DEPARTAMENTO DE
INFORMÁTICA Y AUTOMÁTICA

FACULTAD DE CIENCIAS

VNiVERSiDAD
D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

TESIS DOCTORAL

Sistemas organizativos para la asignación dinámica de recursos computacionales en entornos distribuidos

AUTOR

D. Fernando De la Prieta Pintado

DIRECTORES

Dr. D. Javier Bajo Pérez
Dra. Dña. Sara Rodríguez González

Julio de 2014

La memoria titulada "Sistema organizativos para la asignación dinámica de recursos computacionales en entornos distribuidos" que presenta D. Fernando De la Prieta Pintado para optar al Grado de Doctor por la Universidad de Salamanca ha sido realizada bajo la dirección de la Dra. Dña. Sara Rodríguez González, Profesora Ayudante Doctor en el Departamento de Informática y Automática; así como por el Dr. D. Javier Bajo Pérez, Profesor Titular del Departamento de Inteligencia Artificial de la Universidad Politécnica de Madrid.

Salamanca, Julio 2014

El Graduado

Fdo: D. Fernando De la Prieta Pintado

Los Directores

Fdo: Dra. Dña. Sara Rodríguez González
Ayudante Doctor
Departamento de Informática y Automática
Universidad de Salamanca

Fdo: Dr. D. Javier Bajo Pérez
Profesor Titular de Universidad
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid

F. De la Prieta: Sistemas organizativos para la asignación dinámica de recursos computacionales en entornos distribuidos,
PhD on Computer Engineering, © July 2014

Supervisors:
Dr. D. Javier Bajo Pérez
Dra. Dña. Sara Rodríguez González

Location:
Salamanca

No menos que saber, dudar me gusta más.

La divina comedia. Infierno, Canto XI

Dante Alighieri (1265 - 1321)

RESUMEN

Cloud Computing, el conocido paradigma computacional, está emergiendo en los últimos años con gran fuerza. Este paradigma incluye un novedoso modelo de comercialización basado en el pago por uso que ha cambiado radicalmente el modelo de negocio en Internet, lo que ha permitido que las empresas y usuarios individuales puedan alquilar los recursos computacionales que necesitan en cada momento. Este nuevo modelo computacional también ha derivado en que el modelo de producción de estos recursos computacionales evolucione hasta una aproximación cercana al modelo de producción *just-in-time*, en el que sólo se consumen los recursos necesarios para la producción de los servicios en función de la demanda existente en cada momento, hablándose dentro de este ámbito de *elasticidad* en los servicios ofertados. Para que esto sea posible, no cabe duda, que una gran cantidad de tecnologías subyacentes han tenido que madurar para dar como resultado un nicho tecnológico con la capacidad para variar los recursos asociados a cada servicio en función de la demanda.

Sin embargo, pese a los indudables avances que se han producido a nivel tecnológico, todavía hoy existe una gran capacidad de mejora de estos sistemas. En este sentido, en el marco de esta tesis doctoral se propone el uso de los s*istemas multiagente* y, especialmente, aquellos basados en *modelos organizativos* para el control y monitorización de un sistema Cloud Computing. Gracias a esta aproximación, una de las primeras en este campo de investigación, será posible incluir en las plataformas *Cloud* de nueva generación características derivadas de la Inteligencia Artificial, como son la autonomía, la proactividad y, también, la capacidad de aprendizaje. Para ello se propone un modelo único en su concepción, que permite dotar a la organización de agentes inteligentes con capacidades auto-adaptativas en tiempo de ejecución para entornos abiertos, altamente dinámicos en los que, además, existe un cierto grado de incertidumbre. Así gracias a este modelo, el sistema es capaz de variar los recursos computacionales asociados a cada servicio producido en función de la demanda existe por parte de los usuarios, mediante la auto-adaptación dinámica del propio sistema en su conjunto.

ABSTRACT

Cloud Computing, a known computing paradigm, has been emerging in recent years with great strength. This paradigm includes a new marketing model based on the pay per use revenue model, which has radically changed the business model on the internet and allowed companies and individual users to rent the computational resources they need at any given time. This new computational model has also allowed the production model for these computational resources to evolve very closely along the lines of the *just-in-time* production model, in which only the resources needed for the production of services are consumed according to the demand that exists at a given time. Within this context, the term *elasticity* is used in reference to the services being offered. For this to be possible, a vast number of underlying technologies have had to grow and develop in order to produce a technology niche with the ability to vary the resources associated with each service according to demand.

However, despite the undeniable advances that have been produced at a technological level, there is still much room for improvement in these systems. In this regard, the framework of this doctoral thesis proposes the use of multiagent systems, particularly those based on organizational models to control and monitor a Cloud Computing system. Due to this approach, one of the first in this field of research, new generation Cloud platforms will be able to include characteristics derived from Artificial Intelligence, such as autonomy, proactivity, and learning capabilities. To this end, a new model, unique in its conception, has been proposed; it can provide the organization with intelligent agents with self-adaptive abilities in execution time for open and highly dynamic environments in which there is also a high degree of uncertainty. With this model, the system is able to vary the computational resources associated with each service produced according to existing user demand, using the system's own dynamic self-adaptation.

AGRADECIMIENTOS

Una vez finalizado el trabajo es el momento de echar la vista atrás para agradecer y poder, de este modo, mirar hacia adelante. Ahora que estoy en un estado próximo a finalizar el último gran escalón de mi formación académica, me gustaría dar las gracias a todos aquellos que han hecho posible que hoy por hoy este en condición de poder obtener el Grado de Doctor por la Universidad de Salamanca.

En primer lugar me gustaría dar las gracias a mis directores, hoy amigos, por los buenos momentos que hemos compartido. Gracias a Javier Bajo por enseñarme cómo pensar en este complejo mundo de la universidad, así como por darme la primera oportunidad de impartir docencia en el nivel universitario. Gracias a Sara Rodríguez, por enseñarme cómo trabajar y cómo enfrentarme a los retos del día a día.

Gracias al resto de compañeros del grupo de investigación BISITE, a todos los que son ahora, pero también a los que se fueron. Gracias por todos los buenos momentos que hemos pasado y que hacen posible que seamos una gran familia. En especial, me gustaría dar las gracias a Juan M. Corchado, gracias por darme la oportunidad de vivir esta aventura que representa trabajar en la universidad y por saber sacar lo mejor de mi.

También me gustaría dar las gracias a mis compañeros del Departamento de Informática y Automática, los que hace no mucho eran mis profesores y hoy se han convertido en compañeros.

Más allá del ámbito académico, me gustaría dar las gracias a mis amigos, tanto los zamoranos, como los salmantinos, porque los buenos momentos vividos me han permitido sobrellevar el trabajo que supone realizar una tesis doctoral.

Un agradecimiento muy especial a mis padres, Alfonso y Rosa, quien con no pocos sufrientos han sabido dar forma a mi carácter y personalidad haciendo posible que me convirta en la persona que soy. Me gustaría acordarme también de mis tíos, Papías y Dolores, ellos también forman una parte muy importante de mi vida.

Finalmente, gracias a Gloria, por su apoyo incondicional, por su comprensión, por entender tantas cosas... Sin ti no hubiera sido posible llegar hasta aquí.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

Durante los últimos años la industria tecnológica y la comunidad científica están realizando un gran esfuerzo de investigación e implantación del paradigma tecnológico *Cloud Computing* (CC). Así, el número de plataformas tanto privadas, como públicas está creciendo rápidamente [Fisher *et al.*, 2010] [Lou *et al.*, 2011] [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012]. No cabe duda que la gran aceptación social de este paradigma [Leavitt, 2009] ha motivado en gran medida su desarrollo debido al interés económico de las grandes empresas tecnológicas que subyacen frente a los aspectos púramente técnicos [Buyya *et al.*, 2009] [Armbrust *et al.*, 2010].

Las plataformas CC desde un punto de vista externo, ofrecen tres tipos de servicios ampliamente conocidos (*software*, plataforma e infraestructura) [Mell *et* Grance, 2011], donde la principal novedad con respecto a las tecnologías precedentes radica en el hecho de ofrecer cualquier tipo de capacidad computacional como servicio a través de Internet. El modelo de comercialización también es innovador ya que está basado en el pago por uso (*pay-as-you-go*) [Armbrust *et al.*, 2010], al igual que cualquier otro servicio público tradicional (luz, agua, gas, etc.). En este sentido, CC sigue el modelo propuesto por el paradigm abstracto *Utility computing* [Ross *et* Westerman, 2004] en el que los usuarios para acceder a los servicios deben negociar y establecer previamente un acuerdo de uso (*Service Level Agreement*, SLA) [Alhamad *et al.*, 2010a]. Una vez establecido este contrato de uso de bienes computacionales, tanto los usuarios (habitualmente pagando), como el sistema CC (manteniendo el servicio) están obligados a cumplir este contrato.

Este novesoso modelo de comercialización obliga a las plataformas CC a mantener la calidad de los servicios según se ha acordado previamente, lo que hace necesario analizar cómo la arquitectura interna hace posible producir este tipo de servicios. Así pues, las novedades vienen determinadas por el innovador espectro de tecnologías subyacentes (virtualización, granjas de servidores, servicios web, etc.), cuya madurez recientemente alcanzada hace posible que los servicios sean ofertados con el mismo nivel de calidad con independencia de la demanda existente por parte de los usuarios [Liu *et al.*, 2011] [Wang *et al.*, 2008] [Zhang *et al.*, 2010].

Las nuevas posibilidadades a nivel tecnológico conllevan al nacimiento de un nuevo concepto, el de la elasticidad [Chiu, 2010]. Este concepto está basado en el método de producción *just-in-time* [Hutchins, 1999] que hace referencia a la forma de producción de los servicios (computacionales) y los recursos

necesarios para ello. Así pues, los servicios producidos reciben sólo la cantidad de recursos necesarios para mantener un nivel calidad constante atendiendo a la demanda instantánea [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012]. En la práctica, se traduce en que el usuario final hace uso de un conjunto de servicios a través de internet que siempre están disponibles y que a priori son ilimitados.

Para los usuarios del paradigma, este modelo de comercialización es muy ventajoso, ya que les permite no tener la obligación de provisionarse con recursos computacionales adicionales para contener los picos en la demanda de sus productos o servicios, ni tampoco disponer de la infraestructura necesaria para proporcionarlos. Lo que permite transformar los gastos de capital en gastos operacionales [Armbrust *et al.*, 2010]. En contraposición, los proveedores deben proporcionar servicios CC elásticos que deben ser gestionados a través de entornos de tipo clústers HPC (*High Performance Computing*) cuyo control y monitorización debe realizarse de forma automática, proporcionando un acceso transparente a los usuarios finales

Pese a los indudables avances que se han conseguido en los últimos años en el marco de los sistemas CC, no hay que olvidar que es una tecnología relativamente reciente que comienza en el año 2007 [Lohr, 2007]. Aún hoy en día, en algunos aspectos es un paradigma inmaduro [Subashini *et al.*, 2011] [Bhadauria *et* Sanyal, 2012] [Tripathi *et* Jigeesh, 2013] y por lo tanto todavía existen retos y necesidades que son del interés de una gran parte de la comunidad científica [Ullah *et* Xuefeng, 2013] [Moreno-Vozmediano *et al.*, 2013]. En este sentido, uno de los grandes retos es la gestión automática de los recursos computacionales de un entorno CC [Buyya *et al.*, 2010b] [Buyya *et al.*, 2011] y su correcta asignación a los servicios proporcionados por este tipo de plataformas, ya que debe asignarse en función de la demanda y los acuerdos SLA alcanzados.

Este trabajo de tesis doctoral se plantea como una de las primeras aproximaciones de los Sistemas Multiagente (SMA) basados en Organizaciones Virtuales (OV) en el marco del paradigma CC. El propósito del sistema organizativo propuesto no es sólo el control y monitorización de la infraestructura *hardware* (real o virtual) de un entorno CC, sino también la asignación de recursos computacionales disponibles entre los diferentes servicios ofertados a los usuarios finales, en función de la demanda existente. Este modelo de distribución o reparto de recursos que se propone se sustentará sobre las capacidades auto-adaptativas en entornos abiertos, dinámicos y heterogéneos que proporcionan los SMA con capacidades organizacionales [Davidsson, 2001] [Zambonelli *et al.*, 2003] [Rodríguez, 2010].

Este primer capítulo de la memoria del trabajo de investigación que se ha realizado, comienza la identificación de los problemas abiertos en el marco del paradigma CC y la formulación de la hipótesis inicial de esta tesis doctoral

(apartado 1.1). A continuación, los apartados 1.2 y 1.3 presentan, respectivamente, los objetivos propuestos en el inicio del trabajo y la metodología de investigación que se ha empleado. Finalmente, en el último apartado del capítulo (1.4) se detalla la estructura de la presente memoria.

1.1 PROBLEMAS ABIERTOS, MOTIVACIÓN E HIPÓTESIS

Actualmente, las plataformas CC son principalmente utilizadas por su grandes capacidades computacionales, tanto en términos de potencia de computación, como por su facilidad para el almacenamiento de grandes volúmenes de datos. En este sentido, la mayoría de las plataformas CC conocidas [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012] [Mahjoub *et al.*, 2011] hacen mayor énfasis en proporcionar servicios de *infraestructura* (a través de la virtualización del *hardware*), sin tener en cuenta los servicios de nivel superior (*plataforma* y *software*).

Cualquier sistema CC se caracteriza por las propiedades de su infraestructura subyacente y los servicios (o capacidades computacionales) que ofrecen al usuario a través de Internet. Mientras que la infraestructura aporta recursos computacionales al sistema, los servicios consumen estos recursos. El principal reto en este contexto, consiste en que se debe mantener el nivel de calidad de los servicios (QoS, *Quality of Service*) con independencia de su demanda, lo que implica que sea necesario gestionar dinámicamente los recursos computacionales asignados a cada servicio ofertado al usuario. Para ello estas plataformas, en la mayoría de los casos, utilizan un modelo de gestión (monitorización y control) de la infraestructura propia que es centralizado, lo que constituye en sí mismo una limitación, ya que choca frontalmente con el principio de alta disponibilidad que se le presumen a los sistemas CC. Además, dada la complejidad tecnológica que implica proporcionar servicios de infraestructura elásticos, éstas plataformas tampoco hacen un énfasis especial en la optimización de los recursos *hardware*, los cuáles consumen ingentes cantidades de energía y tienen enormes necesidades de refrigeración [Song *et al.*, 2009].

Formalmente, el problema se plantea desde dos puntos de vista. Por un lado, una vista *micro*, en la que la reasignación de recursos se realiza en cada uno de los servicio ofertados, o dentro de cada servidor físico. Por otro lado, a nivel *macro*, la reasignación de recursos se produce a nivel global del entorno CC, lo que puede implicar varios servicios o nodos computacionales. En el estado del arte se observan diversos trabajos en este ámbito de la asignación de recursos entornos CC, que pueden ser clasificados en tres grandes grupos: (i) los que toman las decisiones de forma centralizada (que son la mayoría) [Buyya *et al.*, 2011] [Beloglazov *et al.*, 2012]; (ii) los que estiman o planifican el uso sin tener en cuenta la demanda instantánea [Gong *et al.*, 2010] [Han *et al.*, 2012]; y, finalmente, (iii) los que optimizan los recursos en función de la demanda instantánea sin tener en cuenta la demanda histórica [Wei *et al.*, 2010a].

Si se realiza una investigación más profunda y detallada; en primer lugar, desde el punto de vista micro, aunque no es posible encontrar una gran cantidad de estudios en el estado de arte, si que existen modelos que satisfacen el problema; ya sea en cuanto la reasignación de recursos entre las máquinas virtuales que aloja un servidor [Miao *et al.*, 2011], como en el balanceo entre las diferentes máquinas virtuales asignadas a un recurso [Cardellini *et al.*, 1999]. En segundo lugar, desde el punto de vista *macro* las referencias existentes siguen modelos principalmente orientados al mercado [Buyya *et al.*, 2008] [You *et al.*, 2009], modelos matemáticos [Wei *et al.*, 2010a] [Van *et al.*, 2009b] y, recientemente, se empiezan a observar trabajos que tienen en cuenta el consumo energético en la distribución de los recursos computacionales [Beloglazov *et al.*, 2012].

Estos modelos no sólo deberían proporcionar un modelo de elasticidad (creciente y decreciente), sino que también deben proporcionar herramientas para reducir los costes operacionales del entorno CC; tanto desde el punto de vista del usuario final, como también desde el punto de vista de proveedor, ya que como se ha indicado previamente, los entornos HPC requieren una gran cantidades de recursos energéticos y de refrigeración [Song *et al.*, 2009]. Sin embargo, los trabajos existentes en el estado del arte están basados en métodos que utilizan algoritmos centralizados basados en modelos matemáticos y heurísticos, que no aseguran la eficiencia del sistema y menos su disponibilidad frente a fallos. Este tipo algoritmos deberían evolucionar hacia un modelo en el que los diferentes componentes dispongan de una representación parcial del entorno, que les permita interaccionar y compartir información con sus iguales, de forma que los algoritmos de gestión de los recursos se distribuyan a lo largo de la infraestructura, lo que por ende facilitará su implantación con independencia del tamaño del centro de datos.

En este contexto de inmadurez tecnológica, la teoría de agentes y SMA [Russel *et* Norving, 1995] [Wooldridge *et* Jennings, 1995] pueden aportar un nuevo modelo de gestión de sistemas CC basado en la distribución de responsabilidades, la flexibilidad y la autonomía. La gestión de las funcionalidades del núcleo de un sistema CC mediante un modelo basado en SMA permitiría a las plataformas resultantes ser mucho más mucho más eficientes, escalables y adaptables que las existentes actualmente. Sin embargo, la unión de ambos modelos computacionales (SMA y CC) es un gran reto, pero un reto asumible en el marco de este trabajo de investigación, debido a que un entorno CC es considerado un sistema abierto debido a su heterogeneidad, dinamicidad e incertidumbre:

- Es **heterogéneo**, ya que puede incluir diversos elementos y componentes muy diversos entre sí, tanto a nivel *hardware*, como a nivel *software*.

- Es **dinámico** ya que la demanda de sus servicios varía a lo largo del tiempo, y además, el funcionamiento de sus componentes tiene que adaptarse a los nuevos estados que puedan darse, incluyendo la admisión y salida de elementos del sistema en pleno funcionamiento.
- Tiene **incertidumbre** ya que cada uno de sus componentes conoce tan sólo parte del sistema; teniendo que tomar decisiones en base a la información parcial disponible en cada momento.

La aplicación de SMA en sistemas abiertos es un reto conocido. En el estado del arte existe una gran variedad de trabajos que ha sido desarrollados y evaluados en diversos contextos abiertos, obtenido un notable éxito en su aplicación [Zambonelli *et al.*, 2004] [Reitbauer *et al.*, 2004] [Weyns *et al.*, 2004] [Capera *et al.*, 2003] [Razavi *et al.*, 2005]. En el marco de este trabajo, se consideran especialmente efectivos los SMA diseñados mediante modelos organizativos ya que pueden aportar soluciones avanzadas e innovadoras [Dignum, 2004] que permitan explotar estrategias diferenciadoras que aporten flexibilidad, capacidad y rapidez de respuesta, dentro de una estrategia dirigida a satisfacer al consumidor [Schertler, 1998]. El uso de SMA permite continuar con la investigación en técnicas, herramientas y metodologías que permitan incorporar a las plataformas CC características inteligentes como pueden ser la autonomía o el aprendizaje, entre otras.

Así pues, la hipótesis de partida es que *es posible modelar el sistema de control y monitorización de una plataforma CC mediante el uso de OV de agentes inteligentes que se auto-adapten y reorganicen en función de las necesidades del contexto. Así pues, los agentes individuales de forma autónoma gestionarán los recursos computacionales y los servicios disponibles, coordinándose con el resto de agentes para proporcionar un modelo de adaptación dinámico y distribuido en función de los intereses del usuario final.*

Aprovechando las ventajas que nos proporcionan las particularidades del desarrollo de SMA desde el punto de vista organizacional y considerando las carencias existentes actualmente en el marco del paradigma CC, se formalizará un modelo arquitectónico dinámico y adaptativo que hará uso de técnicas de optimización basada en la investigación operativa y el aprendizaje de la plataforma a través de los sistemas de razonamiento basados en casos [De Mántaras *et al.*, 2005] [Aamodt *et* Plaza, 1994].

1.2 OBJETIVOS

Considerando las características existentes en el paradigma CC en tanto en cuanto a la redistribución de recursos y dependencia del entorno tecnológico subyacente. Como hipótesis de partida se plantea la necesidad de incorporar modelos asociados a la Inteligencia Artificial (IA) para la gestión de entornos computacionales de última generación. Las particularidades del desarrollo de SMA desde el punto de vista organizacional posibilitan la formalización de una arquitectura que satisfaga las demandas del entorno CC.

Por lo tanto el objetivo principal que se pretende conseguir es *modelar una arquitectura multiagente basada en OV para la monitorización y control de un entorno CC, así como la investigación y desarrollo de un modelo de asignación dinámica y adaptativa de recursos computacionales en entornos distribuidos.*

Para realizar este trabajo también es necesario satisfacer un conjunto de objetivos más específicos que haga posible este objetivo principal:

- Estudiar el paradigma CC a nivel global, incluyendo tipos de servicios, modelos de despliegue, arquitecturas existentes y posibles debilidades en el paradigma.
- Analizar la relación existente entre los SMA y los sistemas CC, identificando las principales áreas aprovechamiento de los SMA en el marco del paradigma CC.
- Estudiar las tendencias en el marco del análisis, diseño y posterior desarrollo de los SMA organizativos, así como de los modelos de adaptación y coordinación.
- Realizar un estudio de las tecnologías complementarias, ya sean aquellas que tienen un perfil claramente tecnológico (virtualización, balanceo de carga, computación de alto rendimiento, etc.). Como aquellas aproximaciones más cercanas a la investigación con son los sistemas de razonamiento basados en casos.
- Diseñar un modelo de integración basado en la teoría de agentes, prestando especial atención en los sistemas organizativos.
- Aplicar mecanismos generales de optimización de recursos en el marco de la distribución de recursos computacionales en entornos CC.
- Diseñar un modelo adaptativo y de aprendizaje sobre la base que proporciona el modelo de razonamiento basados en casos, es decir, que permita apoyarse en experiencias pasadas, para generar la base del conocimiento actual.
- Evaluar empíricamente el modelo de arquitectura y los algoritmos de distribución de recursos propuestos en un entorno de aplicación real.

