t('ServiceConfiguration.cscfg')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<!--

**********************************************************************
****************
  Este archivo se generó con una herramienta del archivo de proyecto:
ServiceConfiguration.Local.cscfg

  Los cambios realizados en este archivo puede provocar un comportamiento
incorrecto y se perderán si el archivo se vuelve a generar.

**********************************************************************
****************
-->
<ServiceConfiguration serviceName="[name.toUpperFirst()/]"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfigurati
on" osFamily="3" osVersion="*" schemaVersion="2013-10.2.2">
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
  <Role name="[wr.name.toUpperFirst()/]">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
  <Role name="[wl.name.toUpperFirst()/]">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
[/for]
</ServiceConfiguration>
[/file]

[file
(aWindowsAzureApplication.name.concat('/'.concat('bin/'.concat('Debug/'.conca
t('ServiceDefinition.csdef')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<!--
**********************************************************************
****************
  Este archivo se generó con una herramienta del archivo de proyecto:
ServiceDefinition.csdef

  Los cambios realizados en este archivo puede provocar un comportamiento
incorrecto y se perderán si el archivo se vuelve a generar.

**********************************************************************
****************-->
```

```
<ServiceDefinition name="[name.toUpperFirst()/]"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition"
schemaVersion="2013-10.2.2">
[for (wr : ServiceWebRole| aWindowsAzureApplication.serviceWebRole)]
  <WebRole name="[wr.name.toUpperFirst()/]" vmsize="Small">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <LocalResources>
      <LocalStorage name="[name.toUpperFirst()/].svclog" sizeInMB="1000"
cleanOnRoleRecycle="false" />
    </LocalResources>
    <Contents>
      <Content destination=".\">
        <SourceDirectory path="" />
      </Content>
    </Contents>
  </WebRole>
[/for]
[for (wl : ServiceWorkRole| aWindowsAzureApplication.serviceWorkRole)]
  <WorkerRole name="[wl.name.toUpperFirst()/]" vmsize="Small">
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
  </WorkerRole>
[/for]
</ServiceDefinition>
[/file]
[/template]

[template public generateWebRole(aWebRole : ServiceWebRole)]
[comment @main/]
[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.cloudService.name.conc
at('/'.concat(name.concat('Content/'.concat('diagnostics.wadcfg')))))),
false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<DiagnosticMonitorConfiguration configurationChangePollInterval="PT1M"
overallQuotaInMB="4096"
xmlns="http://schemas.microsoft.com/ServiceHosting/2010/10/DiagnosticsConfigu
ration">
  <DiagnosticInfrastructureLogs />
  <Directories>
    <IISLogs container="wad-iis-logfiles" />
    <CrashDumps container="wad-crash-dumps" />
  </Directories>
  <Logs bufferQuotaInMB="1024" scheduledTransferPeriod="PT1M"
scheduledTransferLogLevelFilter="Error" />
```

```
  <WindowsEventLog bufferQuotaInMB="1024" scheduledTransferPeriod="PT1M"
scheduledTransferLogLevelFilter="Error">
    <DataSource name="Application!*" />
  </WindowsEventLog>
  <PerformanceCounters bufferQuotaInMB="512" scheduledTransferPeriod="PT0M">
    <PerformanceCounterConfiguration counterSpecifier="\Memory\Available
MBytes" sampleRate="PT3M" />
    <PerformanceCounterConfiguration counterSpecifier="\Web
Service(_Total)\ISAPI Extension Requests/sec" sampleRate="PT3M"/>
    <PerformanceCounterConfiguration counterSpecifier="\Web
Service(_Total)\Bytes Total/Sec" sampleRate="PT3M"/>
    <PerformanceCounterConfiguration counterSpecifier="\ASP.NET
Applications(__Total__)\Requests/Sec" sampleRate="PT3M"/>
    <PerformanceCounterConfiguration counterSpecifier="\ASP.NET
Applications(__Total__)\Errors Total/Sec" sampleRate="PT3M"/>
    <PerformanceCounterConfiguration counterSpecifier="\ASP.NET\Requests
Queued" sampleRate="PT3M"/>
    <PerformanceCounterConfiguration counterSpecifier="\ASP.NET\Requests
Rejected" sampleRate="PT3M"/>
  </PerformanceCounters>
</DiagnosticMonitorConfiguration>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('Properties/'.concat('AssemblyInfo.cs'))))), false, 'UTF-8')]
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// La información general de un ensamblado se controla mediante el siguiente
// conjunto de atributos. Cambie estos atributos para modificar la
información
// asociada a un ensamblado.
['['/]assembly: AssemblyTitle("[name.toUpperFirst()/]")[']'/]
['['/]assembly: AssemblyDescription("")[']'/]
['['/]assembly: AssemblyConfiguration("")[']'/]
['['/]assembly: AssemblyCompany("Microsoft")[']'/]
['['/]assembly: AssemblyProduct("[name.toUpperFirst()/]")[']'/]
['['/]assembly: AssemblyCopyright("Copyright © Microsoft 2014")[']'/]
['['/]assembly: AssemblyTrademark("")[']'/]
['['/]assembly: AssemblyCulture("")[']'/]

// Si establece ComVisible en false, hace que los tipos de este ensamblado no
sean visibles
// a los componentes COM. Si necesita obtener acceso a un tipo en este
ensamblado desde
// COM, establezca el atributo ComVisible en true en este tipo.
['['/]assembly: ComVisible(false)[']'/]