1.3 METODOLOGÍA DE INVESTIGACIÓN

En una tesis donde el principal componente de trabajo está relacionado con la investigación, parece adecuado plantear una metodología que permita establecer objetivos alcanzables en un tiempo fijado. Posteriormente, evaluar estos los resultados de forma conjunta con los tutores y, finalmente, discutir el siguiente nivel de objetivos atendiendo a los resultados obtenidos.

El método de investigación que se ha seguido es conocido como *action-research* [Reason *et* Bradbury, 2001]. Se caracteriza por estar orientado a la acción y el cambio, lo que permite focalizarse en un problema para generar conocimiento de investigación relevante en un periodo de tiempo establecido. Este método es uno de los más habituales en el ámbito la investigación de los sistemas de información, ya que hace posible la investigación empírica.

A continuación se presentan los hitos que se plantearon al inicio del presente trabajo de tesis:

- Revisión detallada del estado del arte e identificación de novedades.
- Diseño de un entorno de pruebas.
- Inclusión de los principios de los SMA en los entorno CC.
- Diseño de un sistema multiagente basado en OV.
- Propuesta, investigación y mejora de algoritmos de distribución de recursos computacionales
- Evaluación de los algoritmos propuestos.

Estos hitos, se han alcanzado a través de un modelo de indagación original planificada, que se ha basado en tres grandes etapas. En primer lugar, la primera etapa se centró en el análisis del estado del arte incluyendo tanto los trabajos relacionados con el paradigma CC, como con los SMA y las OV. Posteriormente, durante la segunda etapa se ha diseñado y desarrollado de forma iterativa e incremental tanto el modelo arquitectónico basado en SMA, como los algoritmos distribución de recursos computacionales. En este sentido, dado que se esperaba obtener una arquitectura *software*, se ha seguido un modelo de desarrollo basado en prototipos de forma que los objetivos perseguidos se han ido incorporando gradualmente al resultado final. Gracias a este modelo ha sido posible alcanzar los objetivos perseguidos en el tiempo estimado. Finalmente, la última etapa ha consistido en la evaluación del trabajo desarrollado, lo que ha dado lugar a la validación tanto de la plataforma multiagente, como la de los algoritmos propuestos.

De forma paralela a estas tres grandes etapas, se ha realizado un trabajo constante de diseminación del conocimiento, resultados y experiencias obtenidas, siendo posible su explotación por parte de la comunidad científica. Esta actividad continua ha dado lugar a la elaboración y presentación de

diversas publicaciones en revistas, congresos, talleres, generación de informes técnicos, etc. que han permitido validar y dar a conocer los avances y resultados parciales de los distintos hitos de la investigación en diferentes ámbitos con la consiguiente realimentación al trabajo.

No cabe duda que para la investigación de un entorno CC es necesario disponer de un entorno de pruebas adecuado. Así, en este sentido, el grupo de investigación BISITE de la Universidad de Salamanca ha desplegado una completa granja 15 de servidores de última generación, con capacidad de virtualización por *hardware* y 16Gb de RAM por equipo. Todos ellos están conectados punto a punto mediante una red 10Gbps. Gracias a esta infraestructura ha sido posible validar todas las innovaciones que se han obtenido en el marco de este proyecto de tesis.

1.4 ESTRUCTURA DE LA MEMORIA

El presente documento se organiza en 6 capítulos y 4 anexos que siguen la estructura secuencial, pero iterativa, del trabajo realizado en el marco de la presente tesis doctoral. A continuación se detalla cada una de estas piezas, las cuáles describen detalladamente el proceso de investigación que se ha seguido.

Los capítulos 2 y 3 abordan el estudio del estado del arte de las tecnologías, métodos, técnicas y herramientas directamente relacionadas con la temática de este trabajo de investigación. En primer lugar, en el **capítulo 2** se presenta un análisis detallado del paradigma computación CC. En este sentido, dado su reciente nacimiento, en primer lugar, se analizan las definiciones existentes con el objetivo de acotar el término, el modelo tecnológico y su ámbito de aplicación (modelos, servicios, roles, etc.). En este segundo capítulo, también se realiza un análisis de las plataformas existentes, así como los riesgos y vulnerabilidades asociadas. El capítulo concluye con el análisis de las debilidades que hoy en día presenta la tecnología CC. A partir de este análisis, el **capítulo 3** se centra en el estudio del estado del arte de los SMA y la incipiente relación que se empieza a observar entre estos sistemas y el paradigma CC. Así este capítulo aporta una extensa relación de los trabajos existentes en el estado de arte sobre la aplicación de SMA en diferentes áreas del paradigma CC y cómo estas aportaciones contribuyen a la mejora del modelo tecnológico en su conjunto. Finalmente, el tercer capítulo aborda el análisis de los modelos organizativos de SMA con el objetivo de facilitar su aplicación en el ámbito del paradigma CC.

El **capítulo 4** está centrado en la descripción del modelo arquitectónico propuesto para la monitorización y control de un entorno CC mediante SMA basados en OV, así como los algoritmos y técnicas desarrolladas para la distribución de recursos en entornos CC. El capítulo comienza con una descripción detallada del entorno computacional, con el objetivo de caracterizar y formalizar la infraestructura computacional subyacente, el modelo, la relación formal existente entre una plataforma CC y los usuarios de la misma. Posteriormente, a partir de esta representación se describe en detalle la arquitectura multiagente propuesta, denominada **+Cloud**. Esta arquitectura se ha modelado mediante el uso de la metodología GORMAS [Argente, 2008] [Argente *et al.*, 2011] la cuál facilita la especificación de SMA abiertos siguiendo la perspectiva de las organizaciones humanas. Finalmente, este capítulo también describe los algoritmos propuestos para la distribución dinámica de recursos en función de la demanda en los servicios. Así pues, dada la naturaleza distribuida de estos algoritmos, el modelo de adaptación dinámica se describe atendiendo a los componentes que realizan la distribución o reparto de recursos computacionales existentes en el sistema CC

entre los diferentes servicios de la plataforma, así como las interacciones que se producen entre ellos.

A continuación el **capítulo 5** recoge la evaluación de la arquitectura y los algoritmos de evaluación propuestos. En este sentido, el entorno de pruebas ha sido una plataforma CC desarrollada en el marco de esta tesis doctoral y que integra la arquitectura de agentes (+Cloud) propuesta. Gracias a esta plataforma ha sido posible evaluar el sistema y algoritmos propuestos en un entorno de explotación realista, lo que por tanto asegura su validez. El capítulo en primer lugar presenta una breve descripción de la plataforma CC y como se ha realizado la integración entre el SMA. Posteriormente, se presenta un caso de estudio que muestra como el sistema se adapta dinámicamente a los cambios en la demanda, así como el proceso de distribución de recursos computacionales. Finalmente, el capítulo concluye con un análisis crítico del modelo propuesto y su comparación con respecto a otros trabajos existentes en el estado del arte y plataformas. Por su parte, el **capítulo 7** presenta las conclusiones obtenidas una vez finalizado el trabajo de investigación, aportando las contribuciones al estado del arte de esta investigación y proponiendo una serie de líneas de trabajo a realizar a medio plazo.

Finalmente, el **capítulo 7** de la memoria recoge un resumen en inglés del trabajo realizado, incluyendo tanto los estudios previos, arquitectura basada en SMA, algoritmos de distribución y evaluación de los mismos. La memoria termina con un listado de las **referencias bibliográficas** que se han utilizado para llevar a cabo este trabajo.

Junto con la memoria, se incluyen cuatro documentos anexos que permiten completar y detallar aspectos que, aunque importantes, no se han recogido en la memoria para facilitar la claridad y comprensión del trabajo de investigación en su conjunto. Estos cuatro anexos recogen ámbitos de estudio y desarrollos realizados a lo largo del proceso de investigación:

- El **Anexo A** describe pormenorizadamente el proceso de análisis y diseño del SMA propuesto para la monitorización, control y adaptación dinámica de una plataforma CC. Para esta tarea se ha utilizado la metodología GORMAS que propone una secuencia de 3 etapas para el diseño de SMA desde un punto de vista organizativo (análisis, diseño y diseño de dinámica). Estas etapas, que su vez se dividen en 8 fases, permiten modelar de forma iterativamente la arquitectura multiagente basada en OV.
- El **Anexo B** se centra en la descripción de la plataforma CC desarrollada por el grupo BISITE y que integra el SMA propuesto, +Cloud. Esta plataforma ha constituido el entorno de pruebas de la arquitectura propuesta y de los algoritmos que hacen posible la distribución dinámica

de recursos entre los diferentes servicios proporcionadas por la plataforma CC en función de su demanda.

- El **Anexo C** recoge un estudio pormenorizado a nivel tecnológico del paradigma CC. El anexo consta de dos partes principales, en el primer apartado se realiza un análisis de los proveedores de servicios CC. Posteriormente, el segundo apartado se centra en analizar el marco tecnológico que hace posible este paradigma, haciendo especial énfasis en la virtualización de infraestructura *hardware*.
- El **Anexo D**, en el que se enumeran los proyectos, publicaciones, trabajos y propiedades intelecturales relacionadas con el desarrollo en este trabajo de investigación.

CAPÍTULO 2
SISTEMAS CLOUD COMPUTING

CC [Mell *et* Grance, 2011], entendido como paradigma computacional, está emergiendo en los últimos años con gran fuerza, hasta el punto de que está empezando a ser muy habitual en cualquier ámbito relacionado con la computación, más allá del contexto social de Internet. Muchas de las iniciativas de la Unión Europea, en el marco del programa Horizonte 2020 (H2020)[1] se centran en el paradidgma tecnológico que proporciona los sistemas CC. Además, empresas punteras en el ámbito tecnológico tales como IBM[2], Amazon[3], Microsoft[4], etc. apuestan decididamente por este modelo computacional en su modelo de negocio y oferta de servicios. Aunque a primera vista pueda ser considerado como un paradigma meramente tecnológico, la realidad demuestra que su rápida progresión está principalmente motivada por los intereses económicos que subyacen alrededor de las características puramente computacionales [Buyya *et al.*, 2009] [Armbrust *et al.*, 2010]. Hoy en día, parece evidente que, aunque todavía existen muchas barreras a superar a nivel técnico [Grobauer *et al.*, 2011] [Bhadauria *et* Sanyal, 2012], el ritmo de innovación viene determinado por los intereses macroecónomicos impuestos por las grandes compañías tecnológicas [Buyya *et al.*, 2009].

Como ya se ha anticipado en el Capítulo 1, es posible identificar en las actuales plataformas CC ciertas debilidades a la hora de gestionar de forma eficiente sus recursos, responder de forma autónoma a los picos en la demanda, automatizar el proceso de toma de decisiones, responder a las caídas en los sistemas, etc. [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012]. Para abordar estos problemas existentes hoy en día, se identifican a los SMA [Russel *et* Norving, 1995] [Wooldridge *et* Jennings, 1995], según la hipótesis de este trabajo como la clave para proporcionar una solución eficiente a estos problemas, permitiendo la evolución del paradigma CC mediante la incorporando nuevas características y capacidades a las plataformas CC existentes [Talia, 2012] [Kang *et* Sim, 2011].

Dada la complejidad que presenta los sistemas CC donde existen una gran variedad de tecnologías, componentes, sistemas, etc. que tienen que trabajar

[1] http://ec.europa.eu/programmes/horizon2020/
[2] http://www.ibm.com/cloud-computing/us/en/
[3] https://aws.amazon.com/en/what-is-cloud-computing/
[4] https://www.microsoft.com/en-us/news/presskits/cloud/

de forma conjunta, se revisará en detalle las características de las platformas CC, identificando fortalezas y debilidades, así como identificando necesidades concretas. De este modo este análisis inicial servirá como punto de partida para la incorporación de los SMA en este contexto tecnológico.

El siguiente apartado del capítulo (2.1) contiene una revisión histórica del paradigma CC, revisión que se enlaza con las tecnologías relacionadas, las cuáles se presentan en el apartado 2.2. A continuación en el apartado 2.3 se analizan de forma crítica las diferentes definiciones existentes en el estado del arte, los tipos de servicios ofertados, roles del paradigma, los modelos de despliegue, así como los patrones de interconexión entre plataformas CC. Posteriormente, el apartado 2.4 analiza algunas de las plataformas más extendidas y, también, presenta un modelo de arquitectura de referencia. Finalmente, el apartado 2.5 muestra las vulnerabilidades y riesgos documentados en el estado del arte, que se analizarán en el apartado final (2.6) a modo de conclusión del capítulo.

2.1 EL PARADIGMA CLOUD COMPUTING

En términos históricos, el término CC fue utilizado por primera vez por el profesor Chellappa [Chellappa, 1997], quien ya avanzaba que el modelo de computación en el futuro estaría mucho más ligado a los intereses económicos, que a las limitaciones impuestas por la tecnología. Aunque hace más de una década, en un contexto con continuos avances tecnológicos esta afirmación resultaba utópica, hoy en día, la realidad demuestra que el desarrollo de la tecnológica está motivado por los intereses del mercado [Buyya *et al.*, 2008].

El concepto fue haciéndose popular de la mano de Salesforce.com Inc.[5], empresa que centraba su estrategia comercial en ofrecer *Software como Servicio* a grandes empresas [Weissman, 2009]. Su modelo de negocio era radicalmente diferente al del resto de empresas de la época. Así, Salesforce.com, en lugar de ofrecer paquetes *software* a medida, lo que comercializaba era acceso a *software* genérico desplegado en sus propios servidores y accesible a través de Internet [Cusumano, 2010]. Este nuevo modelo de negocio, pese a la innovación en su concepción, ya había sido enunciado previamente previamente por Carr [Carr, 2005].

El primer producto de Saleforce.com fue un CRM (*Customer Relationship Management*), que posteriormente daría lugar a la plataforma Force.com[6], lanzada en el año 2007. Esta plataforma puede considerarse el origen *conceptual* de otras plataforma ampliamente conocidas actualmente, como pueden ser Amazon AWS[7], Google App Engine[8], etc. Este nuevo mercado tecnológico se completó, en primer lugar, con el nacimiento de los servicios web al inicio de la década de los 2000 [Staab *et al.*, 2003], lo que permitió el desarrollo de una nueva arquitectura *software* basada en servicios [Newcomer *et* Lomow, 2004]. Así mismo, unido a esta nueva arquitectura *software*, Amazon[9] desarrolló entorno al año 2006 su plataforma Amazon Web Services (AWS) [Sotomayor, 2009] que incluía un novedoso servicio de infraestructura, conocido como Amazon S3[10], basado en la tecnología de virtualización; así como un servicio de computación distribuida basados en el concepto de *Grid Computing* [Bote-Lorenzo *et al.*, 2004] (Amazon EC2[11]).

[5] http://www.salesformace.com
[6] http://www.force.com/
[7] http://aws.amazon.com/
[8] https://appengine.google.com/
[9] http://www.amazon.com/
[10] http://aws.amazon.com/s3/
[11] http://aws.amazon.com/ec2/

Muchos años atrás, desde un punto de vista de investigación, fue IBM[12] quien avanzó, a través del documento *Autonomic Computing Manifesto* [Horn, 2001] las directrices que guiarían el modelo computacional (auto-coonfiguración, auto-monitorización, auto-optimización, etc.). Partiendo de estas directrices, diferentes empresas (Yahoo[13], Amazon, eBay[14] y Microsoft[15]), también investigaron ampliamente en el concepto entre los años 2003 y 2007, con resultados muy dispares, donde la mayoría de aproximaciones estuvieron centradas en la capa de infraestructura [Ogrizovic *et al.*, 2010].

El desarrollo del paradigma CC tal y como hoy lo conocemos, tanto a nivel tecnológico como de comercialización, fue desarrollado por el consorcio formado 2007 por Google e IBM junto con 6 prestigiosas universidades americanas: *Massachusetts Institute of Technology* (MIT), *Standford University*, *University of California*, *University of Berkeley*, *University of Meryland* y *University of Washington* [Lohr, 2007]. Este esfuerzo, se debe a la ventaja competitiva que implica dominar este marco tecnológico, poseer una plataforma propia permite por un lado, (i) al *proveedor Cloud* ofrecer sus servicios siguiendo un modelo de pago por uso [Buyya *et al.*, 2008] [Erdogmus, 2009], siguiendo los principios propuestos por el concepto *Utility Computing* [Ross *et* Westerman, 2004]. Y por otro lado, (ii) al *usuario Cloud*, le permite no tener la obligación de provisionarse con recursos computacionales adicionales para contener los picos en la demanda de sus productos o servicios, ni tampoco disponer de la infraestructura para proporcionarlos; transformando los gastos de capital en gastos operacionales [Armbrust *et al.*, 2010].

[12] http://www.ibm.com/
[13] http://www.yahoo.com
[14] http://www.ebay.com/
[15] http://www.microsoft.com/

2.2 TECNOLOGIAS RELACIONADAS

No cabe duda que el desarrollo de este paradigma computacional CC ha sido motivado por el rápido desarrollo de un conjunto de tecnologías sobre las que se sustenta para ofrecer el catalogo de servicios computacionales bajo demanda. A continuación se presentarán de forma resumida el conjunto de tecnologías precedentes, así como su aportación individual al paradigma CC [Wang *et al.*, 2008] [Zhang *et al.*, 2010] [Buyya *et al.*, 2009]:

- *Grid Computing*. Es un paradigma de computación distribuida cuyo objetivo es coordinar recursos computacionales y de red para satisfacer un objetivo [Zhang *et al.*, 2010]. El desarrollo de este paradigma distribuido fue motivado para satisfacer las necesidades computacionales del *software* científico. Las características principales [Bote-Lorenzo *et al.*, 2004] de este paradigma son la heterogeneidad, reparto de recursos, múltiples administradores, acceso consistente y transparente, y finalmente, distribución geográfica a gran escala.
  Todos estos rasgos identificadores están orientados a satisfacer los objetivos a nivel de aplicación, y también son compartidos por el paradigma *Cloud*. Sin embargo, CC avanza sobre estos principios para proporcionar un entorno *hardware* virtualizado a diferentes niveles (*hardware* y plataforma) gracias al cuál se puede realizar un aprovisionamiento automático de recursos (*pooling*). Dentro de CC, se habla de Big Data [Agrawal et al., 2001] en lugar de Grid Computing cuando se refire al análisis de grandes cantidades de información.
- *Utility computing*. Es un modelo de aprovisionamiento de recursos computacionales (procesamiento y almacenamiento) en el que el proveedor del servicio suministra los recursos a medida que los usuarios hacen uso de los mismos [Ross *et* Westerman, 2004].
  Dentro del concepto es mucho más relevante los aspectos micro y macroeconómicos, que los tecnológicos en sí. Así, la literatura existente [Rappa, 2004] [Ross *et* Westerman, 2004] [Broberg *et al.*, 2008], trata de justificar los beneficios, en términos de ahorro de costes, que obtiene una empresa cuando externaliza sus necesidades computacionales, refiriéndose habitualmente al centro de proceso de datos.
  Este modelo desde el punto de vista del mercado sigue una dirección paralela al paradigma CC, salvando la distancia temporal entre ambos. Pero, a nivel tecnológico, *Utility Computing* centra sus esfuerzos en el sector empresarial; mientras que CC abarca también al gran público debido a que no centra sus esfuerzos en la externacionalización del centro de proceso de datos, sino en cualquier tipo de servicio computacional, con independencia de la escala a la que vaya a ser suministrado.

No se encuentran, en el estado del arte, plataformas que provean servicios computacionales siguiendo un modelo de utilidad a gran escala, por lo que se puede afirmar, que CC es la implementación a gran escala de los principios propuestos por *Utility Computing*.

- *Autonomic computing* [Kepahart *et* Chess, 2003] [Horn, 2001] [Parashar *et* Hariri, 2005]. Aunque no es un término tan conocido como pueden ser los anteriores, resulta clave a nivel computacional dentro del paradigma que nos incumbe. Las directrices de esta tecnología determinaban que los sistemas computacionales fueran capaces de auto-gestionarse y reaccionar al entorno (interno o externo) sin la necesidad de la intervención humana. Aunque esta tecnología sólo fue desarrolla a nivel teórico, en la actualidad no cabe duda de que ha sentado las bases sobre las que se han construido el paradigma computacional CC.
- *Virtualization Technology* [Anderson *et al.*, 2005]. Es la tecnología que más ha favorecido al rápido desarrollo del paradigma CC. Permite crear abstracciones *hardware* de forma dinámica y automática en forma de máquinas virtuales sobre el *hardware* disponible [Barham *et al.*, 2003]. Además, los últimos avances permiten variar los recursos asignados a cada uno de los nodos virtualizados e incluso migrarlos entre diferentes equipos físicos sin tener que detenerlos [Oguchi *et* Yamamoto, 2008]. Así mismo también permite crear y destruir nodos de una red.

  Gracias a estas nuevas y potentes características, la virtualización facilita la gestión de los recursos *hardware*, tarea que en el pasado era altamente compleja, siendo un proceso manual y lento, que habitualmente requería personal altamente especializado. Destacar, en último lugar que gracias a que el usuario final, no tiene acceso a los recursos físicos, sino a la capa superior virtualizada, es posible presentarle una visión ilimitada de los recursos disponibles.