// El siguiente GUID sirve como identificador de la biblioteca de tipos si
este proyecto se expone a COM
['['/]assembly: Guid("f83da3f0-6bc8-48fe-95bf-a8b5c4094fb0")[']'/]

// La información de versión de un ensamblado consta de los siguientes cuatro
valores:
//
//      Versión principal
//      Versión secundaria
```

```
//      Número de compilación
//      Revisión
//
// Puede especificar todos los valores o usar los valores predeterminados
(número de compilación y de revisión)
// usando el símbolo '*' como se muestra a continuación:
// ['['/]assembly: AssemblyVersion("1.0.*")[']'/]
['['/]assembly: AssemblyVersion("1.0.0.0")[']'/]
['['/]assembly: AssemblyFileVersion("1.0.0.0")[']'/]
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
(aWebRole.name.concat('.csproj'))))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import
Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.pro
ps"
Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.
Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == ''
">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProductVersion>
    </ProductVersion>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>{FFBB9F19-8B30-487E-897C-093CE02F1A79}</ProjectGuid>
    <ProjectTypeGuids>{349c5851-65df-11da-9384-00065b846f21};{fae04ec0-301f-
11d3-bf4b-00c04f79efbc}</ProjectTypeGuids>
    <OutputType>Library</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>[name.toUpperFirst()/]</RootNamespace>
    <AssemblyName>[name.toUpperFirst()/]</AssemblyName>
    <TargetFrameworkVersion>v4.5</TargetFrameworkVersion>
    <WcfConfigValidationEnabled>True</WcfConfigValidationEnabled>
    <UseIISExpress>true</UseIISExpress>
    <IISExpressSSLPort />
    <IISExpressAnonymousAuthentication />
    <IISExpressWindowsAuthentication />
    <IISExpressUseClassicPipelineMode />
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU'
">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' ==
'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\</OutputPath>
```

```
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="Microsoft.Data.Edm, Version=5.2.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL">

<HintPath>..\packages\Microsoft.Data.Edm.5.2.0\lib\net40\Microsoft.Data.Edm.d
ll</HintPath>
    </Reference>
    <Reference Include="Microsoft.Data.OData, Version=5.2.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35,
processorArchitecture=MSIL">

<HintPath>..\packages\Microsoft.Data.OData.5.2.0\lib\net40\Microsoft.Data.ODa
ta.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.WindowsAzure.Configuration,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35,
processorArchitecture=MSIL">

<HintPath>..\packages\Microsoft.WindowsAzure.ConfigurationManager.2.0.1.0\lib
\net40\Microsoft.WindowsAzure.Configuration.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.WindowsAzure.Diagnostics, Version=2.2.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35">
      <Private>True</Private>
    </Reference>
    <Reference Include="Microsoft.WindowsAzure.ServiceRuntime,
Version=2.2.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35">
      <Private>False</Private>
    </Reference>
    <Reference Include="Microsoft.WindowsAzure.Storage, Version=2.1.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35,
processorArchitecture=MSIL">

<HintPath>..\packages\WindowsAzure.Storage.2.1.0.0\lib\net40\Microsoft.Window
sAzure.Storage.dll</HintPath>
    </Reference>
    <Reference Include="System.Data.Services.Client" />
    <Reference Include="System.Spatial, Version=5.2.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL">

<HintPath>..\packages\System.Spatial.5.2.0\lib\net40\System.Spatial.dll</Hint
Path>
    </Reference>
    <Reference Include="System.Web.DynamicData" />
    <Reference Include="System.Web.Entity" />
    <Reference Include="System.Web.ApplicationServices" />
    <Reference Include="System" />
    <Reference Include="System.Configuration" />
    <Reference Include="System.Core" />
    <Reference Include="System.Data" />
    <Reference Include="System.Drawing" />
    <Reference Include="System.EnterpriseServices" />
    <Reference Include="System.Runtime.Serialization" />
    <Reference Include="System.ServiceModel" />
```

```
    <Reference Include="System.ServiceModel.Web" />
    <Reference Include="System.Web" />
    <Reference Include="System.Web.Extensions" />
    <Reference Include="System.Web.Services" />
    <Reference Include="System.Xml" />
    <Reference Include="System.Xml.Linq" />
  </ItemGroup>
  <ItemGroup>
    <Content Include="[aWebRole.name.toUpperFirst()/].svc" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="AzureLocalStorageTraceListener.cs" />
    <Compile Include="[aWebRole.name.toUpperFirst()/].svc.cs">
      <DependentUpon>[aWebRole.name.toUpperFirst()/].svc</DependentUpon>
    </Compile>
    <Compile Include="I[aWebRole.name.toUpperFirst()/].cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
    <Compile Include="WebRole.cs" />
  </ItemGroup>
  <ItemGroup>
    <Folder Include="App_Data\" />
  </ItemGroup>
  <ItemGroup>
    <Content Include="packages.config" />
  </ItemGroup>
  <ItemGroup>
    <Content Include="Web.config" />
    <None Include="Web.Debug.config">
      <DependentUpon>Web.config</DependentUpon>
    </None>
    <None Include="Web.Release.config">
      <DependentUpon>Web.config</DependentUpon>
    </None>
  </ItemGroup>
  <PropertyGroup>
    <VisualStudioVersion Condition="'$(VisualStudioVersion)' ==
''">10.0</VisualStudioVersion>
    <VSToolsPath Condition="'$(VSToolsPath)' ==
''">$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(VisualStudioVersion)
</VSToolsPath>
  </PropertyGroup>
  <Import Project="$(MSBuildBinPath)\Microsoft.CSharp.targets" />
  <Import
Project="$(VSToolsPath)\WebApplications\Microsoft.WebApplication.targets"
Condition="'$(VSToolsPath)' != ''" />
  <Import
Project="$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v10.0\WebApplicati
ons\Microsoft.WebApplication.targets" Condition="false" />
  <ProjectExtensions>
    <VisualStudio>
      <FlavorProperties GUID="{349c5851-65df-11da-9384-00065b846f21}">
        <WebProjectProperties>
          <UseIIS>True</UseIIS>
          <AutoAssignPort>True</AutoAssignPort>
          <DevelopmentServerPort>0</DevelopmentServerPort>
          <DevelopmentServerVPath>/</DevelopmentServerVPath>
          <IISUrl>http://localhost:16631/</IISUrl>
          <NTLMAuthentication>False</NTLMAuthentication>
          <UseCustomServer>False</UseCustomServer>
```

```
          <CustomServerUrl>
          </CustomServerUrl>
          <SaveServerSettingsInUserFile>False</SaveServerSettingsInUserFile>
        </WebProjectProperties>
      </FlavorProperties>
    </VisualStudio>
  </ProjectExtensions>
  <!-- To modify your build process, add your task inside one of the targets
below and uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('AzureLocalStorageTraceListener.cs')))), false, 'UTF-8')]
namespace [name.toUpperFirst()/]
{
    public class AzureLocalStorageTraceListener : XmlWriterTraceListener
    {
        public AzureLocalStorageTraceListener()
            :
base(Path.Combine(AzureLocalStorageTraceListener.GetLogDirectory().Path,
"[aWebRole.name.toUpperFirst()/].svclog"))
        {
        }

        public static DirectoryConfiguration GetLogDirectory()
        {
            DirectoryConfiguration directory = new DirectoryConfiguration();
            directory.Container = "wad-tracefiles";
            directory.DirectoryQuotaInMB = 10;
            directory.Path =
RoleEnvironment.GetLocalResource("[aWebRole.name.toUpperFirst()/].svclog").Ro
otPath;
            return directory;
        }
    }
}
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/I'.conca
t(aWebRole.name.concat('.cs'))))), false, 'UTF-8')]
namespace [name.toUpperFirst()/]
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para
cambiar el nombre de interfaz "I[aWebRole.interface.name.toUpperFirst()/]" en
el código y en el archivo de configuración a la vez.
    ['['/]ServiceContract[']'/]
    public interface [aWebRole.interface.name.toUpperFirst()/]
    {
        ['['/]OperationContract[']'/]
```

```
        [self.service.ownedOperation.ownedParameter.type.name/]
[self.service.ownedOperation.name/]();
         // TODO: agregue aquí sus operaciones de servicio
    }

}
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
(aWebRole.name.concat('.svc.cs'))))), false, 'UTF-8')]
namespace [name.toUpperFirst()/]
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para
cambiar el nombre de clase "Service1" en el código, en svc y en el archivo de
configuración.
    // NOTE: para iniciar el Cliente de prueba WCF para probar este servicio,
seleccione Service1.svc o Service1.svc.cs en el Explorador de soluciones e
inicie la depuración.
    public class [aWebRole.service.name.toUpperFirst()/] :
[aWebRole.interface.name.toUpperFirst()/]
    {
        public [self.service.ownedOperation.ownedParameter.type.name/]
[self.service.ownedOperation.name/]([self.service.ownedOperation.ownedParamet
er.type.name/] [self.service.ownedOperation.ownedParameter.name/])
        {
            return "[aWebRole.name/]:" +
[self.service.ownedOperation.ownedParameter.name/]
        }
    }
}
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('WebRole.cs')))), false, 'UTF-8')]
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace [name.toUpperFirst()/]
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // Para habilitar AzureLocalStorageTraceListner, quite las marcas
de comentario de la sección correspondiente en web.config.
            DiagnosticMonitorConfiguration diagnosticConfig =
DiagnosticMonitor.GetDefaultInitialConfiguration();
            diagnosticConfig.Directories.ScheduledTransferPeriod =
TimeSpan.FromMinutes(1);

diagnosticConfig.Directories.DataSources.Add(AzureLocalStorageTraceListener.G
etLogDirectory());
```

```
            // Para obtener información sobre cómo administrar los cambios de
configuración
            // consulte el tema de MSDN en
http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
(aWebRole.name.concat('.csproj.user'))))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ProjectExtensions>
    <VisualStudio>
      <FlavorProperties GUID="{349c5851-65df-11da-9384-00065b846f21}">
        <WebProjectProperties>
          <StartPageUrl>
          </StartPageUrl>
          <StartAction>CurrentPage</StartAction>
          <AspNetDebugging>True</AspNetDebugging>
          <SilverlightDebugging>False</SilverlightDebugging>
          <NativeDebugging>False</NativeDebugging>
          <SQLDebugging>False</SQLDebugging>
          <ExternalProgram>
          </ExternalProgram>
          <StartExternalURL>
          </StartExternalURL>
          <StartCmdLineArguments>
          </StartCmdLineArguments>
          <StartWorkingDirectory>
          </StartWorkingDirectory>
          <EnableENC>False</EnableENC>
          <AlwaysStartWebServerOnDebug>True</AlwaysStartWebServerOnDebug>
        </WebProjectProperties>
      </FlavorProperties>
    </VisualStudio>
  </ProjectExtensions>
</Project>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
(aWebRole.name.concat('.svc'))))), false, 'UTF-8')]
<%@ ServiceHost Language="C#" Debug="true"
Service="[name.toUpperFirst().concat('.').concat(aWebRole.name)/]"
CodeBehind="[name.toUpperFirst()/].svc.cs" %>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('packages.config'))))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Microsoft.Data.Edm" version="5.2.0" targetFramework="net45" />
```