  Como única desventaja de esta tecnología, se observa que la gestión del entorno virtualizado requiere de un *Hypervisor*, que es el módulo que gestiona las abstracciones *hardware* en cada nodo físico para lo cuál consume recursos [McDougall *et* Anderson, 2010]. No obstante, este consumo computacional depende del modelo de virtualización elegido. En los últimos modelos, este sobrecoste no supera el 2% de la potencial computacional del entorno físico [Che *et al.*, 2008] [Che *et al.*, 2010], aunque para ello requieren *hardware* dedicado, algo que también está ampliamente implantado gracias a las tecnologías como INTEL-VT y AMD-V [Chen *et al.*, 2008].

  En la actualidad existen varios sistemas de virtualización que utilizan diferentes técnicas para minimizar el consumo de recursos, destacando Xen [Barham *et al.*, 2003], KVM [Kivity *et al.*, 2007], VMWARE [Rosenblum, 1999], OpenVZ [Mirkin *et al.*, 2008], etc.

- *High Availability Computing*. El nacimiento de la tecnología de virtualización [Oguchi *et* Yamamoto, 2007] y la organización de los centros de procesos de datos en clústers de tipo HPC [Bianchini *et* Rajamony, 2004], ha propiciado el desarrollo de nuevas técnicas de alta disponibilidad [Petrocci, 2010], que van mucho más allá de las vetustas técnicas basadas en la replicación de elementos *hardware* [Gray *et* Sieweorek, 1991].
  CC toma los principios de esta tecnología en tanto en cuanto los servicios computacionales ofrecidos se tienen que ofrecer sin interrupciones. Sin embargo, CC avanza en este sentido, ya que los parámetros de calidad de los servicios también deben ser constantes con independencia en la demanda de los mismos [Andrzejak *et al.*, 2010], asociados a un acuerdo calidad de servicio (SLA)
- *Service Oriented Architecture*, *Service Flow* y *Workflow*. Como ya se ha comentado, el nacimiento de los servicios web [Staab *et al.*, 2003] y las arquitecturas orientadas a servicios (SOA) [Newcomer *et* Lomow, 2004] han contribuido al desarrollo del paradigma CC, especialmente en la capa de los servicios de plataforma, en la que se exponen un conjunto API (*Application Programming Interface*) a los programadores cuya funcionalidad es muy heterogénea.
  La investigación en CC en este sentido, es la misma que la que se venía desarrollando en el paradigma anterior (SOA) y está relacionada con la composición de servicios de forma automatizada. [Tai *et al.*, 2010], tema en el que se ha venido trabajando recurrentemente en los últimos años [Dustdar *et* Schereiner, 2005].

Finalmente, el desarrollo de la Web 2.0, Internet del Futuro [Shenker, 1995] y las nuevas técnicas de programación web enriquecida [Fraternali *et al.*, 2010] han contribuido también en gran medida al notable crecimiento del paradigma CC. Estas tecnologías han evolucionado enormemente en los últimos años [Tan *et* Wang, 2010], permitiendo realizar tareas que antes estaban limitadas al *software* tradicional de escritorio. Esto ha facilitado que el uso de los productos *software* ofertados como servicio lleguen al gran público, lo que sin duda contribuye en gran medida a que las grandes compañías apuesten por el desarrollo del paradigma.

En la Tabla 1 se presenta un cuadro resumen de cada una de las tecnologías y su aportación (a nivel *hardware*, *software* o Negocio), así como una descripción de la influencia en el desarrollo de paradigma *Cloud*.

	Hard.	Soft.	Neg.	Influencia
Grid Computing	X			Computación de altas prestaciones al servicio de los usuarios finales
Utility Computing			X	Origen del modelo de negocio del paradigma CC, basado en la provisión de servicio computacionales bajo demanda
Autonomic Computing		X		Define la bases de la auto-monitorización y auto-control del entorno computacional
Virtualization	X			Entorno *software* subyacente que hace posible la provisión rápida de servicios.
High Availability	X	X		Agrupamiento de centros de datos en clúster y técnicas de alta disponibilidad
SOA, Services flow		X		Modelo de provisión de servicios computacionales a nivel *software/hardware*.

Tabla 1.- Relación Cloud Computing y tecnologías asociadas

2.3 CLOUD COMPUTING HOY EN DÍA: DEFINICIÓN

El concepto CC se ha ido afianzando tanto en el plano empresarial, como en el de la investigación. Por lo que han ido surgiendo una gran variedad de definiciones [Mell *et* Grance, 2011] [Vaquero *et al.*, 2009] [Wang *et al.*, 2008] [Foster *et al.*, 2008] que se presentan a continuación. En cada una de ellas, los autores tratan de resaltar aquellas características que a su juicio son más relevantes. Al analizar un amplio conjunto de trabajos realizados sobre el tema, se puede distinguir dos grandes grupos:

a) Aquellos que centran sus intereses en la definición de los aspectos tecnológicos, diferenciando a su vez a los autores que enfocan la definición a las características de la infraestructura subyacente y otros a los servicios que se ofrecen. Es decir, a las características a nivel interno y externo, respectivamente.
b) Aquellos cuyo interés es destacar los aspectos relacionados con el modelo de negocio que intrínsecamente está asociado con el entorno *Cloud*.

A lo largo de este apartado, se presentarán de forma crítica las definiciones que más interés han despertado por parte de la comunidad científica y empresarial, con el objetivo de acotar claramente el término, analizando y discerniendo qué se entiende hoy en día por *CC*.

En el año 2007, junto con el nacimiento de esta tecnología, muchos autores y directivos de importantes compañías tecnológicas dieron su particular visión sobre CC. La mayoría de estas definiciones no eran precisas, es más, muchas de ellas ni siquiera se acercaban a lo que hoy en día conocemos como modelo de computación en nube. Había los que indicaban que (i) no era una tecnología novedosa [Rapport, 2010], o incluso (ii) confundían este paradigma con tecnologías anteriores (*Grid Computing*, *Utility Computing*, *Autonomic Computing*, etc.) [Rapport, 2010], o simplemente en la mayoría de los casos ofrecían una definición parcial, tal como se presentará a continuación. Tal era el desconcierto sobre esta nueva tecnológica, que algunos como Irving Wladawsky-Berger[16] (IBM) indicaban en el año 2010 que "*there is a clear consensus that there is no real consensus on what cloud computing is*". Teniendo en cuenta que CC nació en el año 2007 y los grandes esfuerzos que se pusieron en su desarrollo, resulta destacable que en el año 2010, todavía no hubiera un consenso claro sobre esta tecnología, debido sobre todo a que era difícil observar el cambio en el modelo computacional, y más aún en el modelo de

[16] Blog de Irving Wladawsky-Berger (IBM). http://blog.irvingwb.com/blog/2009/04/cloud-the-emergence-of-a-new-model-of-computing.html

prestación de servicios que conlleva la adopción de este paradigma tecnológico.

Más allá, de las definiciones dadas a nivel empresarial, a nivel académico las primeras definiciones formales sobre CC empiezan a surgir en el año 2008. Así, una de las primera definiciones fue propuesta por Wang *et al.*, la cuál estaba claramente enfocada a las características tecnológicas del paradigma [Wang *et al.*, 2008]:

A computing Cloud is a set of networks enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing platform on demand, which could be accessed in a simple and pervarsive way.

De la definición dada por Wang *et al.* cabe resaltar que los autores ya enfatizaban la idea conceptual de ofrecer servicios escalables y personalizables a través de Internet, asegurando un nivel de calidad mínimo en los mismos. Lo que denota tres características clave del paradigma CC (escalabilidad, personalización y nivel en la calidad del servicio). Estas características, en ningún caso, suponían un reto tecnológico para la época. Sin embargo, la particularidad viene dada por el hecho de que ya se cita que los servicios se exponen bajo demanda siguiendo los principios del paradigma *Utility computing* [Ross *et* Westerman, 2004], que ya se ha presentado en el apartado anterior. Por el contrario, como crítica a la definición destacar que, en primer lugar, (i) aunque intrínsecamente se habla de escalabilidad, lo que pueden asociarse a un concepto más novedoso como es la elasticidad [Chiu, 2010], no se habla de que ésta se gestione de forma automática. Tampoco, se menciona (ii) nada acerca de los posibles tipos de servicios ofertados, es decir, todavía no se mencionaba la posibilidad de ofrecer servicios de infraestructura a nivel *hardware* algo que diferencia este modelo de las tecnologías anteriores. Además, tampoco se destaca que (iii) el nivel de calidad puede ser establecido mediante un acuerdo SLA, lo que puede implicar implícitamente la escalabilidad automática del sistema para asegurar que se satisfacen los diferentes SLAs acordados. Y, finalmente, y como crítica más importante, (iv) no se hace ninguna mención al modelo de distribución ni comercialización del *hardware* o *software* ofrecido a través de servicios CC, que es una de las bases del paradigma.

Al mismo tiempo, otros autores introducen definiciones más completas, como es el caso de Foster *et al.* que presenta la siguiente definición sobre el paradigma de computación en nube [Foster *et al.*, 2008]:

A large-scalle distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dinamically-scalable, managed computing power, storage, platforms, and service are delivered on demand to external customers over the Internet.

En este caso se trata de una definición adecuada y precisa a nivel tecnológico. Se enfatiza la importancia de los servicios ofertados a través de Internet dentro del paradigma. La definición relaciona explícitamente estos servicios con el entorno *hardware* subyacente, destacando que es un entorno distribuido a gran escala, e introduciendo tecnologías muy novedosas como en su momento era la virtualización, que permite la abstracción de recursos *hardware*, la gestión de la capacidad de cómputo, etc. Se destaca que se habla de gestión dinámica (y, quizás, automática) de los propios servicios. Además se puede sobreentender que se habla de servicios de plataforma e infraestructura, pero no así de servicios *software*. Finalmente, se hace referencia a un modelo computacional cuyo crecimiento está motivado por la economía de escala, pero sin embargo, no dan ninguna información más sobre su modelo de comercialización.

Por su parte, Vaquero *et al.* introducen la que quizás es una de las definiciones más completas, ya que incluye aspectos tecnológicos y de negocio [Vaquero *et al.*, 2009]:

Cloud are a large pool of easily usable and accesible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scalable), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLA.

La primera parte de la definición se centra en los aspectos tecnológicos, presentando características que también consideran otros autores (virtualización, escalabilidad, etc.), y adicionalmente destaca que los recursos computacionales que se ofrecen en forma de servicio pueden ser de cualquier tipo y, que además, tienen que ser fácilmente accesibles a través de Internet. Esta definición es de las primeras que introduce la necesidad de reajuste automático del sistema en función de la demanda, lo que asegura un óptimo uso de los recursos *hardware*. Es decir, estos autores presentan las características que diferencian al paradigma CC a nivel tecnológico de los modelos anteriores, la elasticidad de los servicios, y por consiguiente, la necesidad de reajustar los recursos en función de la demanda de los mismos consiguiendo por tanto, un uso óptimo de la infraestructura *hardware*. En la segunda parte de la definición, se indica que los recursos son explotados a través de un modelo de comercialización basado en el establecimiento de un acuerdo personalizado de la calidad del mismo (SLA) y que a su vez permite fijar el precio del servicio. Es decir, debido a este acuerdo, los proveedores deben asegurar un nivel de calidad mínimo en el servicio [Buyya *et al.*, 2008], o QoS. Este hecho, ha provocado, que los servicios, no se ofrezcan de forma tradicional, sino siguiendo un modelo de pago por uso, conducido por SLA

alcanzado previamente, que fija de forma clara el precio del producto y la calidad mínima del servicio. En esta definición, pese a que habla de diferentes tipos recursos/servicios ofertados, parece que los autores enfocan la definición tan sólo a los aspectos relacionados con la infraestructura *hardware* y plataforma en parte, debido al auge que la tecnología *Grid computing* [Bote-Lorenzo *et al.*, 2004] [Foster *et al.*, 2008] tenía en aquel momento. Además, también parece que no se le atribuye una gran importancia los servicios en las capas superiores del paradigma, es decir, al *software* ofrecido como servicio.

En contraposición, para otros autores las capas superiores constituyen la clave que ha permitido que este paradigma haya adquirido la popularidad que hoy en día atesora [Erdogmus, 2009] [Miller, 2008]. Además, la realidad hoy en día muestra que una gran cantidad de plataformas CC se orientan únicamente a ofrecer servicios dentro de las capas superiores del paradigma. En este sentido, existen incluso autores que centran su definición en este tipo de servicio. Así, por ejemplo Brag [Bragg, 2008] indica que *the key concept behind the Cloud is the Web application - software stored on some company's server and accessed by users through their browsers. That means everything is done through the same window used to surf Web sites. Most of the software used and the data produced remains in the Cloud, i.e., on the main servers of the company.*

Finalmente, a finales del año 2011, el NIST (*National Institute of Standards and Technology*) americano [Mell et Grance, 2011], propone la definición que a nuestro juicio es la más acertada desde un punto de vista técnico y funcional. Aunque, en ella no se presta especial atención a la economía de escala que ha promovido el desarrollo incesante de este modelo computacional.

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

En este caso los autores, además de referirse a cualquier tipo de recursos computacionales, indican un concepto clave: la reducción al mínimo el esfuerzo de obtener los servicios proporcionados; algo que, sin duda, ha sido clave en el gran auge que ha tenido estáatecnología.

Como complemento, esta definición incluye 5 características obligatorias que cualquier entorno CC debería poseer:

- *Servicios automáticos bajo demanda*, refiriéndose a que los servicios, con independencia de su tipo, se tienen que proporcionar automáticamente y sin interacción humana en función de la demanda de los usuarios.

Estas características de simplicidad y automaticidad, ya la habían apuntado números autores de forma previa, como se ha visto a lo largo de este trabajo. Pero como crítica, no se habla de la necesidad de alcanzar un acuerdo en la calidad del servicio, lo que quizás constituye en sí mismo una de las mayores debilidades en la definición.

- *Disponibilidad de servicios a través de la red*, los clientes deben acceder a los servicios a través de la red, y por lo tanto, los proveedores tienen que utilizar este medio para proporcionar sus servicios.
- *Disponibilidad de recursos*, el proveedor debe disponer de la capacidad de ofrecer los servicios con independencia de la demanda de éstos, utilizando recursos *hardware* físicos o virtuales asignados dinámicamente a cada servicio y reasignados en función de la demanda. Esta característica implica que el usuario no conoce dónde está su información en cada momento, ya que existe un principio de no localidad (o deslocalización), donde los datos se trasladan entre los recursos virtuales (o no) que tienen asignados en cada momento.
  En este sentido, existen autores como [Zhang *et al.*, 2010] [Buyya *et al.*, 2009] que hablan directamente de servicios de alta disponibilidad, siguiendo un modelo de computación de alta disponibilidad.
- *Elasticidad*. Los diferentes recursos se deben proporcionar de forma elástica e incluso automáticamente en función de la demanda.
  Esta característica, está directamente ligada con la anterior, la alta disponibilidad de los servicios, lo que implica que el proveedor debe asignar los recursos en función de la demanda de los servicios, algo que ya se venía haciendo en tecnologías anteriores. Sin embargo, CC ofrece servicios elásticos, esto significa, que los recursos asignados a cada servicio varían automáticamente en función de la demanda, lo que implica proporcionar más recursos cuando más demanda existe, pero también reducir recursos cuando disminuye la demanda. Todo este proceso se debería realizar de forma automática sin que el cliente de los servicios tenga conocimiento de la readaptación interna. Gracias a esta característica, los usuarios tienen la sensación de interactuar ante un sistema con recursos ilimitados, aunque en realidad no sea así.
  La existencia de niveles de acuerdo en cuanto a la calidad del servicio, introduce una complejidad mayor en la gestión de la elasticidad, ya que no sólo es necesario proporcionar mayores recursos a más demanda en los servicios, sino que el tiempo necesario para la readaptación está limitado en base a la calidad acordada.
- *Servicios a medida*. Los servicios proporcionados por un sistema CC deben estar totalmente monitorizados, y su control tiene que realizarse de forma automática.
  Gracias a ello, se puede optimizar el uso de recursos; en este sentido, sólo teniendo un conocimiento total sobre el propio sistema, es posible

ofrecer un servicio de pago por uso eficiente y competitivo. Permitiendo ofrecer una transparencia total de la información entre el proveedor del servicio y el usuario del mismo.

Estas características, junto a la definición previa, acotan en gran medida el concepto que hoy se tiene de un entorno de CC y de los servicios que éste ofrece, los cuáles se conocen habitualmente como *Cloud Services* [Sosinsky, 2011]. No obstante, casi parece lógico, teniendo en cuenta que la última versión de la definición es relativamente reciente (septiembre del año 2011). Lo que ha provocado que se hayan ido eliminando los vestigios de tecnologías predecesoras a CC; aunque en algunos casos, estén relacionadas con éste nuevo paradigma (*Utility computing*, *Service Computing*, *Data Centers*, *Market-oriented computing*, etc.), en otros casos el concepto se ha ido alejando (*Grid Computing*, *P2P Computing*, etc.).

Sin embargo, como debilidades en la definición, en cuanto a las características *hardware*, destacar que por un lado, el NIST menciona la elasticidad dinámica, incluyendo como rasgo opcional la automatización de esta característica. Desde nuestro punto de vista, y el de algunos autores [Zhang *et al.*, 2010], esta es una de las características clave del entorno CC, siendo objetivo de estudio a lo largo de este trabajo. Y por otro lado, según [Zhang *et al.*, 2010] [Zhang *et al.*, 2011] también hablan del principio de "multi-tenencia" (*multitenant*), el cuál se refiere a que pueden existir varios entornos CC en un mismo centro de proceso de datos, lo que obliga a una clara división de responsabilidad en la gestión del entorno.

La definición del NIST sin duda es la más acertada y completa, aunque podría completarse incluyendo explícitamente el concepto de SLA, algo que sin duda ayudaría a completar todos los aspectos comerciales y negocios del paradigma [Buyya *et al.*, 2008]; dado que además existe una gran cantidad de estudios centrados en este campo. Finalmente, la definición y características de CC propuestas por el NIST, pueden complementarse mediante el conjunto de características propuestas por Zhang *et al.*: *Multi-tenancy, Shared resource pooling, Geo-distribution and ubiquitous network Access, service oriented, dynamic resource provisioning, self-organizing y Utility-based pricing* [Zhang *et al.*, 2010]. Dentro de estas características, cabe destacar la última, que es una propiedad estrechamente relacionadas con los aspectos económicos que envuelven al paradigma. El precio del servicio está basado en su utilidad, es decir, dado que CC emplea un modelo de pago por uso, el precio concreto de un servicio, depende del propio servicio y del uso que el cliente hace de este, basándose en la teoría clásica de la regulación (oferta y demanda) [Peltzamn, 1976]. Lo que a priori logra reducir el coste operacional debido a:

- No existe inversión inicial, ya que al implementar un modelo de negocio de pago por uso, desde el punto de vista del usuario de servicios CC no

es necesario realizar una inversión previa en infraestructura, sino que se alquilan un conjunto de servicios acordes a las necesidades en cada momento.

- Reducción de los riesgos comerciales y los costes de mantenimiento. Dado que siguiendo el modelo CC, los recursos computacionales no se encuentran ligados a la empresa, los costes derivados de su mantenimiento, actualización, fallos, desconexión por actualización, etc. no constituyen un riesgo empresarial para la empresa.

A lo largo de este apartado se han presentado diferentes definiciones sobre CC que denotan la evolución del paradigma tecnológico desde su nacimiento en 2007. En primer lugar destaca el interés que existe tanto en la comunidad científica, como en el entorno empresarial acerca de este paradigma computacional. Así mismo, también es reseñable la estrecha relación que existe con tecnologías previas, así como que la unión de todas ellas en una plataforma única, lo que da como resultado un entorno tecnológico mejorado. Finalmente, la clave en el desarrollo y la rápida introducción en el mercado ha sido el modelo de comercialización basados en acuerdos específicos de uso, que resultan favorables tanto para los proveedores, como para los consumidores del servicio.

En el modelo de negocio, las tecnologías implicadas y los servicios ofertados, hacen que un sistema CC sea complejo y en algunos aspectos difícil de comprender. Por lo que en los siguientes apartados se presentará con más detalle los tipos de servicios ofertados, los modelos de despliegue posibles y roles de de usuario implicados en el modelo de comercialización.

2.3.1 CAPACIDADES OFERTADAS, SOMETHING AS A SERVICE

Como se indicó anteriormente, en la definición del NIST [Mell et Grance, 2011] proponen tres tipos de servicios y cuatro modelos de despliegue, ampliamente conocidos. Se comenzará describiendo los modelos de servicio, en este sentido dado que se ofrece casi cualquier tipo de servicio computacional, ya sea *hardware* o *software* y siempre a través de Internet, es habitual hablar de *Something as a Service* [Talia, 2011] [Rimal *et al.*, 2009]. No obstante, antes de enumerar estos servicios, resulta clave indicar que en la propia definición se habla de *capabilities* o capacidades, en lugar de servicios. Este concepto es quizás mucho más intuitivo, alejando estos servicios computacionales del concepto tradicional de servicio en las arquitecturas anteriores, como por ejemplo SOA [Jones, 2005].

Según el NIST, los principales modelo de servicios ofrecidos son los siguientes:

- *SaaS. Software as a Service.* Este tipo de capacidad permite al proveedor, ofrecer al consumidor sus aplicaciones, de forma que éstas se ejecutan directamente en la infraestructura en nube. Lo que conlleva una gran cantidad de ventajas como son la ubicuidad de las aplicaciones o el uso de clientes ligeros. Sin embargo, también lleva asociadas un conjunto de dificultades directamente relacionadas con que el hecho de que el consumidor pierde el control sobre la infraestructura (red, almacenamiento, sistema operativo, dificultad de configuración, etc.), aunque quizás desde un punto de vista más moderno, pueden ser incluso consideradas como fortalezas.
  Dentro de este tipo de servicio, algunos autores como [Lenk *et al.*, 2009] proponen una división en *Applications* y *Applications Services*, dado que existen numerosos trabajos relacionados con la composición de servicios en entornos *Cloud* [Wei, 2010] [Gutierrez-Garcia *et* Sim, 2010].
  Ejemplos de servicios: gestión de contenido, correo electrónico, herramientas de productividad, CRM, gestión de ventas, ERP, etc.
- *PaaS. Platform as a Service.* Este tipo de capacidades proporcionadas por los proveedores, permiten al consumidor disponer de las herramientas necesarias para crear sus propias aplicaciones. Entre estos servicios se encuentran entornos de programación, librerías, herramientas, etc. De la misma manera que los servicios de la capa anterior, el programador no controla la infraestructura subyacente, ni el entorno de despliegue de sus aplicaciones.
  Este nivel puede subdividirse según [Lenk *et al.*, 2009] en *Programming environment* y *Execution environment*, separando de este modo los servicios ofrecidos dentro del entorno de despliegue, del paradigma de computación sobre el que son desarrollados.
  Ejemplos servicios: business Intelligence, base de datos, despliegue de aplicaciones, entorno de desarrollo y prueba, etc.
- *IaaS. Infrastructure as a Service* (o *Hardware as a Service* –HaaS– para [Wang *et al.*, 2008]). El tipo de capacidad que se proporciona al consumidor es de tipo *hardware*, es decir, procesamiento, almacenamiento, red, etc. Al igual que en servicios anteriores, el consumidor o usuario no tiene el control sobre la infraestructura subyacente, en este caso, sistemas operativos nativos, características de red, etc. Aunque sí que puede tener la configuración sobre el entorno *hardware* virtualizado que se le presenta.
  Dentro de este tipo de servicio, [Lenk *et al.*, 2009] separa el entorno *hardware* subyacente de los servicios de infraestructura ofertados:
  - *Resource Set*, representan los recursos computacionales que posee el entorno CC, es decir, el conjunto de computadores físicos. Constituye la visión interna del entorno de computación, encargado de ofrecer potencia computacional a las que ofrece el

entorno CC a nivel externo. Su objetivo es facilitar la autogestión de los recursos, proporcionando una visión externa de recursos ilimitados. La gestión de esta capa debe realizarse de forma dinámica (y preferiblemente automática), optimizando el uso de los recursos, tal y como indicaba [Vaquero *et al.*, 2009]. En este sentido, en la actualidad, existen diferentes investigaciones centradas en maximizar la eficiencia de este capa subyacente a todo el entorno CC [Liu *et al.*, 2009] [Heras *et al.*, 2012].

Esta capa, a su vez es dividida en dos subgrupos:

- *Physical Resource* Set (simplemente *Hardware* para [Zhang *et al.*, 2010]) es el propio *hardware* físico que proporciona los recursos computacionales al sistema.
- *Virtual Resource Set* habitualmente construida sobre una tecnología de virtualización subyacente. También es posible encontrar algún entorno CC que no utiliza virtualización, pero son casos aislados.

Esta división entre recursos físicos y virtuales tiene como principal ventaja el incremento en la capacidad de la gestión del entorno. Teniendo en cuenta que esta gestión tiene que ser realizada de forma dinámica y en la mayoría de los casos automática, resulta fundamental y uno de los aspectos diferenciadores de la tecnología CC. Sin embargo, también existen debilidades, la gestión de un entorno virtualizado requiere el uso de *Hypervisores* que gestionen y encapsulen su complejidad intrínseca, lo que induce una carga adicional de recursos computacionales [McDougall *et* Anderson, 2010].

- *Infraestructure Services*, divididos a su vez en (i) *Basic infrastructure services* englobando en este grupo servicios computacionales (almacenamiento, procesamiento, red, etc.); y (ii) *Higher infrastructure servic*e (o *Data as a Service* (DaaS) según [Wang *et al.*, 2008], y agrupa servicios estructurados de almacenamiento de información. Este segundo grupo es denominado
  - *Basic infrastructure services*, englobando en este grupo servicios computacionales (almacenamiento, procesamiento, red, etc.).
  - *Higher infrastructure Service*, englobando en este grupo servicios estructurados de almacenamiento de información. Es decir, en la práctica, las bases de datos.

Ejemplos servicios: almacenamiento, gestión de servicios, gestión de plataforma, servicios de computación, servicio de copia de seguridad, etc.

Finalmente, con respecto a las capacidades proporcionadas, añadir que algunos autores [Lenk *et al.*, 2009] introducen el concepto de *Human as a Service (HuaaS*), dentro de esta categorización de modelos de servicios que ofrecen las plataformas CC. Dentro esta categoría se enmarcan los servicios de recolección de información de forma masiva procedente de las bien conocidas plataformas sociales (Facebook, Twitter, etc.). Esta información es utilizada, principalmente, como complemento de otros servicios.

2.3.2 TIPOS DE USUARIOS Y MODELOS DE DESPLIEGUE

La definición propuesta en el NIST, presentada en el apartado 2.3, se complementa con 3 tipos de servicios (Apartado 2.3.1) y 4 modelos de despliegue que se describirán a continuación. De forma previa a enumerar los entornos de despliegue, resulta conveniente hacer referencia a los actores y roles que intervienen en el modelo de negocio CC. Así pues, en este apartado se presentarán en primer lugar los diferentes roles que intervienen en el paradigma CC, teniendo en cuenta las diferentes perspectivas existentes. Y, a continuación, se presentarán los modelos de despliegue propuestos por el NIST.

En la literatura, es posible encontrar varios trabajos sobre los roles o tipos de usuarios de un entorno CC [Vouk, 2008] [Buyya *et al.*, 2009] [Armbrust *et al.*, 2010] [Zhang et al., 2010] [Badget *et al.*, 2012]. Al analizar estos estudios, se observan dos vertientes que se presentan a continuación.

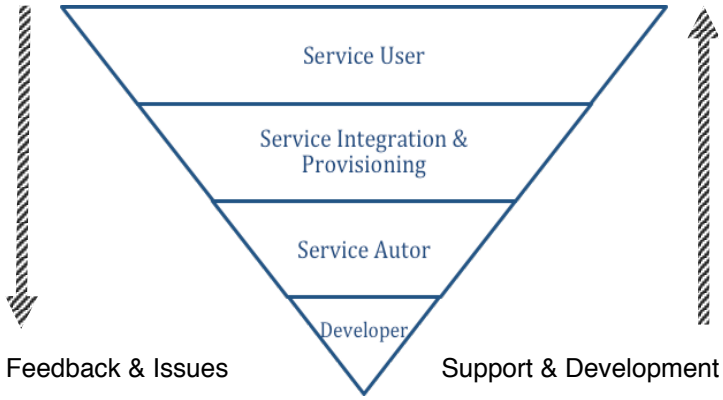

Figura 1.- Usuarios en Cloud Computing [Adaptada de Vouk, 2008]

Por un lado, a nivel técnico, siguiendo a [Vouk, 2008] se definen cuatro categorías de actores no excluyentes (Figura 1):

- *Service User.* Son los consumidores de servicios del entorno CC, su rol es el más importante y del que dependen el resto de usuarios.

- *Service Integration & Provisioning.* Expertos en composición de servicios, cuyo objetivo es satisfacer la demanda de un cliente concreto.
- *Service Author.* Responsables del desarrollo de servicios que puedan ser utilizados de forma individual, o como parte de otros.
- *Cyberinfrastructure developer.* Responsables del desarrollo y mantenimiento de la infraestructura.

Por otro lado, existe otra clasificación de roles que constituye un estándar, está más aceptada y constituye un estándar de facto. En ella, no sólo se tiene en cuenta la perspectiva técnica del despliegue *hardware* y *software*, sino también los intereses económicos que motivan el desarrollo del paradigma. Dentro de estos roles, se distinguen los siguientes, no siendo excluyentes ninguno de ellos:

- *Cloud provider.* Es el rol que proporciona capacidades *Cloud*. Normalmente, dentro de esta categoría están los servicios de tipo IaaS y PaaS. Fue propuesto inicialmente por [Armbrust *et al.*, 2010]. A su vez puede subdividirse en:
  - *Infrastructure Providers.* Identifica al rol que proporciona servicios *hardware*, es decir, IaaS. Propuesto por [Vaquero *et al.*, 2009].
  - *Service Provider.* Rol que proporciona servicios de tipo *software* en el modelo *Cloud*. Puede convertirse de forma simultánea en *Cloud User* si a su vez utiliza otros servicios de tipo SaaS, mediante la composición de servicios a este nivel. Propuesto por [Zhang *et al.*, 2010], aunque Armbrust *et al.* lo denomina *SaaS Provider* [Armbrust *et al.*, 2010].
- *Cloud User.* Este rol hace uso de los capacidades ofrecidas por el rol *Cloud provider*, con referencia a los niveles PaaS e IaaS. Fue propuesto por [Armbrust *et al.*, 2010], denominado *User/Broker* en [Buyya *et al.*, 2009].
- *SaaS User/End User.* Es el usuario final de las aplicaciones SaaS.

Además, [Buyya *et al.*, 2009] propone la existencia de un actor adicional, *SLA Resource Allocator*, entidad intermedia entre el entorno *Cloud* y los usuarios externos, con las funciones de monitorización de peticiones, control de admisión, gestión de precios, gestión de cuentas y monitor de infraestructura.

En el marco de investigación del NIST, se propone una división mucho más sencilla, en 6 actores que modelan las interacciones tanto a nivel de mercado, como técnico, integrando por tanto las dos perspectivas anteriores. Así pues, por un lado, se proponen tres actores desde un punto de vista de mercado: (i) *customer*, que es un usuario u organización que utiliza las capacidades que ofrece un entorno *CC*; (ii) *client*, que es una agente artificial que accede al entorno CC para utilizar y desplegar sus servicios y ofrecer productos al

anterior rol; y finalmente (iii) *provider*, que es una organización que proporciona las capacidades CC. Por otro lado, otros tres actores que modelan la relación entre el consumidor y el proveedor desde un punto de vista técnico: (iv) *Cloud broker* que es el intermediario que gestiona la relación comercial entre proveedor y consumidor; (v) *Cloud Carrier*, que proporciona la conexión desde el proveedor y el consumidor; (vi) *Cloud Auditor*, que evalúa de forma independiente los servicios ofrecidos en cuanto a rendimiento y seguridad.

Una vez que se han presentado los diferentes roles, siguiendo con la definición propuesta por el NIST, y por lo que componen un entorno CC, el último paso consiste en identificar los diferentes modelos de despliegue que existen, comenzando por los públicos y privados:

- *Private Cloud.* La infraestructura CC es utilizada por una única organización, que a su vez puede incluir diferentes consumidores.
  La infraestructura puede pertenecer a la propia organización, o ser ofrecida por un tercero.
- *Public Cloud.* Este tipo de infraestructuras se proporcionan para el uso abierto del público en general. Están alojadas por organizaciones, universidades, gobiernos o una combinación de ellos. Obligatoriamente debe existir el actor proveedor.

A partir de estos dos modelos de despliegue propuestos por el NIST, [Krautheim *et al.*, 2009] [Zhang *et al.*, 2010] proponen la existencia de un modelo híbrido denominado *Virtual Private Cloud*, también denominado *Outsources Private Cloud* por [Liu *et al.*, 2011] en el que se combinan entornos CC públicos y privados. De forma que se construye un entorno *Cloud privado* sobre las capacidades de infraestructura a modo de entorno subyacente que proporciona un entorno *public Cloud*. [Zhang *et al.*, 2010] define este tipo de despliegue como una plataforma situada por encima de los sistemas CC públicos, siendo la principal diferencia el hecho de que se permite el diseño de una topología específica y un modelo de seguridad específico, aunque en contrapartida el entorno CC esté deslocalizado.

Como consecuencia de la existencia de *Virtual Private Cloud*, se puede dar la situación de que varios entornos CC, que aún perteneciendo a organizaciones diferentes tengan que compartir la misma infraestructura, es decir *Cloud* público. En este caso se estaría hablando de multitenencia [Bezemer, 2010] [Mell *et* Grance, 2011] [Zhang *et al.*, 2010]. La investigación en los últimos tiempos también se ha dirigido al desarrollo de modelos que hagan posible esa convivencia, principalmente a nivel SaaS [Pervez *et al.*, 2010] [Kang *et al.*, 2011], y a nivel PaaS [Azeez *et al.*, 2010] [Mietzner *et al.*, 2011].

Finalmente, el NIST, también define otros dos modelos de despliegue, el de comunidad y el híbrido:

- *Community cloud.* La infraestructura del sistema CC es utilizada por un grupo de consumidores u organizaciones específicos que comparten intereses relacionados. La infraestructura puede estar alojada por una o varias organizaciones dentro de los grupos de interés (*on-site Comunity Cloud*), o incluso, puede pertenecer a un tercero (*out-source comunity cloud*).
- *Hybrid Cloud.* Este tipo de infraestructuras es la combinación de alguna de las anteriores, cada una de las cuáles tiene una entidad propia, pero trabajando de forma coordinada. La Figura 2 muestra un ejemplo de un sistema CC híbrido que se ha construido mediante la agregación de 5 variantes.

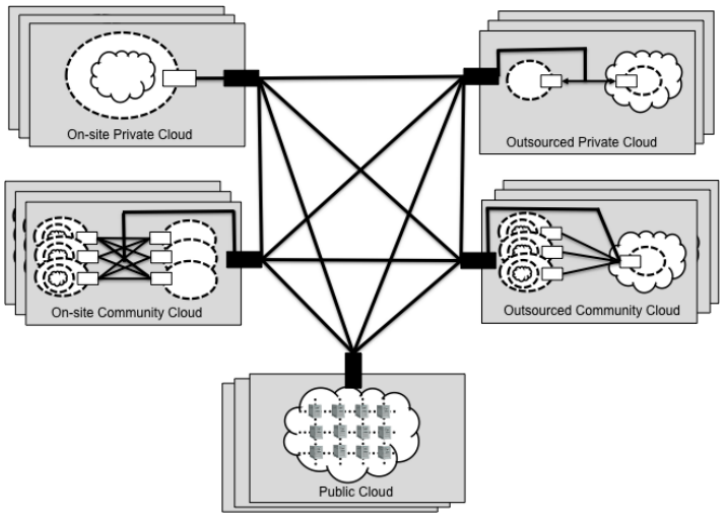

Figura 2.- Cloud híbrido [Liu *et al.*, 2011].

Cuando se habla de *Hybrid clouds* es necesario hablar de interoperabilidad entre plataformas [Ortiz, 2011], algo que también puede hacerse extensible al resto de despliegues CC. Un usuario de este tipo de entornos, habitualmente trabaja con diferentes plataformas al mismo tiempo, cada una de ellas tiene su propia interfaz de comunicación, habitualmente en forma de API, lo que las hace cerradas y dificulta su interoperabilidad con el resto de plataformas.

La dificultad en la interoperabilidad en cierto modo está basada en la falta de madurez de la tecnología, recordemos que nació hace tan solo 7 años. Sin embargo, también está motivada por los intereses que tienen las grandes empresas tecnológicas en imponer su plataforma CC sobre sus competidores.

En cuanto a la operabilidad, diferentes autores han propuesto dos clasificaciones diferentes atendiendo al nivel de portabilidad, aunque como se puede observar ambas tienen un gran paralelismo:

- Celestini *et al.* [Celesti et al., 2010a], habla de tres fases de ejecución para la consecución del objetivo de federación: (i) la *fase monolítica* que es aquella en la que los servicios son independientes y propietarios; (ii) la *fase de distribución vertical*, en la que los proveedores de servicios utilizan los servicios proporcionados por proveedores terceros, para proporcionar sus propios servicios a los consumidores; y, por último, (iii) la *federación horizontal*, en el que todos los proveedores con independencia de su tamaño disponen de servicios federados que ofrecen al consumidor siguiendo un modelo de economía en escala.
  Hoy en día nos encontramos a medio camino entre la fase 1 y la fase 2, donde la mayoría de sistema CC, sobre todo aquellos proporcionados por las grandes compañías, los cuáles son monolíticos (fase 1); pero al mismo tiempo, también empiezan a aparecer una gran cantidad de *Virtual Private Clouds* [Liu et al., 2011] que están directamente relacionados con la fase 2.
- Machado [Machado, 2009], considera tres tipos diferentes de escenarios. Aquel en el que (i) un usuario debe portar la configuración de sus servicios entre dos entornos CC diferentes. Otro sería en el que (ii) los usuarios tienen contratados servicios con diferentes servicios proveedores y desean que todos ellos trabajen de forma conjunta. Y finalmente, el último es el demonimado (iii) *cloud* federado, donde se ofrecen servicios al usuario final haciendo uso de los recursos de varios entornos CC, para lo que es necesario utilizar herramientas de coordinación entre ellos. La Figura 3 muestra una contextualización de este último escenario.

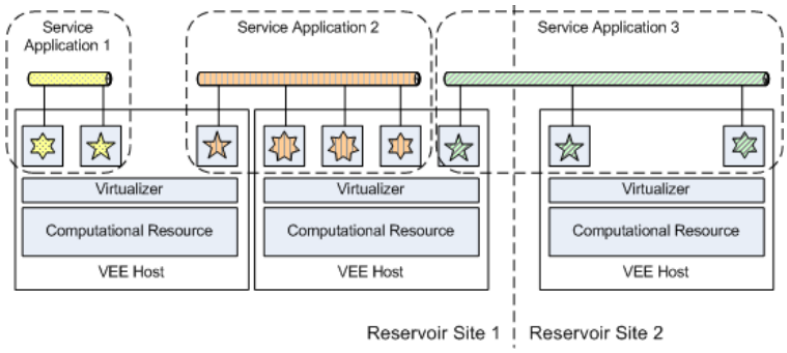

Figura 3.- Arquitectura en el proyecto Reservoir [Rochwerger *et al.*, 2009]

En la actualidad, para coordinar la investigación en el ámbito de la interoperabilidad, especialmente en la capa IaaS y la interoperabilidad entre sistemas CC [Loutas *et al.*, 2011] [Machado, 2009] [Petcu, 2011], comienzan a aparecer las principales iniciativas y organizaciones que pretenden desarrollar y facilitar la interoperabilidad entre plataforma, donde se destaca:

- *DMTF's Open Cloud Standard Incubator (OCSI)*[17], que se centra en el desarrollo de estándares de interacción entre proveedores y usuarios.
- *Open Cloud Computing Interface Working Group (OCCI-WG)* [18], que desarrolla especificaciones para la interoperabilidad de servicios de infraestructura.
- *Cloud Data Management Interface (CDMI)*[19], que trata de especificar funcionalmente como las aplicaciones crean, recuperan, actualizan y eliminan datos desde un entorno Cloud.
- *Global Inter-Cloud Technology Forum (GICTF)* [20], centro en la interoperabilidad entre plataformas.

[17] http://dmtf.org/standards/cloud
[18] http://occi-wg.org/
[19] http://www.snia.org/tech_activities/standards/curr_standards/cdmi
[20] http://www.gictf.jp/

2.4 MODELOS DE REFERENCIA: PLATAFORMAS Y ARQUITECTURAS

No cabe duda que un sistema CC es complejo y forma parte de un entorno abierto en el que se conjugan diferentes tecnologías, usuarios e intereses económicos para dar lugar a un nuevo modelo computacional que ha revolucionado la forma de ofertar servicios a través de Internet. Para que todos estos aspectos trabajen de forma coordinada con el objetivo de lograr unos objetivos comunes, y además teniendo que tener presentes las limitaciones técnicas existentes, se han desarrollado arquitecturas complejas.

Hasta hace no mucho tiempo las grandes compañías como IBM, Intel, Google, Amazon, etc. *no* mostraban información relevante sobre las arquitecturas de sus respectivos modelos CC. De hecho hoy en día el ocultismo en este aspecto sigue siendo la tónica general ya que, aunque muchas de ellas han hecho públicas las directrices generales, en ningún caso se han liberado algoritmos u otros aspectos técnicos o computacionales que permitan el avance de la tecnología sobre la base existente. Así por ejemplo, las Figura 4 y Figura 5 presentan la pila tecnológica de sistemas CC propuestas por Intel [Chahal *et al.*, 2010] y Cisco [Cisco, 2009], respectivamente. En ellas se pueden observar los componentes básicos de cualquier sistema CC como infraestructura subyacente (servidores, equipamiento de red, almacenamiento, etc.), así como componentes *software* (entorno de virtualización, gestión, monitorización, seguridad, etc.). Sin embargo, ninguna de estas compañías es un gran proveedor de servicios CC, por lo que ambas utilizan esta información para ofrecer sus productos a los verdaderos proveedores CC, los cuáles no publican sus arquitectura CC.

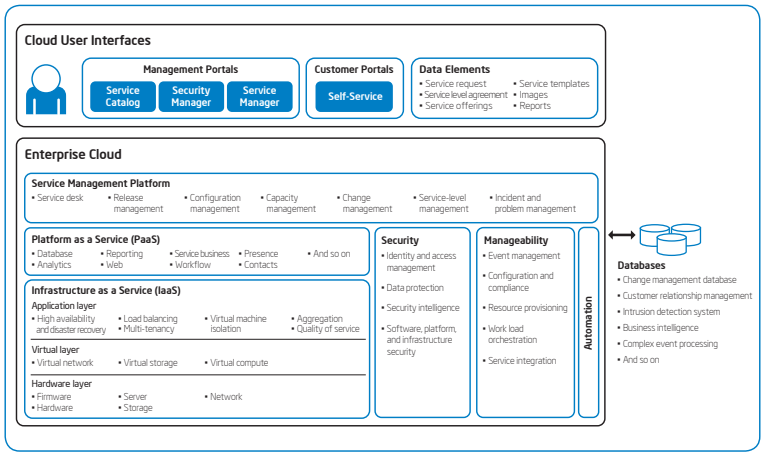

Figura 4.- Arquitectura Intel [Chahal *et al.*, 2010]

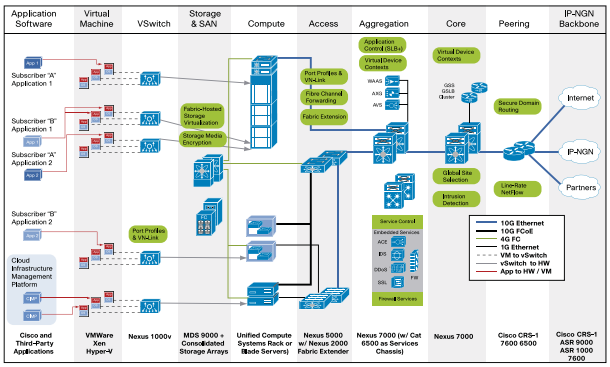

Figura 5.- Modelo de Arquitectura para Cisco [Cisco, 2009]

Siguiendo el trabajo realizado por el NIST, este organizanismo también propone una arquitectura de referencia [Liu *et al.*, 2011], que trata de describir y acotar correctamente todos los aspectos a tener en cuenta en un entorno CC. Esta arquitectura se presentará a continuación. No obstante, resulta interesante presentar de forma previa la clasificación que propone Tianfield [Tianfield, 2011] respecto a las arquitecturas CC:

- La arquitectura de plataforma, es aquella que tiene en cuenta el entorno tecnológico subyacente al paradigma CC
- La arquitectura de aplicación, es aquella que encapsula la gestión de la calidad de los servicios que se le ofrecen a los usuarios finales.