```
  <package id="Microsoft.Data.OData" version="5.2.0" targetFramework="net45"
/>
  <package id="Microsoft.WindowsAzure.ConfigurationManager" version="2.0.1.0"
targetFramework="net45" />
  <package id="System.Spatial" version="5.2.0" targetFramework="net45" />
  <package id="WindowsAzure.Storage" version="2.1.0.0"
targetFramework="net45" />
</packages>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('Web.config'))))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <!--  Para recopilar seguimientos de diagnóstico, borre los comentarios de
la siguiente sección o combínela con la sección system.diagnostics existente.
        Para mantener los seguimientos en el almacenamiento, actualice la
configuración de DiagnosticsConnectionString con sus credenciales de
almacenamiento.
        Para evitar un deterioro del rendimiento, no olvide deshabilitar el
seguimiento en las implementaciones de producción.
  <system.diagnostics>
    <sharedListeners>
      <add name="AzureLocalStorage"
type="[name.toUpperFirst()/].AzureLocalStorageTraceListener,
[name.toUpperFirst()/]"/>
    </sharedListeners>
    <sources>
      <source name="System.ServiceModel" switchValue="Verbose,
ActivityTracing">
        <listeners>
          <add name="AzureLocalStorage"/>
        </listeners>
      </source>
      <source name="System.ServiceModel.MessageLogging"
switchValue="Verbose">
        <listeners>
          <add name="AzureLocalStorage"/>
        </listeners>
      </source>
    </sources>
   </system.diagnostics> -->
  <system.diagnostics>
    <trace>
      <listeners>
        <add
type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener,
Microsoft.WindowsAzure.Diagnostics, Version=2.2.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
          name="AzureDiagnostics">
          <filter type="" />
        </add>
      </listeners>
    </trace>
  </system.diagnostics>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
  </system.web>
```

```
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- Para evitar que la información de metadatos se haga pública,
establezca el valor siguiente en FALSE antes de la implementación -->
          <serviceMetadata httpGetEnabled="true"/>
          <!-- Para recibir detalles de las excepciones en los errores para
la depuración, establezca el siguiente valor en true. Para no revelar
información sobre las excepciones establézcalo en false antes de la
implementación -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true"/>
    <!--
        Para explorar el directorio raíz de aplicaciones web durante la
depuración, establezca el valor siguiente en TRUE.
        Establézcalo en FALSE antes de la implementación para evitar que la
información de la carpeta de aplicaciones web se haga pública.
    -->
    <directoryBrowse enabled="true"/>
  </system.webServer>
</configuration>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('Web.Debug.config')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>

<!-- Para obtener más información sobre el uso de la transformación de
web.config, visite http://go.microsoft.com/fwlink/?LinkId=125889 -->

<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-
Transform">
  <!--
    En el ejemplo siguiente, la transformación "SetAttributes" cambiará el
valor de    "connectionString" para que solamente use "ReleaseSQLServer"
cuando el localizador "Match"    encuentre un atributo "name" con el valor
"MyDB".    <connectionStrings>     <add name="MyDB".
connectionString="Data Source=ReleaseSQLServer;Initial
Catalog=MyReleaseDB;Integrated Security=True"
xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>
</connectionStrings>
  -->
  <system.web>
    <!--
      En el ejemplo siguiente, la transformación "Replace" reemplazará toda
la
      sección <customErrors> del archivo web.config.
      Tenga en cuenta que, como solo hay una sección customErrors bajo el
nodo
      <system.web>, no es necesario usar el atributo "xdt:Locator".
```

```
      <customErrors defaultRedirect="GenericError.htm"
        mode="RemoteOnly" xdt:Transform="Replace">
        <error statusCode="500" redirect="InternalError.htm"/>
      </customErrors>
    -->
  </system.web>
</configuration>
[/file]

[file
(aWebRole.cloudService.name.concat('/'.concat(aWebRole.name.concat('/'.concat
('Web.Release.config')))), false, 'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>

<!-- Para obtener más información sobre el uso de la transformación de
web.config, visite http://go.microsoft.com/fwlink/?LinkId=125889 -->

<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-
Transform">
  <!--
    En el ejemplo siguiente, la transformación "SetAttributes" cambiará el
valor de    "connectionString" para que solamente use "ReleaseSQLServer"
cuando el localizador "Match"    encuentre un atributo "name" con el valor
"MyDB".    <connectionStrings>     <add name="MyDB".
connectionString="Data Source=ReleaseSQLServer;Initial
Catalog=MyReleaseDB;Integrated Security=True"
xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>
</connectionStrings>
  -->
  <system.web>
    <compilation xdt:Transform="RemoveAttributes(debug)" />
    <!--
      En el ejemplo siguiente, la transformación "Replace" reemplazará toda
la
      sección <customErrors> del archivo web.config.
      Tenga en cuenta que, como solo hay una sección customErrors bajo el
nodo
      <system.web>, no es necesario usar el atributo "xdt:Locator".