El modelo de referencia de arquitectura CC propuesto por el NIST [Liu *et al.*, 2011] aúna ambos modelos (según [Tianfield, 2011]), es decir, tecnología de plataforma y nivel de calidad hacia el usuario final, tal y como se muestra en la Figura 6.

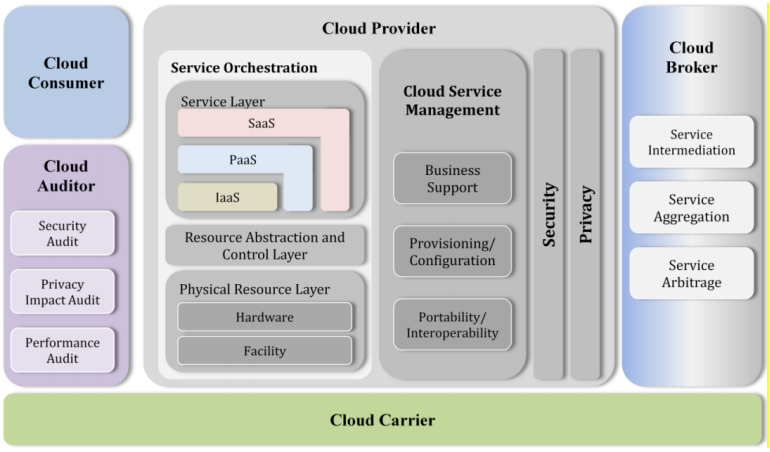

Figura 6.- Arquitectura Cloud de referencia para el NIST [Liu *et al.*, 2011]

La arquitectura que se presenta en la Figura 6 tiene en cuenta no sólo los componentes tecnológicos, sino también los roles que participan dentro del paradigma CC. Así, en función de cada uno de los roles del sistema, tenemos los siguientes módulos o componentes:

- *Cloud Provider*. Es el rol encargado de la (i) *Coordinación de Servicios*, proporcionando éstos a terceros, con independencia de su tipo (IaaS, PaaS o SaaS), por lo que necesita gestionar la infraestructura subyacente (física y virtual) a través de una capa de control. Así mismo, también debe (ii) *facilitar la gestión de esos servicios* ofertados a través de una capa de soporte a la comercialización y negocio que facilite la monitorización acerca del uso de esos servicios (características de calidad y acuerdos en el nivel del servicio) y su cobro en función del uso que se realice (facturación, consumos, etc.), así mismo también deberá proporcionar características de portabilidad e interoperabilidad entre plataformas, y un modelo que permita la rápida configuración y la gestión del cambio de aprovisionamiento de los recursos. Finalmente, también tendrá que hacerse cargo de la (iii) *seguridad* y (iv) *privacidad*, que son características clave para que un *Cloud Provider* sea aceptado por los consumidores de sus servicios.
- *Cloud Auditor.* Es un agente externo que capaz de monitorizar el servicio con el objetivo de validar si se están cumpliendo los requisitos acordados mediante SLA de seguridad, privacidad y rendimiento.
- *Cloud Broker.* Es un agente externo que actúa como intermediario entre consumidores y proveedores con el objetivo de buscar y proporcionar los servicios más adecuados a los objetivos de los consumidores con independencia del proveedor. Sus labores principales son la (i) *intermediación*, encargada de buscar y encontrar los mejores servicios y validar que efectivamente se están cumpliendo los valores de calidad acordados; (ii) *agregación*, proporcionando la capacidad de integrar servicios de diferentes entornos CC; y, finalmente (iii) *arbitraje*, proporcionando un entorno similar a la agregación de servicios, pero en el que estos servicios no están predefinidos y cambian dinámicamente.
- *Cloud Carrier.* Que es el agente que proporciona conectividad entre el proveedor y el consumidor. Resulta importante, ya que muchos de los objetivos acordados en el SLA dependen de que el *Cloud Carrier* sea capaz de proveer con la tecnología para transportar la información con la suficiente rapidez.

No obstante, más allá de esta arquitectura de referencia también es posible estudiar las arquitecturas de las plataformas abiertas CC, cuya información es pública. Así pues, hoy en día existe una gran variedad de plataformas CC abiertas, distribuidas bajo la licencia *Open Source* y como no podía ser de otro modo también existen diferentes trabajos que comparan estas plataformas

[Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012] [Mahjoub *et al.*, 2011] [Voras *et al.*, 2011] [Sempolinski *et* Thain, 2010].

A continuación se presentarán aquellas plataformas CC, que a nuestro juicio son las más importantes y que además son las más utilizadas, es decir, OpenStack[21], OpenNebula[22] y, en menor medida, Eucalyptus[23]. No cabe duda que existen otras plataformas, a las que a priori se les asigna el apellido cloud, como por ejemplo OpenQRM [Halinka, 2008], Nimbus [Keahey, 2009], CloudStack [Sabharwal, 2013], Cloud Fondry [Cohen, 2013], Open Monbster [Dlamini, 2013], etc. Sin embargo después de analizarlas estas plataformas no están diseñadas para cubrir un espectro generalista, o bien su único interés es el de facilitar la gestión de la capa *hardware* física y virtual, tarea que la mayoría de los sistemas de virtualización modernos ya incorpora como tarea por defecto. En el anexo Anexo C se incluye una descripción con mayor detalle sobre el conjunto de plataformas existentes en el marco del paradigma CC.

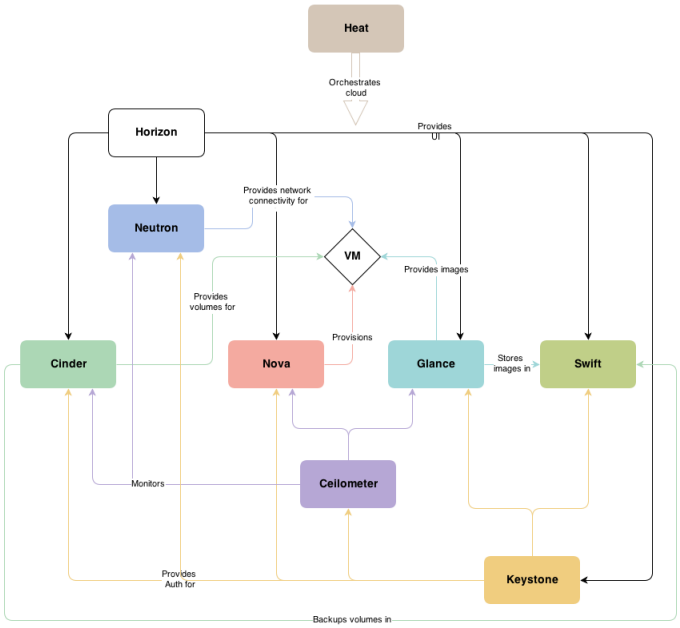

Figura 7.- Arquitectura conceptual de OpenStack[24]

En primer lugar, OpenStack apareció en escena en torno a Julio de 2010, y fue convirtiéndose en popular debido a que grandes empresas fueron

[21] http://www.openstack.org/
[22] http://opennebula.org/
[23] https://www.eucalyptus.com/
[24] http://docs.openstack.org/admin-guide-cloud/content//index.html

aceptando su modelo (Citrix[25], Dell[26], Suse[27], Telefónica[28], etc.), hoy en día agrupa a más de 13.500 personas en 132 países. Es un conjunto de proyectos de código abierto que pueden ser utilizados para configurar un entorno CC a medida.

En la Figura 7 se presentan los principales componentes de esta arquitectura [Sefraoui *et al.*, 2012]. Éstos son el servicio de identidad (*Keystone*), computación que incluye diferentes subservicios de almacenamiento, red, planificador y comunicaciones (*Nova*), almacenamiento (*Swift*), imágenes (*Glance*) y el servicio de configuración (*Dashboard*). Openstack es abierta y cada uno de estos servicios puede desplegarse en cualquier servidor, ya que están desacoplados entre sí, comunicándose a través de una cola de mensajes basada en el protocolo *Advanced Message Queuing Protocol* (AMQP), no obstante el componente NOVA que gestiona los recursos, debe estar disponible en todos servidores, o al menos en sus componentes básicos.

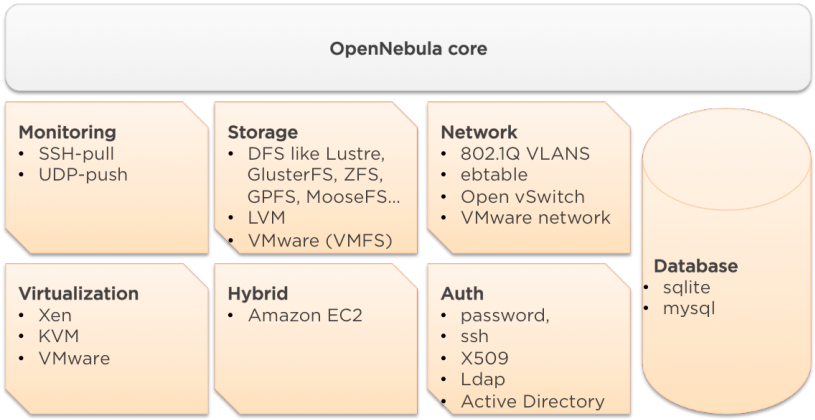

Figura 8.- Componentes OpenNebula[29]

Por su parte, OpenNebula [Milojičić *et al.*, 2011] fue la primera versión de un sistema CC abierto, su desarrollo empezó en 2005 y su primera versión estuvo disponible en 2008. Es un proyecto mucho más extendido que Openstack y es utilizado por una gran cantidad de instituciones y empresas para construir su propio entorno CC. OpenNebula está pensado para su uso en grandes centros de datos de forma que pueda transformar esos recursos en un

[25] https://www.citrix.es/
[26] http://www.dell.es/
[27] https://www.suse.com/
[28] http://www.telefonica.es/
[29] http://docs.opennebula.org/stable/design_and_installation/getting_started/index.html

entorno CC para ofrecer servicios de infraestructura. Su arquitectura (Figura 8) es flexible y modular, permitiendo integrar una gran cantidad de almacenamiento, red y entornos de virtualización.

Finalmente, Eucalyptus (*Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*) fue lanzada inicialmente en el año 2008. Al igual que las anteriores, su objetivo es la transformación de un centro de proceso de datos tradicional en otro basado en CC con la capacidad de ofrecer servicios de tipo infraestructura. Sus componentes principales son un el NC (*Node controller*), situado en cada máquina física, el SC (*Storage controller*) y el CLC (*Cloud Controller*) que es único en cada entorno de despliegue, adicionalmente dispone del módulo Walrus que actúa de puerta de entrada de las peticiones del entorno CC. Su principal característica es la compatibilidad con Amazon EC2 y S3, permitiendo incluso un modo de funcionamiento mixto, con gestión simultánea de máquinas virtuales en la instalación local y en EC2.

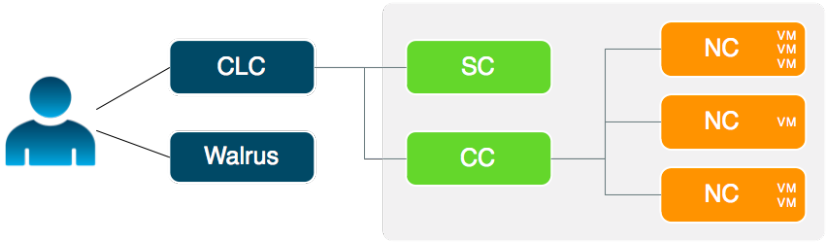

Figura 9.- Arquitectura de referencia en Eucalyptus[30]

Aunque Eucalyptus, OpenNebula y OpenStack (así como el resto de plataformas abiertas) comienzan a ser muy utilizadas, la realidad demuestra que su desarrollo todavía está lejos de las características deseables de una plataforma CC, ya que en su amplia mayoría se centran en la gestión de la infraestructura, sin tener en cuenta al usuario final. Una arquitectura moderna debería tener en cuenta la calidad de los servicios que ofrece (QoS) y los acuerdos (SLA) alcanzados con los usuarios. Sin embargo, estas arquitecturas abiertas están más centradas en la gestión de la compleja infraestructura subyacente, es decir, en las capas inferiores, que en los servicios que se le ofrecen a los usuarios finales.

Finalmente, en la Tabla 2 se presenta una relación de otras plataformas CC existentes en la actualidad.

[30] https://www.eucalyptus.com/docs/eucalyptus/3.1/ig/index.html

Prov.	Niv.	Empr.	Lic.	Plataf.	Descripción
vSphere	IaaS	VMWare	Comer.	N/A	Sistema operativo orientado a la formación de infraestructuras CC a través de virtualización.
Cloud Foundry	PaaS	VMWare	Libre	vSphere EC2	Capa PaaS para *vSphere*, que puede ser desplegada en infraestructuras de otros proveedores.
CloudStack	IaaS	Citrix	Libre	VMWare, Oracle VM, KVM, Xen	Capa de gestión de infraestructura que abstrae la plataforma tecnológica de virtualización subyacente.
AppScale	PaaS	Comun.	Libre	KVM, XEN, EC2, Eucalyptus	Ejecución en CC privado de aplicaciones de *Google App Engine*.
Nimbus	IaaS	Comun.	Libre	KVM, Xen	Gestión de infraestructura.
OpenQRM	IaaS	Comun.	Libre	KVM, Xen, VMWare	Gestión de infraestructura de centros de proceso de datos.

Tabla 2.- Plataformas Cloud Computing

2.5 RIESGOS Y VULNERABILIDADES

Tal y como se ha presentado, el paradigma computacional CC ha crecido con fuerza en los últimos años y su desarrollo ha motivado el avance de una gran cantidad de plataformas, tanto públicas, como privadas. También es relevante el cambio que ha producido en el modelo de negocio tradicional, así como la externalización de la información empresarial a los grandes centros de datos.

Sin embargo, todavía existen una serie de retos que la tecnología tiene que superar para que el uso de este paradigma computacional se extienda a todos los niveles. Estos retos existentes son principalmente los riesgos relacionados con la seguridad, entre las que se destacan las siguientes vulnerabilidades [Subashini *et al.*, 2011]: problemas de acceso, riesgos de seguridad en la tecnología de virtualización, vulnerabilidades en la tecnología web (SQL injection, cross-site scripting, seguridad física, IP spoofing, etc.) Así mismo, también existen dificultades debido a la pérdida del control de la información que se produce al alojar las bases de datos y el *software* de forma externa en grandes centros de datos, que son una caja negra para la empresa que hace uso de servicios. Entre estas vulnerabilidades destaca [Subashini *et al.*, 2011] [Hamdi, 2012]: la privacidad y confidencialidad de los datos, la pérdida y robo de información, problemas de autenticación, cumplimiento con las leyes de protección de datos y alta disponibilidad.

Diferentes autores han tratado de clasificar y acotar los problemas abiertos [Subashini *et al.*, 2011] [Grobauer *et al.*, 2011] [Bhadauria *et* Sanyal, 2012]. Estos estudios han propuestos diferentes taxonomías para organizar las vulnerabilidades, principalmente, atendiendo respectivamente (i) al tipo final de servicios en el que se produce (SaaS, PaaS o IaaS); (ii) la tecnología a la que se asocia el problema de seguridad; y (iii) finalmente, una clasificación atendiendo al tipo de despliegue (Público, Privado y/o de Comunidad).

Para analizar los problemas de seguridad, siguiendo el trabajo de [Grobauer *et al.*, 2011], en primer lugar hay que destacar que CC está basado en la integración de diferentes tecnologías subyacentes que trabajan de forma coordinada. Así, resulta necesario distinguir entre los problemas de seguridad intrínsecos a cada una de esas tecnologías y los problemas derivados de su uso de forma conjunta en un entorno CC. En este sentido es posible referirse a las vulnerabilidades de las tecnologías asociadas a un entorno CC:

- *Aplicaciones web y servicios.* Donde se enmarcan las vulnerabilidades relativas a las capas PaaS y SaaS de la pila de servicios del paradigma. Destacan vulnerabilidades relacionadas con la gestión de los datos [Subashini *et al.*, 2011] (localidad, integridad, segregación, control de acceso, confidencialidad, disponibilidad, violación de la información y la

gestión de copia de seguridad), así como las vulnerabilidades clásicas de la tecnología Web [Subashini *et al.*, 2011] (*Cross-site scripting*, inyecciones OS, SQL o LDAP, manipulación de cookies, problemas de configuración, vulnerabilidades en la capa TCP, ataques Captcha, etc.). No obstante, la mayoría de estas vulnerabilidades no están relacionadas con la capa de aplicación, sino que en torno al 60% están relacionadas con los niveles inferiores de la pila de tecnologías (sistema operativo y vulnerabilidades conocidas y no conocidas) [Baker *et al.*, 2011] [Bhadauria *et* Sanyal, 2012].

- *Protocolos de Comunicaciones.* Dado que la información es accedida de forma ubicua a través de diferentes redes, habitualmente Internet, utilizando para ello protocolos estándares de redes. Todas las vulnerabilidades asociadas a estos protocolos también constituyen riesgos de seguridad para el entorno CC. Entre estos riesgos destacan [Subashini *et al.*, 2011] [Bhadauria *et* Sanyal, 2012] el análisis de paquetes y la violación de redes, la gestión débil de sesiones, las peticiones SSL inseguras, ataques DNS, sniffers, reutilización de direcciones IP y Prefix hijacking.

  Todas estas vulnerabilidades pueden ser utilizadas para realizar ataques de tipo *Man-In-The-Middle* lo que puede llegar a comprometer la seguridad de los datos y de la infraestructura que subyace.
- *Persistencia de la información.* Como no podía ser de otro modo, también existen problemas relacionados con las bases de datos utilizadas en el entornos CC, que habitualmente son bases de datos orientadas a documentos [Han *et al.*, 2011]. Los problemas habituales están realizados por el uso de protocolo de encriptación de la información obsoletos [Grobauer *et al.*, 2011], los cuáles están ampliamente estudiados y las soluciones propuestas ofrecen un nivel de seguridad aceptable [Kamara *et al.*, 2010].

  También existen problemas de confidencialidad, en cuanto a la verificación de la identidad [Subashini *et al.*, 2011] y, fundamentalmente, debido a que los datos de diferentes empresas (*Cloud users*) son almacenados en el mismo centro de proceso de datos, lo que puede provocar que puedan tener acceso a información confidencial de otros usuarios. En este sentido, el rol *Cloud provider* debe asegurar la confidencialidad de los datos entre sus clientes, pero además implementar medidas de seguridad adicionales para impedir el acceso de sus propios empleados a datos confidenciales de terceros.

  Finalmente, se observan problemas relativos a la necesidad de disponibilidad de Internet para acceder a la información o los problemas de latencia derivados del uso de redes como medio de transmisión.
- *Gestión de la infraestructura.* El uso de una capa de virtualización tiene innumerables ventajas a la hora de gestionar de forma dinámica la

infraestructura. Sin embargo, también tiene más problemas de seguridad debido a que los usuarios potencialmente malintencionados comparten un mismo medio, e incluso un mismo servidor físico. Aunque la virtualización en sí misma puede ser considerada como una técnica de seguridad debido al aislamiento que proporciona. Los principales riesgos se enmarcan en la posibilidad de romper este aislamiento y conceder acceso a la máquina física, lo que por ende, da acceso al núcleo del entorno CC [Subashini *et al.*, 2011]. En este sentido, existen amplios estudio sobre las posibles vulnerabilidades [Reuben, 2007] y las soluciones a las mismas [Pal *et al.*, 2011].

Finalmente, en cuanto a los riesgos en la capa de virtualización, teniendo en cuenta que un entorno CC ofrece servidores virtualizados en el nivel IaaS, los proveedores de servicios tienen un nuevo reto, ya que deben implementar un modelo de seguridad compartida, donde el *Cloud Provider* debe asegurar la seguridad del entorno CC a nivel físico y el *Cloud User* debe ocuparse de la seguridad de la máquina virtual que adquiere como servicio [Grobauer *et al.*, 2011]. El reto reside en que el *Cloud provider* debe asumir problemas de seguridad dentro de su propia infraestructura debido a una gestión incorrecta de la configuración por parte del *Cloud user*.

En cuanto a los problemas de seguridad inherentes propios al paradigma, cabe destacar, en primer lugar las (i) *Vulnerabilidades en la monitorización*, ya que un entorno CC es aquel que tiene que autogestionarse, constituyendo un paradigma a medio camino entre *Utility Computing* [Ross *et* Westerman, 2004] y *Autonomic Computing* [Horn, 2001], es indudable que debe recoger datos sobre el uso a lo largo de toda su infraestructura (*hardware* y *software*). Con esta información, se evalúa el uso que cada usuario realiza del entorno CC y por lo tanto cuál es el coste asociado. Un problema de seguridad en esta monitorización, puede provocar la pérdida de los rendimientos derivados del alquiler de infraestructura o servicios a terceros [Grobauer *et al.*, 2011]. En segundo lugar, la (ii) *dificultad para la gestión de la identidad*, que debe seguir un modelo de tipo *Single sing-on* [De Clercq, 2002] y que es uno de los retos de este tipo de plataformas, no sólo por el hecho de asegurar la autenticación de usuarios en diferentes aplicaciones desplegadas en entornos geográficamente distribuidos, sino por el hecho de un almacenamiento compartido de las credenciales de seguridad de cada usuario. Las soluciones a estos problemas de seguridad son [Subashini *et al.*, 2011] sistemas de gestión de la identidad independientes entre los diferentes *Cloud users*, sincronización de credenciales de seguridad a nivel SaaS, y la federación del modelo de gestión de la identidad. Y, finalmente, la (iii) *Interoperabilidad entre plataformas*, que como ya se ha comentado es un campo de investigación en la actualidad, pero que además dificulta en gran medida su implantación a gran escala. Para una

empresa deslocalizar su información en el centro de procesos de datos de un tercero es un reto en sí mismo, pero además existe el problema de *lock-in*, que consiste en que una empresa no está dispuesta a depender únicamente de un solo proveedor debido a la dificultad de conexión entre plataformas.

Tal y como se ha presentado, los entornos CC como cualquier sistema de información y especialmente aquellos que son distribuidos, tiene un conjunto de riesgos y vulnerabilidades asociadas no sólo a las tecnologías subyacentes, sino al propio paradigma en sí mismo. Existen investigaciones activas y estudios sobre como asegurar la seguridad e interoperabilidad de las diferentes plataformas CC. La organización *Cloud Security Alliance* (CSA)[31] publica una guía sobre buenas prácticas en materia de seguridad en el marco de los entornos Cloud [CSA, 2011]. Aunque actualmente existen trabajos en todas las áreas vulnerables anteriormente señaladas, cuyo estudio se escapa de este trabajo, destacar que la gestión de seguridad depende en gran medida del modelo de despliegue del entorno CC (público o privado) [Bhadauria *et* Sanyal, 2012] ya que el tipo de usuarios y servicios que implementa son totalmente opuestos.

[31] https://cloudsecurityalliance.org/

2.6 CONCLUSIONES

A lo largo de este segundo capítulo se ha revisado de forma detallada todo el contexto tecnológico y social que envuelve al paradigma CC, así como su caracterización interna y marco de aplicación. No cabe duda que este modelo tecnológico ha experimentado una notable evolución desde sus inicios en el segundo lustro de la década pasada, donde ni si quiera IBM[16], que hoy en día es uno de los grandes proveedores CC, tenía una idea clara de cuál iba a ser su evolución.

Según Gartner [32] la gran aceptación por parte del tejido empresarial [Mitchell *et al.*, 2012], así como su fácil y rápida integración con las arquitecturas tecnológicas tradicionales [Desisto, 2013] ha motivado el rápido desarrollo hasta nuestros días. Así mismo, el modelo de comercialzación (*pay-as-you-go*) [Armbrust *et al.*, 2010], similar al de los productos de utilidad tradicionales, también ha sido otras de las claves de su rápida evolución. Ya que, las tecnologías subyacentes del paradigma CC han permitido producir los recursos computacionales que se ofertan como servicio atendiendo a las necesidades de los usuarios y permitiéndoles utilizar (y pagar) sólo la cantidad y calidad de recursos que realmente necesitan en cada momento.

A lo largo de este capítulo se ha analizado detalladamente como las tecnologías relacionadas y predecendentes han hecho posible la explosión de este modelo computacional. En este sentido, también se ha revisado como ha sido la evolución de las definiciones propuestas por los diferentes grupos y centros de investigación que centran su trabajo en el desarrollo del paradigma CC. En este sentido, se ha destacado sobre el resto la definición propuesta por el NIST [Mell *et* Grance, 2011] en la que, no sólo se define el propio paradigma y lo que representa, sino que también se describen las características que debe cumplir (principalmente, la disponibilidad de servicios a medida de forma automática y la provisión de recursos computacionales mediante la elasticidad). Así mismo, la definición propone los tres tipos de servicios ofertados ampliamente conocidos, pero que denomina capacidades (*software*, plataforma e infrastructura) y cuatro modelos de despliegue (público, privado, híbrido y comunitario). Estos modelos de despliegue, como también se ha detallado a lo largo del capítulo, dan lugar a un amplio debate sobre la interoperabilidad entre plataformas, el denominado InterCloud [Buyya *et al.*, 2010a] [Sim, 2013].

Sin embargo, pese a la calidad y amplitud de esta definición, no son pocas las voces [McKendrick, 2012] que indican que la definición no es lo

[32] http://www.gartner.com/technology/home.jsp

suficientemente precisa y que, en cierta medida, está influenciada por el interés de los gobiernos, principalmente estadounidense, por la implantación de este paradigma computacional en las diferentes admistraciones con el objetivo de incrementar la eficiencia y reducir de costes operacionales. En este sentido, tampoco son pocas las nuevas innovaciones tecnológicas que han surgido al amparo de este paradigma [Low *et al.*, 2011] [Azodolmolky *et al.*, 2013]. Sin embargo, después de realizar un estudio acerca de las plataformas existentes a lo largo del capítulo; se observa una pobre implantación de estas novedades. Así pues, la mayoría de estas plataformas simplemente basan su esfuerzos en proporcionar servicios de infraestructura *hardware* mediante el uso de la tecnología de virtualización subyacente, sin tener en cuenta las capacidades de los niveles superiores como plataforma y *software*.

El capítulo, para terminar, también revisa las vulnerabilidades del paradigma CC, determinando que la mayoría de las debilidades están asociadas a las tecnologías sobre las que se sustenta. Sin embargo, si que ha sido posible encontrar debilidades directamente asociadas al paradigma CC, las cuales puen ser agraupadas en las tres siguientes categorias: (i) servicios y aplicaciones, (ii) persistencia de la información y, finalmente, (iii) las debilidades relacionadas con los protocolos de comunicaciones.

Partiendo de estas debilidades y vulnerabilidades descubiertas, así como de las limitaciones en las plataformas CC y siguiendo la hipótesis propuesta en el presente trabajo de investigación, se seleccionan los SMA y sobre todo aquellos basados en OV para combatir los obstáculos hallados. Este modelo arquitectónico permite la auto-adaptación dinámica mediante la integración de innovaciones basadas en métodos, técnicas, herramientas y modelos derivados de la IA y de la distribución de responsabilidades entre los diferentes componentes del entorno CC. Por tanto, en el siguiente capítulo se analizará en detalle la relación existente actualmente entre SMA y el paradigma CC con el objetivo de determinar un ámbito claro de aplicación del modelo arquitetónico de los SMA basados en sociedades artificiales en el marco tecnológico propuesto por CC.

CAPÍTULO 3
SISTEMAS MULTIAGENTE Y CLOUD COMPUTING

Una vez que se ha acotado y caracterizado correctamente el término CC, a lo largo de este apartado se presentará una revisión del estado del arte de los SMA, prestando especial atención a la relación existente con el paradigma CC, de forma que sea posible la identificación de cómo los SMA pueden contribuir a la mejora del paradigma CC.

En primer lugar, los agentes y SMA han sido, desde su nacimiento, un campo de gran interés por parte de la comunidad científica. Este interés ha sido motivado, sin duda alguna, por el conjunto de capacidades diferenciales que presentan este tipo de sistemas con respecto a los sistemas distribuidos tradicionales, entre las que cabe destacar la autonomía, la capacidad de interacción, la racionalidad, así como sus características inteligentes y de aprendizaje [Wooldridge *et* Jennings, 1995]. La creciente complejidad de las aplicaciones *software* y los sistemas de información está motivada por el incesante avance de la infraestructura *hardware* y de los sistemas de comunicación. Éste es el caso de los entornos CC, los cuáles aúnan la gestión de grandes cantidades información, junto con interfaces web de última generación, todo ello, gestionado a través de un entorno masivamente distribuido. No cabe duda que el diseño de este tipo de entornos complejos requiere de nuevas técnicas de Ingeniería del *Software* [Boehm, 1984] [Pressman *et* Jawadekar, 1992] que sean capaces de afrontar las posibilidades que hoy en día ofrece la técnica, teniendo en cuenta la perspectiva de la IA. La propia teoría de agentes, ha tenido que evolucionar en gran medida desde sus inicios, allá en la década de los 90, para facilitar la creciente complejidad *software*, desde el concepto inicial de entidad con comportamiento autónomo, capaz de actuar en un medio para satisfacer unos requisitos individuales. Hoy en día, estos sistemas no pueden ser entendidos sin su capacidad para trabajar de forma coordinada, comunicándose y cooperando entre sí, formando SMA. La estructura de estos sistemas también se ha desarrollado utilizando la base que proporcionan las organizaciones en las sociedades humanas, en el que se tiene un fin común y están sujetos a un conjunto preestablecido de normas, acuerdos y protocolos de comunicación, sin perder, el dinamismo y la pro-actividad así como la capacidad de adaptarse al medio de forma autónoma.

Aunque en un primer momento, se puede pensar que ambos sistemas distribuidos (SMA y CC) son incompatibles, un análisis más detallado demuestra que son complementarios, siendo posible identificar un buen

número de sinergias entre ellos. Mientras que por un lado, los entornos CC puede cubrir las necesidades computacionales de persistencia de información y el potencial de cómputo que requieren los SMA en diferentes aplicaciones como la minería de datos, gestión de servicios complejos, etc. Por otro lado, los SMA pueden ser utilizados para el diseño de entornos CC mucho más eficientes, escalables y adaptables que los existentes actualmente. Además, el uso de SMA en el marco del diseño de sistemas CC permite sumarle a este paradigma nuevas características, como el aprendizaje o la inteligencia, lo que hace posible desarrollar entornos de computación mucho más avanzados en todas sus facetas (servicios inteligentes, interoperabilidad entre plataformas, distribución de recursos más eficiente, etc.). La relación simbiótica entre ambos paradigmas no es muy amplia y como se verá a lo largo del capítulo, por la naturaleza de los trabajos existentes, puede caracterizarse incluso como incipiente.

Este capítulo 3 de la memoria se estructura tal y como sigue, el apartado 3.1 contiene una breve revisión histórica sobre la teoría de agentes y los SMA. Posteriormente, el apartado 3.2 se realiza un análisis exhaustivo de los trabajos que aúan el paradigma CC con los SMA. A continuación, dado que se propone el uso de SMA basados en organizaciones virtuales para la formalización de la arquitectura, en el apartado 3.3 se realiza una revisión de este modelo de construcción de SMA, así como de las metodologías que hacen posible el análisis y diseño de sistemas complejos con el que nos incumbe siguiente la aproximación propuesta por los SMA. Finalmente, el apartado 3.4 contiene las conclusiones del capítulo.

3.1 TEORÍA DE AGENTES Y LOS SISTEMAS MULTIAGENTE

La teoría, y por ende, el paradigma computacional que conforman los agentes y los SMA, están basados en el concepto de agente, inicialmente propuesto por Russel *et* Norving [Russel *et* Norving, 1995]. Esta teoría de agentes no tiene que ser visto únicamente desde la perspectiva técnica (Ingeniería del *Software*, Sistemas Distribuidos, IA, etc.) sino que, como se verá más adelante a lo largo de esta sección, es necesario ir más allá, incluyendo otras ciencias como pueden ser la Psicología y la Sociología [D'Inverno *et* Luck, 2004] [Kolp *et al.*, 2006] [Davidsson, 2001].

En primer lugar, cuando se estudia la teoría de agentes, es necesario enunciar una definición del concepto de agente. Esta definición, aunque en el pasado fue un foco de discusión en la comunidad científica [Russel *et* Norving, 1995] [Wooldridge *et* Jennings, 1995] [Labidi *et* Lejouad, 1993]; hoy en día, el concepto de agente está suficientemente acotado. Entre todas las definiciones que han surgido, cabe destacar por su simplicidad, precisión y claridad la definición que enunció Wooldridge [Wooldridge, 2002]:

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

Posteriormente se refina esta definición, en colaboración con Jennings [Wooldridge *et* Jennings, 1995] en la que atribuía al agente las capacidades de *autonomía*, *habilidades sociales*, *reactividad* y *pro-actividad*. El paradigma de agentes surgió para satisfacer las deficiencias que la Ingeniería del *Software* clásica tenía para modelar sistemas *software* complejos. En este sentido, según [Dignum, 2004] [Jennings *et* Wooldridge, 1995], el paradigma de agentes puede ser aplicado en tres tipos de sistemas principalmente:

- **Sistemas abiertos.** Estos sistemas son capaces de cambiar dinámicamente, debido a que sus componentes no se conocen a priori y son altamente heterogéneos (diferentes entidades, implementaciones y técnicas, etc.).
  Para dar solución a este tipo de sistemas son necesarias técnicas de negociación y cooperación, cuya base forma parte de los sistemas multiagente [Bond *et* Gasser, 1988].
- **Sistemas complejos.** Se relacionan con sistemas grandes e impredecibles, cuya forma de abordarlos son el uso de técnicas de abstracción y modularidad. Ambas características forman parte intrínseca de los SMA, ya que la noción de agente autónomo es en sí mismo una abstracción de un módulo que encapsula procedimientos y

datos (objetos) que permiten resolver de forma autónoma un problema [Jennings, 2001].

- **Sistemas ubicuos.** Este tipo de sistemas mejoran el uso de un sistema informático mediante la utilización de la potencia computacional de un entorno físico, normalmente distribuido, pero que de algún modo se abstrae su complejidad de cara al usuario final. El sistema debe cooperar con el usuario para alcanzar los objetivos perseguidos. Existen diferentes ejemplos dentro de este tipo de sistemas, destacando el sector del control de procesos [Jennings *et al.*, 1995] y la manufacturación [Wooldridge *et al.*, 1996]

El otro gran foco de interés ha sido el de clasificar a los agentes atendiendo a diferentes criterios [Russell *et* Norving, 1995] [Nwana, 1996] [Brenner *et al.*, 2012] [Franklin y Graesser, 1997] [Wooldridge, 2002] [Cetkovic et Parmee, 2002]. Así pues, hoy en día existen diferentes clasificaciones de agentes, destacando principalmente las siguientes taxonomías:

- **Clasificación atendiendo al tipo de implementación del agente** [Russell *et* Norving, 1995]. Según esta clasificación se distinguen los siguientes cuatro tipos de agentes que se presentan ordenados en función de su complejidad: agente reflejo simple, agente reflejo con estado interno, agente basado en metas y agentes basados en utilidad.
- **Clasificación atendiendo al tipo de atributos de los agentes** [Nwana, 1996]. Nwana realiza una compleja clasificación de los agente, distinguiendo a los agentes en función de diferentes características. Así, en función de la (i) movilidad existen agentes estáticos y móviles; en función del (ii) modelo de razonamiento interno existentes agentes deliberativos y reactivos; y finalmente, atendiendo a (iii) otros atributos como la autonomía, la cooperación y el aprendizaje, se distinguen agentes colaborativos, de interfaz, de aprendizaje colaborativo y agentes Smart.
  Finalmente, la unión de características de estos tipos de agentes, da lugar, a un nuevo tipo de agente, denominado híbrido.
- **Clasificación atendiendo al diseño conceptual** [Cetkovic *et* Parmee, 2002]. Teniendo en cuenta el diseño conceptual de los agentes, existen agentes interfaz, búsqueda e información.
- **Clasificación atendiendo a la arquitectura** [Wooldridge, 2002]. La última clasificación que se presentará en este documento atiende a la arquitectura interna del agente. Siguiendo este criterio se distinguen arquitecturas reactivas, deliberativas e híbridas.

Dentro de estas clasificaciones de agentes destaca la que clasifica atendiendo a la arquitectura [Wooldridge, 2002], ya que la arquitectura de una agente determina cuáles son sus componentes principales, y como interactúan entre sí

para alcanzar la misión final. Si se considera un agente como un sistema complejo, la arquitectura deberá describir la estructura interna del agente, explicando como se descompone el agente en módulos independientes que interactúan para conseguir la funcionalidad requerida. En este sentido [Wooldridge, 2002] propone tres arquitecturas clásicas. La (i) arquitectura *reactiva* en el que el agente carece tanto de razonamiento como de capacidad para representar su entorno, y sus acciones se modelan mediante reglas básicas [Maes, 1990]. La (ii) arquitectura *deliberativa* donde el agente es capaz de mantener una representación simbólica del conocimiento y planificar el conjunto de acciones a tomar para conseguir sus metas. Y, finalmente, (iii) la arquitectura *híbrida* que es un modelo de arquitectura que aúna características de las dos anteriores.

La arquitectura deliberativa más ampliamente estudiada y extendida es el modelo BDI [Rao *et* Georgeff, 1992], que ha sido el más difundido dentro de los modelos de razonamiento, ya que combina un modelo filosófico asociado al razonamiento humano y un número considerable de implementaciones [Georgeff *et* Lansky, 1987]. Para el desarrollo del modelo se plantean un conjunto actividades mentales de los agentes inteligentes [Bratman et al., 1988] [Bratman, 1987]:

- **Creencias**. Relacionado con el conjunto de proposiciones que el agente acepta como verdaderas. Es decir, la visión que el agente tiene del entorno y el estado del resto de agentes.
- **Deseos o metas u objetivos**. Que hacen referencia al conjunto de propiedades que el agente trata de hacer verdaderas.
- ***Intenciones***. Asociadas al conjunto de acciones planificadas que le permiten llegar a un estado deseado.

Estos agentes proporcionan soluciones en entornos dinámicos e inciertos. Además, son capaces de enfrentarse a problemas del mundo real, incluso cuando sólo cuentan con una visión parcial del problema y con un número limitado de recursos.

3.1.1 LOS SISTEMAS MULTIAGENTE

Los agentes son entendidos por definición como entidades que interaccionan con el medio y también con otros agentes. En este sentido, cuando dos o más agentes son capaces de trabajar de forma conjunta con el objetivo de resolver un problema se habla de SMA [Mas, 2005]. Además, la mayoría de sistemas *software* existentes hoy en día son altamente complejos (como los entornos CC), ya que suelen ser sistemas concurrentes, que interaccionan entre sí y con otros sistemas externos [Zambonelli *et al.*, 2003]. Como consecuencia, los sistemas *software* tienen que ser abiertos, es decir,

existe la posibilidad de que nuevos componentes se incorporen y deban trabajar con el resto de componentes ya existentes. Éste es el entorno ideal de trabajo de los SMA.

Los SMA extienden la idea de agente individual mediante la descripción de la infraestructura, que proporciona la comunicación e interacción, adaptándose a la complejidad de un sistema abierto y a los retos existentes en el entorno gracias a sus características particulares [Kolp et al., 2006]:

- **Previsibilidad.** Los agentes individuales tienen diferentes niveles de libertad, de forma que pueden tomar acciones dentro de sus respectivos dominios. Los SMA tienen que ser lo suficientemente predecibles como para ser capaces de anticipar las situaciones futuras y elaborar planes o secuencias de tareas que les permita asegurar los objetivos preestablecidos. Entre las técnicas asociadas a la planificación cabe destacar la (i) teoría de grafos [Rahmani et al., 2009], la (ii) planificación basada en la satisfactibilidad [Kautz et al., 2006], el (iii) razonamiento basado en modelos [Jensen et Veloso, 2000] [Cimatti y Roveri, 2000], la (iv) utilización de heurísticas [Szer et al., 2005], la (vi) planificación con tiempo y recursos [Shen et al., 2006] y, finalmente, la (vii) planificación basada en casos [De Paz et al., 2009].
- **Seguridad.** Es una medida del grado en el que el sistema puede protegerse de usos indebidos, así como de la capacidad para asegurar la integridad de los datos y los recursos. [Tesauro et al., 2004].
- **Adaptabilidad.** Un SMA debe de ser capaz de adaptarse a los cambios que se produzcan en el entorno. Esta adaptación depende de la capacidad de los agentes individuales para aprender y predecir los cambios [Bousquet et al., 2002], e incluso de su capacidad para evaluar el estado del propio entorno [Horling et al., 1999].
- **Comunicación.** La coordinación es una de las características de los sistemas multiagente [Omicini et al., 2001]. Es la capacidad de distribuir la experiencia, recursos e información entre las diferentes entidades del sistema [Ramchurn et al., 2004].
  La comunicación puede ser vista desde dos puntos de vista. En primer lugar, (i) la c*oordinación y negociación* donde los agentes son capaces de armonizar sus acciones con otros agentes para satisfacer objetivos comunes; y (ii) la *competitividad* en los que existe una disputa entre los diferentes miembros del sistema. En este tipo de sistemas, el éxito de un subconjunto de agentes, implica el fallo de otro subconjunto.
- **Disponibilidad.** Aunque la disponibilidad puede ser considerada un subobjetivo de seguridad, la necesidad de cooperación en un SMA requiere que sea necesario considerar la disponibilidad en el diseño de los SMA, ya sea desde un punto de vista implícito o explícito.

- **Tolerancia a fallos.** En un SMA, el fallo de un agente no tiene que provocar el fallo de todo el sistema. En este sentido, un fallo de un individuo debe provocar la coordinación del sistema para determinar quien debe asumir las tareas del agente erróneo.
- **Modularidad.** La modularidad de un SMA incrementa la eficiencia en la definición de las tareas, reduce la comunicación y, finalmente, incrementa la flexibilidad del sistema.
- **Agregación.** Es una medida del grado en que los agentes pueden convertirse en partes de otros agentes, o formar parte de un sistema de nivel superior. La composición de agentes, implica la pérdida del control por cada uno de los componentes para, en contrapartida, tener un control global de los agentes agregados, lo que proporciona mayor flexibilidad [Sabater, 2002].

En definitiva, los sistemas multiagente proporcionan herramientas para modelar sistemas complejos y altamente distribuidos [Bauer et al., 2001]. Además, los SMA definen formalmente las tareas de comunicación, coordinación y negociación, lo que habilita el intercambio de conocimiento y, por lo tanto, coordinar el desarrollo de las actividades, pudiendo ser posible la negociación en caso de conflictos [Mas, 2005].