      <customErrors defaultRedirect="GenericError.htm"
        mode="RemoteOnly" xdt:Transform="Replace">
        <error statusCode="500" redirect="InternalError.htm"/>
      </customErrors>
    -->
  </system.web>
</configuration>
[/file]

[/template]
```

Referencias

[1] Acens. ¿Qué es el SLA? [Online]. Available: https://www.acens.com/file_download/176/acens_que_es_el_sla_baja.pdf. [Accessed: 06-Feb-2014].

[2] Ali N, Nellipaiappan R, Chandran R, Babar MA. Model driven support for the Service Oriented Architecture modeling language, in *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems - PESOS '10*, 2010, 8, DOI:10.1145/1808885.1808888.

[3] Amazon. Amazon Elastic Compute Cloud (Amazon EC2), 2013. [Online]. Available: http://aws.amazon.com/es/ec2/. [Accessed: 08-Oct-2013].

[4] Amazon Web Services. What is Cloud Computing by Amazon Web Services. [Online]. Available: http://aws.amazon.com/what-is-cloud-computing/. [Accessed: 08-Oct-2013].

[5] Andrikopoulos V, Binz T, Leymann F, Strauch S. How to adapt applications for the Cloud environment. *Computing* 2012; **95**: 493–535, DOI:10.1007/s00607-012-0248-2.

[6] AppLabs. Testing the Cloud, 2009. [Online]. Available: http://www.qaguild.com/Document/app_whitepaper_testing_the_cloud_1v00.pdf. [Accessed: 21-Jan-2014].

[7] ATL. ATL - A model transformation language. [Online]. Available: http://www.eclipse.org/atl/. [Accessed: 06-Jan-2014].

[8] Barber S. SOA Testing Challenges, 2006. [Online]. Available: http://www.perftestplus.com/resources/SOA_challenges_ppt.pdf. [Accessed: 06-Feb-2014].

[9] Baresi L, Nitto E Di. *Test and Analysis of Web Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[10] Barry DK. Service-Oriented Architecture (SOA) Definition. [Online]. Available: http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html. [Accessed: 19-Sep-2013].

[11] Bézivin J. On the unification power of models. *Softw. Syst. Model.* 2005; **4**: 171–188, DOI:10.1007/s10270-005-0079-0.

[12] Bezivin J, Gerbe O. Towards a precise definition of the OMG/MDA framework, in *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, 2001, 273–280, DOI:10.1109/ASE.2001.989813.

[13] Brambilla M, Cabot J, Wimmer M. *Model-Driven Software Engineering in Practice*, 1: 2012, 1–182.

[14] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Futur. Gener. Comput. Syst.* 2009; 25: 599–616, DOI:10.1016/j.future.2008.12.001.

[15] Cagle K. SOA is Dead? It's About Time! - O'Reilly Broadcast, 2009. [Online]. Available: http://broadcast.oreilly.com/2009/01/soa-is-dead-its-about-time.html. [Accessed: 06-Oct-2013].

[16] Carvalho JFS, Neto PA da MS, Garcia VC, Assad RE, Durao F. A Systematic Mapping Study on Cloud Computing. *arXiv Prepr. arXiv* 2013; .

[17] Chappel D. Introducing the Windows Azure Platform, 2010. [Online]. Available: http://www.davidchappell.com/writing/white_papers/Introducing_the_Windows_Azure_Platform,_v1.4--Chappell.pdf. [Accessed: 08-Oct-2013].

[18] Chappel D. Introducing Windows Azure, 2009. [Online]. Available: http://www.davidchappell.com/writing/white_papers/Introducing_Windows_Azure_v1-Chappell.pdf. [Accessed: 08-Oct-2013].

[19] Chou D. Microsoft Cloud Computing Platform, 2010.

[20] CIOL. CIOL: Latest IT News | Latest Enterprise News | Latest SMB News | Hot Tech Topic news, 2008. [Online]. Available: http://www.ciol.com/ec/feature/saas-—-a-revolutionary-approach-for-building-web-applications/23108103050/0/. [Accessed: 08-Oct-2013].

[21] Cisco System. Planning the Migration of Enterprise Applications to the Cloud, *White Paper*, 1–9, 2010.

[22] Complutense University of Madrid. Distributed Systems Architecture Research Group. [Online]. Available: http://dsa-research.org/doku.php?id=start. [Accessed: 05-Jan-2014].

[23] Delgado A. MINERVA. [Online]. Available: http://alarcos.esi.uclm.es/MINERVA/. [Accessed: 25-Jan-2014].

[24] Dubey A, Wagle D. DeliveringSWasaService.pdf, 2007. [Online]. Available: http://ai.kaist.ac.kr/~jkim/cs489-2007/Resources/DeliveringSWasaService.pdf. [Accessed: 08-Oct-2013].

[25] Dublin City University. School of Computing. [Online]. Available: http://www.computing.dcu.ie/. [Accessed: 05-Jan-2014].

[26] Durkee D. Why cloud computing will never be free. *Commun. ACM* 2010; 53: 62, DOI:10.1145/1735223.1735242.

[27] Elvesæter B, Ict S. System Development Lecture # 9 : Service Modelling with SoaML Tutorial outline • PART I : Introduction Service Modelling with SoaML Language. 2011; 1–38,.

[28] Erl T. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005, 792.

[29] Erl T. *SOA: principles of service design*. Prentice Hall, 2008, 608.

[30] Extremadura U de. Quercus Software Engineering Group. [Online]. Available: http://www.unex.es/investigacion/grupos/quercus. [Accessed: 05-Jan-2014].

[31] Fraunhofer Cloud. Cloud IT Services. [Online]. Available: http://www.cloud.fraunhofer.de/en.html. [Accessed: 04-Feb-2014].

[32] Frey S, Reich C, Lüthje C. Key Performance Indicators for Cloud Computing SLAs. *Emerg. 2013, Fifth Int. Conf. Emerg. Netw. Intell.* 2013; .

[33] Google. ¿Qué es Google App Engine? - Google App Engine — Google Developers, 2013. [Online]. Available: https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=es-FI&csw=1. [Accessed: 08-Oct-2013].

[34] Google. Google Apps for Business, 2011. [Online]. Available: http://www.google.com/intx/en/enterprise/apps/business/#utm_campaign=fi&utm_source=fi-ha-emea-fi-bk&utm_medium=ha&utm_term=google apps. [Accessed: 08-Oct-2013].

[35] Gotel O, Joseph M, Meyer B, Eds. *Software Engineering Approaches for Offshore and Outsourced Development*, **35**: Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[36] Greengard S. Cloud computing and developing nations. *Commun. ACM* 2010; **53**: 18, DOI:10.1145/1735223.1735232.

[37] Haines MN, Rothenberger MA. How a service-oriented architecture may change the software development process. *Commun. ACM* 2010; **53**: 135, DOI:10.1145/1787234.1787269.

[38] Hamdaqa M, Livogiannis T, Tahvildari L. A Reference Model for Developing Cloud Applications. *Proc. 1st Int. Conf. Cloud Comput. Serv. Sci.* 2011; 98–103, DOI:10.5220/0003393800980103.

[39] Hull R, Su J. Tools for composite web services. *ACM SIGMOD Rec.* 2005; **34**: 86, DOI:10.1145/1083784.1083807.

[40] Hurwitz J, Bloor R, Baroudi C, Kaufman M. *Service oriented architecture for dummies*. 2007.

[41] IBM. Rational Software Architect. IBM Corporation, 25-Jan-2014.

[42] ICCLAB. The InIT Cloud Computing Lab. [Online]. Available: http://www.cloudcomp.ch/research/foundation/projects/the-init-cloud-lab/. [Accessed: 04-Feb-2014].

[43] Inc. NM. Magic draw - Unified Modeling Language (UML), SYSML, UPDM, SOA, Business Process Modeling Tools. [Online]. Available: http://www.nomagic.com/. [Accessed: 25-Jan-2014].

[44] IT University of Copenhagen. Software Development Group. [Online]. Available: http://sdg.wikit.itu.dk/. [Accessed: 05-Jan-2014].

[45] Josuttis NM. *SOA in practice*. O'Reilly Vlg. Gmbh & Co., 2007, 342.

[46] Jouault F, Allilaire F, Bézivin J, Kurtev I. ATL: A model transformation tool. *Sci. Comput. Program.* 2008; **72**: 31–39, DOI:10.1016/j.scico.2007.08.002.

[47] Jouault F, Kurtev I. Transforming models with ATL. *Satell. Events Model. 2005 Conf.* 2006; 128–138,.

[48] Kerherve B, Nguyen KK, Gerbe O, Jaumard B. A Framework for Quality-Driven Delivery in Distributed Multimedia Systems, in *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, 2006, 195–195, DOI:10.1109/AICT-ICIW.2006.14.

[49] Khajeh-Hosseini A, Greenwood D, Sommerville I. Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS, in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, 450–457, DOI:10.1109/CLOUD.2010.37.

[50] Kitchenham B, Charters S. Guidelines for performing systematic literature reviews in software engineering. 2007; .

[51] Kommalapati H. Cloud Computing - Windows Azure for Enterprises, 2010. [Online]. Available: http://msdn.microsoft.com/en-us/magazine/ee309870.aspx. [Accessed: 07-Oct-2013].

[52] Kommalapati H. SaaS and SOA - Hanuk's Microsoft Platform Strategy Blog - Site Home - MSDN Blogs, 2007. [Online]. Available: http://blogs.msdn.com/b/hanuk/archive/2007/04/08/saas-and-soa.aspx. [Accessed: 07-Oct-2013].

[53] Kothari C, Arumugam AK. Cloud Application Migration, *Cloud Computing Journal*, 2010.

[54] Krafzig D, Banke K, Slama D. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall, 2004, 408.

[55] Lero. The Irish Software Engineering Research Centre. [Online]. Available: http://www.lero.ie/. [Accessed: 05-Jan-214AD].

[56] Linthicum DS. *Cloud computing and SOA convergence in your enterprise : a step-by-step guide*. Addison-Wesley, 2010, 264.

[57] Louridas P. Up in the Air: Moving Your Applications to the Cloud. *IEEE Softw.* 2010; 27: 6–11, DOI:10.1109/MS.2010.109.

[58] Manes AT. Application Platform Strategies Blog: SOA is Dead; Long Live Services, 2009. [Online]. Available: http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html. [Accessed: 06-Oct-2013].

[59] Mell P, Grance T. The NIST Definition of Cloud Computing, Recommendations of the National Institute of Standards and Technolog. *Natl. Inst. Stand. Technol.* 2011; .