3.2 LOS SISTEMAS MULTIAGENTE EN EL MARCO DEL PARADIGMA CLOUD COMPUTING

El número de trabajos que se pueden encontrar en el estado del arte que relacionan CC y la tecnología de agentes es muy reducido, aunque sin embargo esta tendencia está diminuyendo y cada vez es más fácil encontrar estudios y aplicaciones centradas en este campo. Pese al limitado número de estudios sobre la materia, empieza a ser habitual el concepto **Agent-based cloud computing** o **Agent-based cloud platform**, enunciado por diferentes autores en los últimos años [Talia, 2011] [Talia, 2012] [Kang *et* Sim, 2011] [Sim, 2012] [Venticinque *et al.*, 2010]. Así pues, a lo largo de este apartado, se hará un estudio de los principales trabajos existentes en el estado del arte que relacionan ambos paradigmas computacionales.

El estado del arte no presenta un gran número de estudios acerca de la relación entre agentes y CC. Únicamente es posible encontrar la clasificación que propone Talia [Talia, 2011] [Talia, 2012], aunque con un límitado número de referencias. Esta taxonomía se realiza desde un punto de vista agentivo, diferenciando dos grandes grupos. Por un lado, (i) las aplicaciones de MAS que utilizan los servicios CC o están directamente desplegados sobre él (*Agents using cloud*). Y por otro lado, (ii) los entornos CC que utilizan la tecnología de agentes para la gestión de sus recursos (*Cloud using agents*).

Sin embargo, después de realizar un análisis detallado del estado del arte, no resulta fácil enmarcar los trabajos en uno u otro grupo. Por ello, en este trabajo, se propone una nueva clasificación desde el punto de vista CC, en base a la arquitectura de referencia propuesta por el NIST [Liu *et al.*, 2011] y las diferentes responsabilidades de cada uno de los roles participantes en el paradigma CC según la citada arquitectura: *Provider*, *Consumer*, *Broker*, *Carrier* y *Auditor*. Si comparamos ambas clasificaciones, el grupo de *Agents using cloud*, estaría totalmente alineado con rol *Consumer*, en la clasificación que se propone; pero el grupo *Cloud Using Agents* tendría tres tipos de roles implicados (*Provider*, *Broker* y *Auditor*) por lo que esta propuesta aporta una mayor capacidad semántica. Dentro de esta clasificación, no se incluye al rol *Cloud Carrier*, ya que es la entidad encargada de proporcionar transporte a la información y no aportaría ninguna funcionalidad posible entre SMA y CC. Los siguientes apartados presentarán los trabajos existentes en el estado del arte según esta clasificación propuesta.

3.2.1 CLOUD CONSUMER

El rol *Cloud consumer* es aquel que hace uso de las capacidades que proporciona un proveedor de servicios computacionales de tipo CC. Existen números trabajos en el que los sistemas SMA utilizan los servicios

computacionales proporcionados por los sistemas CC. A continuación se presentan los trabajos en los que los SMA utilizan servicios proporcionados por los entornos CC; organizados en función de los servicios más habituales:

- **Servicios de persistencia y computación.** La persistencia de la información es uno de los grandes servicios en el marco de los entornos CC, permitiendo el almacenamiento de grandes cantidades de datos; o liberando a dispositivos móviles ligeros de la necesidad de almacenar información. Existen autores que incluso llaman a estos servicios *Data as a Service* (DaaS) [Wang *et al.*, 2008]. En el estado del arte se pueden encontrar diferentes aplicaciones en ámbitos muy diversos, como la gestión del transporte [Li *et al.*, 2011a], la persistencia de objetos de aprendizaje [De la Prieta *et al.*, 2014], almacenamiento de vulnerabilidades de seguridad [Oberheide *et al.*, 2008], etc.
  El almacenamiento de grandes cantidades de información en entornos CC suele denominarse *Big Data* [Manyika *et al.*, 2011] y su análisis se denomina *Big Data Analysis* [Russom *et al.*, 2011], para lo cuál además de utilizarse las capacidades de persistencia de las plataformas CC, también se requiere su potencia computacional. En este campo puede encontrarse una variedad amplia de trabajos aplicados en diferentes áreas como el análisis de mercado [Amato *et al.*, 2013] [Cristelli *et al.*, 2014] [Surendran *et al.*, 2012], fusión de información [Blasch *et al.*, 2014] [Chamorro *et* De Solis Montes, 2013], ámbito socio-sanitario [Eggleston, 2013] [Taboada *et al.*, 2013], etc.
  Aunque estas aplicaciones están basadas en plataformas CC, muchas de ellas siguen utilizando el concepto del paradigma *Grid Computing* [Zhang *et al.*, 2010], como por ejemplo la herramienta CASE de simulación propuesta por Decraene *et al.* [Decraene *et al.*, 2010] [Decraene *et al.*, 2011], el uso del cluster Hadoop[33] de Apache [Sethia *et* Karlapalem, 2011], la simulación sobre población [Chen *et al.*, 2013], las redes de sensores [Haghighi, 2013], etc.
  No obstante, también existe otro tipo de aplicación de CC y agentes en las cuáles el entorno CC absorbe las necesidades computacionales que de otro modo no podrían realizarse por el SMA, como en dispositivos ligeros [Oberheide *et al.*, 2008] [Aversa *et al.*, 2010] [Yun *et al.*, 2012], o el análisis de grandes cantidades de información utilizando técnicas de minería de datos [Othame *et* Hebri, 2012] en diferentes campos como la bioinformática [Bajo *et al.*, 2010], la gestión de tráfico rodado [Wang *et* Shen, 2011], recolección y análisis de información [Krol *et al.*, 2012], etc.

[33] http://hadoop.apache.org/

Finalmente, otra gran aplicación es el uso de las plataformas CC para paralelizar secuencias de tareas, así pues, es posible encontrar herramientas para la ejecución de tareas utilizando algoritmos de planificación mediante agentes [Gutierrez-Garcia *et* Sim, 2012]; o el que, quizás, sea uno de los trabajos más prometedores en el campo ya que se trata de una pionera modificación sobre la metodología Moise+ [Hübner *et al.*, 2002], denominada ParaMoise [Guzek *et al.*, 2013a] [Guzek *et al.*, 2013b] y que trata de facilitar y mejorar la ejecución paralela de los diferentes agentes que forman una sociedad u organización de agentes inteligentes a través de las características computacionales de un entorno CC.

- **Servicios de infraestructura.** Además de los servicios anteriormente identificados de computación y persistencia, el otro de gran capacidad que proporciona un entorno CC es la infraestructura, es decir, la capacidad de ofertar máquinas virtuales de múltiples características bajo demanda. Estas capacidades pueden utilizarse por el SMA para su despliegue en entornos de supercomputación y especialmente han sido utilizados para la simulación basada en agentes en un entorno de supercomputación [Wittek *et* Rubio-Campillo, 2012] [Leitao *et al.*, 2013], o mediante el uso de una plataforma de simulación específica como Repast HPC [Collier *et* North, 2011] [Nouman *et al.*, 2013] [Minson *et* Theodoropoulos, 2008].

  Finalmente, también destacar que es creciente el interés de la comunidad científica en ofertar no sólo una infraestructura tradicional en el marco del paradigma CC, sino también otro tipo de infraestructura como redes inalámbricas de sensores en el que los sistemas multiagente cobran especial importancia [Tapia *et al.*, 2013] [Yuriyama *et* Kushida, 2010] [Hassan *et al.*, 2009] [Cuzzocrea *et al.*, 2013] [Baktashmotlagh et *al.*, 2011].

Tal y como se observa en los trabajos presentados, cuando el SMA juegue el rol de *Cloud user* es posible mejorar sustancialmente las capacides de los SMA ya que les permite disponer de mayores recursos para realizar las tareas. Por ende, el marco de aplicación de los SMA crece debido a que es posible realizar razonamientos más complejos, se puede modelar el entorno con un mayor nivel de detalle y, además, se reduce en el tiempo de procesamiento a la hora de computar los algoritmos de razonamiento.

3.2.2 CLOUD BROKER

El rol del *Cloud broker* es el intermediario entre proveedores y consumidores, constituyendo un rol fundamental en el paradigma CC desde un punto de vista de la comercialización de servicios. El objetivo del *Cloud broker*

es proporcionar al consumidor servicios computacionales, con independencia del proveedor, adecuados a sus necesidades computacionales en función de su presupuesto. Para ello, el *Cloud broker* está en contacto con varios proveedores de forma simultánea, manteniendo un archivo con las características de los servicios que ofrece cada uno y el coste asociado. Así el consumidor, en este caso, delega en el *Cloud Broker* la capacidad de contratación, lo que le permite obtener la mejor calidad/precio en los servicios que utilizan los consumidores. En definitiva, las tareas principales del *Cloud Broker* son la intermediación y gestión de acuerdos, la agregación de servicios y el arbitraje.

Un *bróker* tiene que tener como cualidades principales la negociación, extracción de información compleja, así como la gestión de información y servicios; características que están perfectamente alineadas con las características de los SMA. Así pues, en el estado del arte se encuentran un buen número de trabajos, que además tienen la suficiente madurez como para ser utilizados en la vida real. Lo que en parte se argumenta debido a la relación existente de los trabajos existentes con campos ampliamente estudiados en los últimos años (búsqueda, composición de servicios, negociación, etc.).

K.M Sim en unos de sus trabajos más conocidos [Sim, 2012a], hace una disertación acerca de la aplicación de agentes en el marco CC, enunciando conceptos como *agent-based computing*, *agent-based cloud computing*, *agent-based cloud search engine*, *negociation agents and agent-based cloud commerce* y, finalmente, *agent-based cloud service composition*; siguiendo sus trabajos en este contexto. Sin embargo, de forma más generalista en el presente trabajo únicamente propones tres subgrupos clave tal y como sigue:

- **Buscador de proveedor Cloud.** Este tipo de herramientas se basan en la centralización de la búsqueda de proveedores de servicios CC en función de las características de calidad/precio requeridas por los consumidores. El propio Sim propone el buscador *Cloudle* [Kang *et* Sim, 2011] [Sim, 2012a] que hace uso de un SMA que utiliza un *web crawler* [Heydon *et* Najork, 1999] para recoger la información sobre servicios y precios que disponen de los proveedores, para posteriormente relacionar estas características a través de una ontología que permita la interoperabilidad entre proveedores que usen terminología distinta para referirse al mismo servicio o característica. Dentro de esta línea, existen otros trabajos como el propuesto por Alhamad *et* al. [Alhamad et all, 2010a] [Alhamad *et al.*, 2010b] en el que proponen un modelo de confianza y reputación, basado en diferentes pesos sobre parámetros del SLA, de forma que pueda servir como referencia a los autores a la hora de elegir un proveedor de servicios. Para ello, en función de los parámetros requeridos por el consumidor y los proporcionados por el proveedor valida que, efectivamente, no se viola el acuerdo SLA. More

et al. proponen un modelo similar mucho menos avanzado [More *et al.*, 2014]. En esta línea, otro enfoque es el propuesto por *Habib et al.* [Habib *et al.*, 2011] en el que se presenta un modelo de confianza multifaceta [Dasgupta *et* Serageldin, 2000] más complejo. Finalmente, Mouratidis *et al.* proponen un modelo de selección de proveedores CC en base a criterios de seguridad [Mouratidis *et al.*, 2013].

- **Negociación de acuerdos SLA.** En este grupo enmarcamos a los SMA que permiten la gestión automática o semiautomática de los acuerdos SLA entre el consumidor y el proveedor, liberando al consumidor de esta tarea.

  Uno de los primeros modelos de negociación basado en agentes, fue propuesto por An *et al.* que utiliza una máquina de estados que tenía en cuenta las fases de negociación en el que únicamente incluía tiempo y costes [An *et al.*, 2010]. K.M. Sim también propone un modelo propio de negociación [Sim, 2010] [Sim, 2012a] [Sim, 2012b] bastante más complejo que el anterior, ya que se realiza entre varios consumidores y proveedores, se basa en una estrategia de negocio y tiene en cuenta aspectos como el tiempo, regateo, información incompleta y la estimación de costes. Otros trabajos que destacan en este sentido son el *Cloud Agency* [Aversa *et al.*, 2010] [Venticinque *et al.*, 2010] [V enticinque *et al.*, 2011], desarrollado en el marco del proyecto FP7 mOSAIC[34]. El modelo de este *framework* utiliza, al igual que Cloudle, una ontología para facilitar la interoperabilidad semántica entre plataformas, así como un SMA que realiza la negociación, monitorizando los servicios proporcionados y renegociando en el caso de que cambien las necesidades del consumidor, o los servicios del proveedor. Finalmente, también existen otros modelos de negociación específicamente diseñados para dispositivos ligeros [Lai *et al.*, 2012].
- **Composición de servicios.** Se refiere a la composición de servicios CC para da lugar a un único servicio más complejo proporcionado por uno o varios proveedores de forma simultánea. Diversos autores se refieren a la composición de servicios como *Cloud Manufacturing* [Li *et al.*, 2011b] [Tao *et al.*, 2011] [Xu *et al.*, 2012].

  La composición de servicios en CC puede verse desde dos vertientes [Gutierrez-Garcia *et* Sim, 2010]: (i) horizontal que se refiere a servicios heterogéneos de múltiples proveedores; y, (ii) vertical, que se refiere a la composición de servicios homogéneos orientados a incrementar la capacidad del servicio en un determinado momento.

[34] Open Source API and platform for multiple Clouds. http://www.mosaic-cloud.eu/

En cuanto a las aplicaciones existentes [Gutierrez-Garcia *et* Sim, 2010] [Sim, 2012a], propone un servicio automático de composición vertical basado en el FSCNP (*Focused Selection Contract Net Protocol*) para la selección automática de servicios y una tabla de compatibilidad de servicios, previamente establecida; de forma, que si un plataforma CC no es capaz de asumir el SLA acordado, se puede seleccionar otro servicio de las mismas características. Estos mismos autores, disponen de otro modelo [Gutierrez-Garcia *et* Sim, 2010] en el que desarrollan un SMA para la composición de servicios verticales en una ontología que proporcione interoperabilidad y un modelo de composición basado en redes de Petri, redes que han sido ampliamente utilizadas en este campo [Hamadi *et* Benatallah, 2003].

La combinación de estos tres tipos de componentes (además de la autenticación multicloud [Celesti *et al.*, 2010b]) habilita uno de los grandes objetivos perseguidos en el marco de la investigación de los sistemas CC, como es la federación [Buyya *et al.*, 2010a] [Celesti *et al.*, 2010a] [Celesti *et al.*, 2010b] [Sim, 2013]. Esto es, tal y como se ha estudiado (apartado 2.3.2), la posibilidad de que un mismo consumidor pueda trabajar con varios proveedores de forma simultánea para una misma tarea, pudiendo alterar su relación contractual con cada proveedor de servicio de forma dinámica. Entre los trabajos en el marco de los sistemas CC federados destacan, los propuestos por [Singh *et* Malhotra, 2012] que proponen un modelo basado en agentes inteligentes para la provisión de servicios de infraestructura. Chen *et al*. propone un modelo de interacción entre CCs basado en agentes móviles [Chen *et al.*, 2010], abstrayendo de la complejidad a los usuarios. Un enfoque mucho más avanzado que incluye experimentación en un entorno real es el presentado por [Venticinque *et al.*, 2012] y basado en el framework *Cloud Agency* [Aversa *et al.*, 2010] previamente presentado; en él, se utiliza un sistema multiagente para la provisión, gestión, ejecución y reconfiguración de servicios CC de tipo infraestructura proporcionados por terceros. Otros dos modelos similares, pero adaptados a cualquier tipo de servicio, no sólo de infraestructura, son los desarrollados de forma paralela por [Calheiros *et al.*, 2012] [Fan *et al.*, 2011] en el que los servicios migran entre entornos CC con el objetivo de satisfacer los SLA acordados con los consumidores. Otra perspectiva es la posibilidad de distribuir las tareas entre varios entornos de forma simultánea tal y como propone Palmieri *et al.* [Palmieri *et al.*, 2013], configurando un modelo altamente escalable. Finalmente, una aproximación diferente es de SERA [Ejarque *et al.*, 2010] en la que el modelo distribución de servicios en múltiples entornos CC se basa en la negociación con varios entornos CC utilizando un motor de inferencia ontológico.

Aunque el número de trabajos existentes en este campo es muy amplio, en todos ellos se observa un problema de fondo, y es la dependencia de entornos

CC. La búsqueda y composición de servicios no es posible si los entornos CC no proporcionan las herramientas que los hagan posible. Así mismo, en cuanto al aseguramente de la calidad de los servicios, si que es posible medir el grado de cumplimiento de la calidad acordada mediante SLA. Sin embargo, sin un control de la infrastructura subyacente no es posible asegurar el cumplimiento el nivel de calidad en los servicios.

3.2.3 CLOUD PROVIDER

El rol *Cloud provider* es aquel encargado de proporcionar servicios a terceros, y el más interesante desde el punto de vista del trabajo de tesis que se propone. Las tareas asignadas a este rol son fundamentalmente el control y organización de las plataformas CC a nivel interno, lo que que implican tanto el despliegue y orquestación de servicios, como la gestión de la infraestructura tecnológica subyacente. En este sentido, es necesario asegurar al mismo tiempo la privacidad y seguridad tanto del entorno, como de la información que se gestiona; así como que se satisfacen los acuerdos SLA acordados con los usuarios.

A continuación se presentan los trabajos existentes más importantes agrupados tal y como sigue:

- **Seguridad y privacidad.** Un entorno computacional de altas prestaciones, como es el caso de un entorno CC está sometido a una gran variedad de ataques [Hamdi, 2012]. En estos entornos, la seguridad puede verse a nivel interno, como medio para asegurar que toda la infraestructura subyacente funciona correctamente; y también a nivel externo, que hace énfasis en las comunicaciones con el exterior, validando que no comprometan al propio entorno.
  Como no podía ser de otro modo, los SMA cubren ambos puntos de vista. A nivel externo, los trabajos están principalmente relacionados con la autenticación de los usuarios en el entorno CC, como el proyecto ABAC [Venkataramana *et* Padmavathamma, 2012], el modelo de autenticación en el lado del cliente [Hajivali *et al.*, 2013], o el trabajo propuesto por [Habiba *et al.*, 2013] en el que un SMA se basa en políticas de acceso para conceder privilegios sobre la información. A nivel interno, por un lado se observan trabajos iniciales en diferentes aspectos como la privacidad de los datos [Damiani *et* Pagano, 2010], la monitorización de la plataforma [Pal *et al.*, 2011] y la seguridad de la infraestructura *hardware* (real o virtual) [Li *et al.*, 2012] [Meera *et* Swamynathanb, 2013]. Por otro lado, los trabajos que consideramos más maduros están relacionados con el almacenamiento seguro de los datos, destacando el proyecto de almacenamiento *CloudZone* [Talib *et al.*, 2012a] [Talib *et al.*, 2012b] [Talib *et al.*, 2012c] [Talib *et al.*, 2011a]

[Talib *et al.*, 2011b] [Talib *et al.*, 2010] que está basado en un SMA para asegurar la privacidad, acceso a la información y otros parámetros de seguridad en el almacenamiento. Unido a este proyecto, los mismos autores han diseñado la metodología *Prometheus* [Talib *et al.*, 2011c] que pretende garantizar la seguridad en el almacenamiento de datos en entornos CC. Existen otros trabajos, como el presentado por [Govinda *et* Sathiyamoorthy, 2012] en el que se trata de garantizar la seguridad de información mediante la ofuscación de los datos. Finalmente, el último trabajo encontrado es el modelo de seguridad multicapa para el acceso a la información persistente [Islam *et* Habiba, 2012a] [Islam *et* Habiba, 2012b] que también esta basado en agentes.

- **Oferta de Servicios.** Todo el marco tecnológico y comercial que envuelve al paradigma CC está orientado a ofrecer servicios computacionales, con independencia de su tipo o complejidad. En este sentido en el estado del arte se pueden encontrar diferentes tipos de servicios gestionados por un SMA aplicados a entornos de oficina [Chao *et* Sun, 2013], gestión eficiente de energía [Yang, 2012] [Yang *et al.*, 2011], seguridad [Oberheide *et al.*, 2008], e-learning [De la Prieta *et al.*, 2014] [Babu *et al.*, 2014] [Kalagiakos *et* Karampelas, 2011], etc. Incluso, existen perspectivas diferentes en la que se proponen ofrecer los propios agentes como servicio AaaS [Lopez-Rodriguez *et al.*, 2011].
  Sin embargo, cuando se habla de servicios en el marco del paradigma CC, es necesario cumplir las restricciones y objetivos acordados con los consumidores en el SLA, y por lo tanto los servicios tienen que monitorizarse para cumplir los niveles de calidad, en este sentido se ha acuñado el concepto de servicios sensibles a la calidad (*QoS-aware*) [Zeng *et al.*, 2004]. En este campo los SMA tienen una gran aplicación y por ello han surgido en los últimos años diferentes trabajos relacionados [Wei *et al.*, 2010b] [Wei *et al.*, 2012] [Li *et al.*, 2013], aunque todos ellos son trabajos iniciales, que abordan el problema de forma parcial y que tratan de satisfacer las demandas en la calidad del servicio. Más allá de la monitorización, se hace necesario disponer de la capacidad de gestión de la infraestructura subyacente que es la que aporta potencia computacional y permite satisfacer la calidad demanda para un servicio dado en un tiempo establecido.
- **Gestión de la infraestructura subyacente.** La gestión de la infraestructura subyacente es una de las tareas más complejas debido a la incertidumbre, dinamicidad y heterogeneidad de un entorno en el que existen diferentes máquinas físicas, sistemas de virtualización, topologías de red, modelos de almacenamiento, etc.; además de la complejidad en el *hardware* y *software* existente.
  Los SMA tienen un gran campo de aplicación en este contexto, sin embargo, no existen muchos ejemplos en el estado del arte. No

obstante, es posible encontrar trabajos preliminares en el que un SMA gestiona recursos en función de los acuerdos SLA de forma centralizada [Van *et al.*, 2009b], el trabajo propuesto por [Wei *et* Blake, 2013] en el que se asignan recursos a servicios sigue un modelo de flujos de trabajo, y finalmente, la arquitectura MASCloud que utiliza un SMA para optimizar costes en el marco del paradigma CC [Nuñez *et al.*, 2012].