[60] Mellor SJ, Scott K, Uhl A, Weise D. *MDA Distilled: Principles of Model-Driven Architecture*. Addison-Wesley, 2004, 176.

[61] Mendes E. A systematic review of web engineering research, in *2005 International Symposium on Empirical Software Engineering, 2005.*, 2005, 00: , 481–490, DOI:10.1109/ISESE.2005.1541857.

[62] Microsoft. Cliente de prueba de WCF (WcfTestClient). [Online]. Available: http://msdn.microsoft.com/es-es/library/bb552364(v=vs.110).aspx. [Accessed: 09-Feb-2014].

[63] Microsoft. Tips for Migrating Your Applications to the Cloud, *MSDN Magazine*, 2010.

[64] Microsoft. Visual Studio. [Online]. Available: http://www.visualstudio.com/. [Accessed: 08-Feb-2014].

[65] Microsoft. WebRole Schema. [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/gg557553.aspx. [Accessed: 07-Feb-2014].

[66] Microsoft. Windows Azure Service Configuration Schema (.cscfg File). [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/ee758710.aspx. [Accessed: 07-Feb-2014].

[67] Microsoft. Windows Azure Service Definition Schema (.csdef File). [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/ee758711.aspx. [Accessed: 07-Feb-2014].

[68] Microsoft. WorkerRole Schema. [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/gg557552.aspx. [Accessed: 07-Feb-2014].

[69] Microsoft Research. Cloud Computing Research Group. [Online]. Available: http://research.microsoft.com/en-us/groups/ccrg/. [Accessed: 23-Jul-2013].

[70] Miller J, Mukerji J. MDA Guide Version 1.0. 1. *Object Manag. Gr.* 2003; .

[71] MMT Project. Model to Model Transformation - MMT - Eclipsepedia. [Online]. Available: http://wiki.eclipse.org/MMT. [Accessed: 08-Feb-2014].

[72] ModelDriven. ModelPro | ModelDriven.org. [Online]. Available: http://portal.modeldriven.org/project/ModelPro. [Accessed: 25-Jan-2014].

[73] ModelioSoft. ModelioSoft, download UML modeling tool and free products. [Online]. Available: http://www.modeliosoft.com/. [Accessed: 25-Jan-2014].

[74] Mossburg G. 4 keys to success in the cloud, *Government Computer News Journalernment Computer News Journal*, 2011.

[75] MTL. Acceleo. [Online]. Available: http://www.eclipse.org/acceleo/. [Accessed: 08-Feb-2014].

[76] OASIS. OASIS Security Services (SAML) TC, 2013. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security. [Accessed: 05-Oct-2013].

[77] OASIS. UDDI Version 3.0.2., 2004. [Online]. Available: http://uddi.org/pubs/uddi_v3.htm. [Accessed: 05-Oct-2013].

[78] Obeo. Model Driven Company. [Online]. Available: http://www.obeo.fr/. [Accessed: 08-Feb-2014].

[79] OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. *Final Adopt. Specif. (November 2005)* 2008; .

[80] OMG. MOF Model to Text Transformation Language. *2008-01-16]. http//www. omg. org/spec/MOFM2T/1.0/ …* 2008; .

[81] OMG. Object Constrain Languaje (OCL) Specification. 2006; 03: .

[82] OMG. OMG Meta Object Facility (MOF) Core Specification. 2013; .

[83] OMG. Service oriented architecture Modeling Language (SoaML) Specification. *Object Manag. Gr.* 2012; .

[84] OMG. Software & Systems Process Engineering Meta-Model Specification. 2008; .

[85] Oracle. Oracle SaaS Platform: Building On-Demand Applications, 2008. [Online]. Available: http://www.oracle.com/us/technologies/cloud/026989.pdf. [Accessed: 08-Oct-2013].

[86] Özyeğin University. Research in Computer Science. [Online]. Available: http://ozyegin.edu.tr/RESEARCH/Departmental-Research/CSResearch?lang=en-US. [Accessed: 05-Jan-2014].

[87] Pace E, Betts D, Densmore S, Dunn R. *Moving Applications to the Cloud on the Microsoft AzureTM Platform*. 2010.

[88] Palacios M, García-Fanjul J, Tuya J. Testing in Service Oriented Architectures with dynamic binding: A mapping study. *Inf. Softw. Technol.* 2011; **53**: 171–189, DOI:10.1016/j.infsof.2010.11.014.

[89] Papazoglou M. *Web services: principles and technology*. Prentice Hall, 2007, 784.

[90] Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic Mapping Studies in Software Engineering. 2007; 1–10,.

[91] Petersen, K., Feldt, R., Mujtaba, S., Mattsson M. Systematic mapping studies in software engineering. *12th Int. Conf. Eval. Assess. Softw. Eng.* 2008; 68–77,.

[92] Polithecnic University of Madrid. Distributed Systems Laboratory. [Online]. Available: http://lsd.ls.fi.upm.es/lsd. [Accessed: 05-Jan-2014].

[93] Reese G. *Cloud Application Architectures*. O'Reilly Media, 2009, 208.

[94] Research Institute for Symbolic Computation, Linz, AustriaUniversitatea de Vest Timisoara R, Universitatea Politehnica Timisoara R. Institute e-Austria Timisora. [Online]. Available: http://www.ieat.ro/. [Accessed: 05-Jan-2014].

[95] Rimal BP, Choi E, Lumb I. A Taxonomy and Survey of Cloud Computing Systems. *2009 Fifth Int. Jt. Conf. INC, IMS IDC* 2009; 44–51, DOI:10.1109/NCM.2009.218.

[96] Rittinghouse JW. *Cloud Computing: Implementation, Management, and Security*. CRC Press, 2009, 340.

[97] Riungu LM, Taipale O, Smolander K. Research Issues for Software Testing in the Cloud, in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, 557–564, DOI:10.1109/CloudCom.2010.58.

[98] Salesforce. Cloud apps and platform, 2011. [Online]. Available: http://www.salesforce.com/eu/crm/products.jsp. [Accessed: 08-Oct-2013].

[99] Sattaluri R. Application Migration Considerations for Cloud Computing, *Cloud Computing Journal*, 2011.

[100] Seidewitz E. What models mean. *IEEE Softw.* 2003; **20**: 26–32, DOI:10.1109/MS.2003.1231147.

[101] Sirtl H. Software plus Services: New IT- and Business Opportunities by Uniting SaaS, SOA and Web 2.0, in *2008 12th International IEEE Enterprise Distributed Object Computing Conference*, 2008, xviii–xviii, DOI:10.1109/EDOC.2008.60.

[102] SOA Systems Inc. SOA Specifications - XML - SOAP in a Nutshell, 2009. [Online]. Available: http://www.servicetechspecs.com/soap2. [Accessed: 06-Oct-2013].

[103] Sparx Systems. Enterprise Architect - UML® para Negocio, Software y Sistemas. [Online]. Available: http://www.sparxsystems.es/. [Accessed: 25-Jan-2014].

[104] SPIRENT. Data Center Testing, 2009. [Online]. Available: http://www.spirent.com/~/media/White Papers/Broadband/PAB/Data_Center_Testing_WhitePaper.ashx. [Accessed: 07-Oct-2013].

[105] SPIRENT. The Ins and Outs of Cloud Computing, 2010. [Online]. Available: http://www.spirent.com/White-Papers/Broadband/PAB/Cloud_Computing_WhitePaper.aspx. [Accessed: 07-Oct-2013].

[106] Stahl T, Völter M. *Model-Driven Software Development - Technology, Engineering, Management*. John Wiley and Sons, Ltd., Chichester, England, 2006, 446.

[107] The Eclipse Foundation. Eclipse Open Source Community. [Online]. Available: http://www.eclipse.org/. [Accessed: 06-Jan-2014].

[108] The Vienna University of Technology. Distributed Systems Group. [Online]. Available: http://www.infosys.tuwien.ac.at/research.html. [Accessed: 05-Jan-2014].

[109] Tran V, Keung J, Liu A, Fekete A. Application migration to cloud, in *Proceeding of the 2nd international workshop on Software engineering for cloud computing - SECLOUD '11*, 2011, 22, DOI:10.1145/1985500.1985505.

[110] Umeå University. Department of Computer Science. [Online]. Available: http://www.cs.umu.se/english/?languageId=1. [Accessed: 05-Jan-2014].

[111] Universita degli Studi di Napoli. PARSeC Research Group. [Online]. Available: http://parsec.unina2.it/~parsecj/. [Accessed: 05-Jan-2014].

[112] Universitat Rovira i Virgili. AST Research Group. [Online]. Available: http://ast-deim.urv.cat/web/. [Accessed: 05-Jan-2014].

[113] Universität Stuttgart. Institute of Architecture of Application Systems. [Online]. Available: http://www.iaas.uni-stuttgart.de/institut/indexE.php. [Accessed: 05-Jan-2014].

[114] Vaquero LM, Rodero-Merino L, Caceres J, Lindner M. A break in the clouds. *ACM SIGCOMM Comput. Commun. Rev.* 2008; **39**: 50, DOI:10.1145/1496091.1496100.

[115] Varia J. Migrating your existing applications to the aws cloud - A Phase-driven Approach to Cloud Migration, *Amazon Web Services*, 1–23, 2010.

[116] W3C. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), 2011. [Online]. Available: http://www.w3.org/TR/soap12-part1/. [Accessed: 06-Oct-2013].

[117] Watson R. How to migrate applications to the cloud | Information Age, *Information Age*, 2012. [Online]. Available: http://www.information-age.com/technology/applications-and-development/1681988/how-to-migrate-applications-to-the-cloud. [Accessed: 01-Oct-2013].

[118] Wikipedia. Computación grid. [Online]. Available: http://es.wikipedia.org/wiki/Computación_grid. [Accessed: 09-Feb-2014].

[119] Wikipedia. Microsoft Visual Studio. [Online]. Available: http://es.wikipedia.org/wiki/Microsoft_Visual_Studio. [Accessed: 08-Feb-2014].

[120] Yu Q, Liu X, Bouguettaya A, Medjahed B. Deploying and managing Web services: issues, solutions, and directions. *VLDB J.* 2006; **17**: 537–572, DOI:10.1007/s00778-006-0020-3.

[121] Yunus M. Federated SOA: A Pre-Requisite for Enterprise Cloud Computing | Cloud Computing Journal, 2010. [Online]. Available: http://cloudcomputing.sys-con.com/node/1233457. [Accessed: 06-Oct-2013].

[122] Yunus. M. Understanding enterprise to cloud migration costs and risks, *ebizQ Journal*, 2010.