A excepción de los relacionados con la seguridad de sistemas, el número de trabajos existentes en el que los SMA toman el rol de *Cloud Provider* son mucho menores que en el caso de roles anteriores, además, su grado de madurez es menor. Así es muy difícil encontrar SMA para el control de infrastructura subyacente, la orquestación de servicios y recursos computacionales y la coordinación del entorno, en general. Esto sin duda es debido a la novedad de las tecnologías subyacentes empleadas como es el caso de la virtualización *hardware*, la monitorización de sistemas de alta disponibilidad en clúster, etc.

3.2.4 CLOUD AUDITOR

Finalmente, el *Cloud auditor* es el último rol cuyas responsabilidades consisten en asegurar desde un punto de vista externo y objetivo donde los acuerdos SLA se cumplen satisfactoriamente. Este es el grupo donde existen menos trabajos en lo que se pueden ver las relaciones entre SMA y entornos CC, observándose únicamente dos referencias destacables, la arquitectura FOSII [Emeakaroha *et al.*, 2012] en el que una arquitectura multiagente detecta violaciones en los SLA de forma automática; y el trabajo de [Ramaswamy *et al.*, 2011] en el que un SMA, en el lado del cliente detecta posibles violaciones en los acuerdos SLA.

3.2.5 CONCLUSIÓN

Después de analizar la relación existente entre ambos sistemas computacionales, es posible concluir que hoy en día el uso común de SMA en entornos CC continúa siendo incipiente. Aunque por la variedad de trabajos presentados se puede afirmar que existe un gran interés en la comunidad científica en el desarrollo de técnicas y herramientas que aúnen las ventajas de ambos entornos tecnológicos.

Así, cuando los SMA toman el rol de *Cloud Consumer*, el entorno CC ofrece tecnología de altas prestaciones, con un alto rendimiento, disponibilidad escalabilidad [Talia, 2012]. Permiten y facilitan la aplicación de SMA en una gran variedad de aplicaciones complejas. El uso de las capacidades computacionales permite expandir el modelo de razonamiento y conocimiento de los SMA tradicionales, ya que reduce las restricciones temporales y ofrece diferentes modelos de almacenamiento de alto rendimiento. En la Tabla 3 se

presenta un resumen con los principales usos de los servicios CC en el marco de los SMA, incluyendo el tipo de servicios que se utiliza y si éstos servicios sirven para proporcionar otros servicios a terceros. Como se aprecia, los principales usos están realizados con los servicios de infrastructura, computación y persistencia. Existiendo un buen número de trabajos que utilizan los servicios CC para proveer servicios más complejos a los usuario finales.

Uso principales	**Tipo de servicio utilizado**			**Propor. servicios terceros**
	SaaS	**PaaS**	**IaaS**	
Persistencia y Computación				
Servicios BigData			X	
Análisis de información			X	
Servicios Grid Computing			X	
Uso con dispositivos móviles		X	X	X
Ejecución de tareas			X	X
Servicio de infrastructura				
Entorno de simulación		X	X	X
Uso con redes de sensors		X	X	X

Tabla 3.- Resumen SMA y CC, rol *Cloud Consumer*

En segundo lugar, cuando los SMA toman el rol de *Cloud Auditor,* o *Cloud Broker*, actúan como terceras partes (intermediarios) que intervienen en la relación comercial existente entre usuarios y consumidores, también disponen de una gran aplicación, principalmente como *Cloud Broker*. En este sentido, destacan las facilidades que ofrecen los SMA para la búsqueda, selección de servicios, así como para la negociación automatizada de los acuerdos en la calidad del servicio con diferentes proveedores de forma simultánea. Incipientemente se observa un rápido crecimiento del uso de SMA en la composición en servicios en nube (*Cloud manufacturing*). El uso de agentes facilita la tarea de composición de servicios ofertados por diferentes proveedores CC, lo que en sí mismo constituye una aportación relevante hacia el objetivo de federación e interoperabilidad de entornos CC, el denominado InterCloud [Buyya *et al.*, 2010a].

Aplicaciones principales	Nivel externo			
	Neg. SLA	Ofer. serv.	Comp.	Eval. Serv.
Búsqueda de proveedores				
Buscador de servicios		X	X	
Interoperabilidad	X	X		
Modelo de confianza en SLA	X		X	
Contratación de servicios	X		X	
Negociación de acuerdos SLA				
Negociación de condiciones		X		X
Monitorización de acuerdos	X		X	X
Composición de servicios				
Búsqueda de servicios compatibles		X	X	
Interoperabilidad		X		
Composición Horizontal y Vertical		X		

Tabla 4.- Resumen SMA y CC, rol *Cloud Broker*

Finalmente, cuando actúan como *Cloud Provider*, la principal aportación de los SMA se relaciona con la seguridad y privacidad de los datos, gracias a la capacidad de los agentes para monitorizar, razonar y responder proactivamente a los cambios en el entorno. Sin embargo, pese a que existen diferentes trabajos relacionados que ofertan servicios de tipo CC, tan sólo es posible encontrar en el estado del arte prometedores e incipientes trabajos relacionados con la calidad en el servicio y la provisión de recursos computacionales. Teniendo en cuenta que el entorno tecnológico que envuelve a los proveedores de servicios CC es un entorno abierto por su dinamicidad, heterogeneidad e incertidumbre; no cabe duda que la aplicación de SMA en este contexto añade características relevantes como la detección automatizada de fallos, monitorización de servicios y operaciones avanzadas, negociación automatizada en la calidad del servicio, interoperabilidad, planificación dinámica, gestión eficiente de recursos, aprendizaje, etc.

Más allá de los trabajos en los que los SMA hacen uso de servicios CC generalistas (rol *Cloud User*); los trabajos que son más interesantes en el marco de esta tesis doctoral son aquellos en los que los SMA se aplican a problemas o debilidades existentes en los sistemas CC. Estos trabajos están basados en la experiencia en tecnologías relacionadas o precedentes y su aplicación en entornos CC es directa. Sin embargo, lo que se observa es que todos ellos solucionan problemas parciales o específicos, pero que no abordan el control de un entorno CC de forma integral, haciendo frente a todos los retos

existentes de forma conjunta (gestión de infraestructura, acuerdos SLA, orquestación de servicios, etc.). Las tablas anteriormente presentadas (Tabla 4 y Tabla 5) dan una idea clara de este problema, ya que las soluciones existentes en el marco del rol *Cloud Broker* abordan los problemas abiertos desde una perspectiva externa, mientras que aquellos trabajos enmarcados como *Cloud Provider* se centrán más en los cuestiones internas de un entorno Cloud. En definitiva, no existen soluciones integrales que hagan frente a los problemas desde ambos puntos de vista. En este sentido, los SMA que pretendan gestionar un entorno CC deberán abordar los problemas existentes como un todo, de forma que sea posible abarcar todos los aspectos que estos complejos sistemas requieren.

Usos principales	**Nivel interno**			
	Monit.	**Contr.**	**Seg. Priv.**	**Gest. Infr.**
Seguridad y privacidad				
Monitorización de infraestructura	X			
Modelos de autenticación			X	
Privalegios de acceso			X	
Seguridad de la información			X	
Almacenamiento seguro			X	
Oferta de servicios				
Servicios sensibles a la calidad	X	X		
Oferta de servicios				
Gestión de nfrastructura				
Gestión de recursos	X			
Optimización de costes		X		X
Gestión de flujo de trabajo				X

Tabla 5.- Resumen SMA y CC, rol *Cloud Provider*

Para llevar a cabo esta tarea, un entorno CC de última generación basado en un SMA debe hacer uso de las últimas innovaciones existentes en el marco de los SMA, es decir, las OV [Rodriguez González, 2010] [De la Prieta *et al.*, 2009] y las modernas metodologías de desarrollo de *software* alineadas con modelos de ingeniería de desarrollo de *software* complejos que aseguran el desarrollo de productos de alta calidad, estabilidad y escalabilidad. Este tipo de sistemas se consideran adecuados en este contexto, ya que permiten dividir las diferentes responsabilidades entre grupos de agentes con objetivos específicos y modelos de razonamiento adecuados a cada problema. No obstante, pese a poder modelar el sistema como grupos de trabajo con responsabilidades y

objetivos independientes, todos trabajan en conjunto para conseguir los objetivos globales del sistema visto como un todo.

En el estado del arte no se han encontrado referencias en las que se apliquen modelos sociales en el contexto CC, por lo que en el siguiente apartado se estudiarán las OV de agentes, y las metodologías de desarrollo de *software* con el objetivo de determinar cuál es el mejor modelo de desarrollo de una plataforma CC basada en SMA organizativos (*Agent-based Cloud platform*).

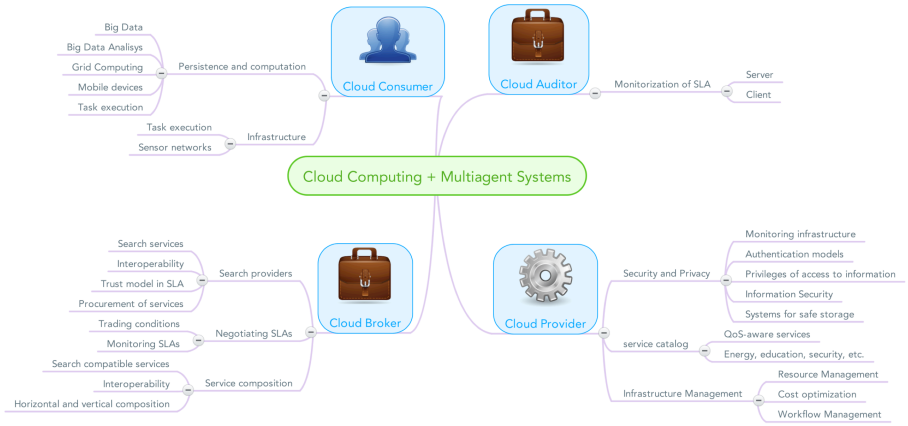

Figura 10.- Cloud Computing y sistemas multiagente

3.3 ORGANIZACIONES DE AGENTES

En los anteriores apartados de este capítulo se ha realizado una revisión histórica que facilita al lector la comprensión de los sistemas CC, así como la incipiente relación existente entre ambos. Así, después de analizar las SMA en el marco de los entornos CC durante el apartado anterior se concluyó que los SMA basados en sistemas organizativos constituyen la solución más adecuada para facilitar su implantanción en un sistema CC. Por ello, a lo largo de este apartado se hará una revisión de los SMA que siguen un modelo organizativo para el diseño de sistemas *software*.

En la literatura se distinguen dos aproximaciones en cuanto al diseño de SMA [Hannoun *et al.*, 2000] [Hübner *et al.*, 2002] [Ferber *et al.*, 2004], estas son las que se centran en el propio agente o en la organización. En el primero, son los propios agentes los que incluyen sus objetivos individuales, relacionales, etc.; mientras que en la segunda aproximación, existe un nivel de abstracción mayor, el de la organización, que es el que define las tareas, objetivos globales y relacionales durante la fase de diseño.

La primera aproximación, quizás mucho más tradicional y ligada a los SMA clásicos, sólo tiene en cuenta la perspectiva funcional en el diseño, por lo que se hace mayor énfasis en los modelos y algoritmos que permiten abordar problemas complejos [Claus *et* Boutilier, 1998] [Stone *et* Veloso, 2000]. Siguiendo esta perspectiva, resulta difícil predecir el comportamiento individual, y por ende, el comportamiento agregado del sistema en su conjunto [Woods *et* Barbacci, 1999]. Este hecho, unido a la capacidad de situarse en un sistema abierto, que es intrínsecamente impredecible, convierte a los SMA en entornos difíciles de aplicar en entornos reales por la dificultad de coordinación [Ciancarini *et al.*, 2000].

Sin embargo, la segunda aproximación, trata de resolver el problema de diseñar sistemas *software* de gran envergadura [Zambonelli *et al.*, 2003], añadiendo al paradigma tradicional nuevos capacidades como roles, normas, organizaciones, etc. que maximizan sus cualidades intrínsecas (autonomía, habilidades sociales, reactividad y pro-actividad). Este conjunto de aproximaciones se puede dividir en los siguientes grupos en función de los mecanismos utilizados [De la Prieta *et al.*, 2009]:

- **Mecanismos basados en interacciones directas.** Proporcionan principios básicos de comunicación y cálculos locales de los agentes para proporcionar un estado global del sistema [Zambonelli *et al.*, 2004].
- **Mecanismos basados en interacciones indirectas.** Consiguen comportamientos complejos a partir de interacciones entre los agentes a través del entorno [Reitbauer *et al.*, 2004].

- **Mecanismos basados en refuerzo.** Basados en la capacidad de los agentes para modificar su comportamiento en función de las entradas (positivas o negativas) [Weyns *et al.*, 2004].
- **Mecanismos basados en cooperación.** Basados en la descomposición de agentes para llevar a cabo tareas complejas; y la composición de agentes para llevar a cabo tareas simples [Capera *et al.*, 2003].
- **Mecanismos basados en arquitecturas genéricas.** Basadas en meta-modelos y arquitecturas de referencias que son modificadas (automática o manualmente) en función del contexto de aplicación [Razavi *et al.*, 2005].

Dentro de este conjunto de propuestas, el modelo que más ha sido estudiado en los últimos años es, sin duda, el de las OV de agentes [Kolp *et al.*, 2006]. Este tipo de mecanismos modelan los SMA siguiendo un esquema semejante a la de las organizaciones humanas, acotando la impredecibilidad del sistema dentro de un conjunto de normas institucionales, haciendo posible la evolución del SMA gracias al uso de sociedades artificiales como abstracción de nivel superior [Annunziato *et* Pierucci, 2003]. Las OV se enmarcan dentro del mecanismo de comunicación directa, sin embargo, las últimas referencias en la literatura apoyan la definición de estas organizaciones mediante arquitecturas genéricas (meta-modelos) [Garcia-Ojeda *et al.*, 2008], así como en el uso del entorno no sólo como medio de interacción indirecta sino también como parte del propio SMA [Ricci *et al.*, 2007].

El nacimiento de esta nueva metodología para diseñar SMA está motivado por la autonomía de los propios agentes, lo que les convierte en entidades que, por su naturaleza son impredecibles. Este hecho conlleva a que el propio SMA se convierta en un entorno cuyo nivel de variabilidad está condicionada por el medio en el que se sitúa [Zambonelli *et al.*, 2003], esta característica se acentúa aún más si cabe en un sistema CC donde el entorno es altamente complejo e impredible, dependiente en gran medida de las condiciones externas y las necesidades de los usuarios finales.

El uso de un nivel mayor de abstracción para el diseño de SMA en el marco de estos entornos complejo, mediante el uso de las OV de agentes inteligentes, acota la autonomía de los agentes a los intereses comunes de la organización de la que forman parte, más allá de sus intereses individuales. De este modo, un agente dentro de una sociedad necesita considerar no sólo su propio entorno, sino también el comportamiento del sistema como un todo, así como los agentes influyen en el comportamiento del resto.

Hoy en día, como no podría ser de otro modo, el diseño de SMA se orienta al uso de modelos organizativos que facilitan el diseño e implementación de los sistemas abiertos [Dignum *et al.*, 2002]. Tal y como ya se ha indicado, los sistemas abiertos se caracterizan por la heterogeneidad de sus participantes,

la limitación en cuanto a la confianza debido a la existencia de objetivos individuales en conflicto, y finalmente debido a la gran probabilidad de disconformidad con las especificaciones [Giret *et al.*, 2005]. El dinamismo es otra de las características clave de este tipo de sistemas [Rodríguez, 2010]. Estas dos propiedades (dinamismo y heterogeneidad) provocan que modelar un sistema abierto constituya hoy en día un reto [Zambonelli *et al.*, 2004] [Reitbauer *et al.*, 2004] [Weyns *et al.*, 2004] [Razavi *et al.*, 2005] [Sims *et al.*, 2008] [Ommicini *et al.*, 2008] [DeLoach *et* Garcia-Ojeda et al., 2010]. Para abordar el diseño de este tipo de sistemas es necesario disponer de nuevos modelos y mecanismos, que permitan abordar el diseño de sistemas con capacidad de adaptación dinámica a cambios en la organización y en el propio entorno [Dignum *et* Dignum, 2006]. Además estas nuevas teorías, deben estar apoyadas por herramientas que guíen el proceso de diseño.

Antes de detallar en profundidad todos los aspectos que envuelven a esta aproximación de diseño y desarrollo de SMA, se presentará la Teoría de Organizaciones humanas que sustenta el diseño SMA modernos.

3.3.1 ORGANIZACIONES ARTIFICIALES, UNA PERSPECTIVA SOCIOLÓGICA

Las estructuras organizativas han sido estudiadas por la *Teoría de la Organización* [Mintzberg, 1992] [Morabito *et al.*, 1999], con el objetivo de comprender la estructura y el diseño de un organización, así como las *Alianzas Estratégicas* [Gomes-Casseres, 1996] [Dossauge *et* Garrete, 1999] entre los diferentes integrantes que permitan alcanzar los objetivos perseguidos. Estos estudios son relativamente reciente, ya que no han existido grandes organizaciones hasta los inicios del siglo XX.

Antes de continuar, con el análisis es necesario definir el concepto principal, la organización [Morabito *et al.*, 1999]:

Una organización es una entidad social conscientemente coordinada, con un límite relativamente identificable, que funciona de forma continua para lograr una o un conjunto de metas comunes.

El estudio de las organizaciones trae consigo innumerables ventajas, entre las que cabe destacar [Daft, 2008]:

- Permite producir bienes y servicios de forma eficientemente.
- Reúne los recursos para alcanzar las metas deseadas.
- Facilita la innovación.
- Permite considerar la influencia que conlleva los cambios en el entorno.
- Crea un valor para los clientes, empleados y propietarios.
- Permite gestionar los retos dentro de un entorno social diverso, ético, coordinado y motivado.

El estudio o diseño de una organización es un proceso complejo, para el cuál es necesario tener en cuenta una gran cantidad de puntos de vista que se organizan en las siguientes dimensiones [Daft, 2008]:

- **Dimensiones estructurales.** Proporciona un método para describir las características internas de la organización. Entre las principales subdimensiones estructurales, cabe destacar: la formalización, la centralización, la especialización, la jerarquía de autoridad, el profesionalismo y las proporciones de personal.
- **Dimensiones contextuales.** Caracteriza a la organización como un todo, incluyendo tamaño, tecnologías, entorno y objetivos. Entre las dimensiones contextuales cabe destacar la cultura, el entorno, los objetivos y estrategia, el tamaño y la tecnología.

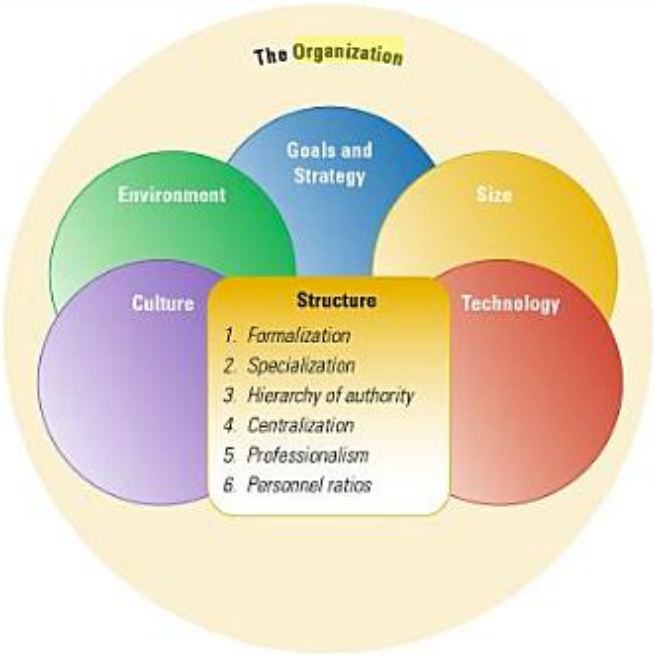

Figura 11.- Teoría de la organización [Daft, 2008]

Si se extrapola el concepto de organización (según la teoría de organizaciones) al contexto de los SMA, una organización puede verse como un conjunto de entidades reguladas por mecanismos de orden social, que persiguen objetivos comunes. Además, la teoría moderna de organizaciones, y la evolución de los SMA convergen basando su desarrollo en los mismos principios directores [Daft, 2008]:

- Se evoluciona el control formal de los sistemas y la información, hasta el desarrollo de organizaciones con información compartida y distribuida.

- La toma de decisiones y el control que gobierna la organización sigue una estrategia colaborativa, en detrimento de las estrategias competitivas previas.
- La organización es en sí misma una estructura que evoluciona y que se adapta a los cambios que se producen en el entorno, promoviendo la cultura de la adaptación en detrimento de las estructuras rígidas.

La funcionalidad principal de una organización, es permitir a sus miembros coexistir en un entorno compartido y llevar a cabo sus respectivos objetivos, cooperando (o no) con el resto de miembros. Este entorno común de consecución de los objetivos individuales permite que el diseño de SMA basándose en el concepto de organización facilite su comprensión, diseño y por lo tanto su posterior implementación [Artikis *et al.*, 2001], [Castelfranchi, 2000], [Zambonelli *et al.*, 2001]. No obstante, aunque existe una gran cantidad de elementos convergentes en ambas teorías, también se pueden encontrar una gran variedad de diferencias, tal y como sigue [Parunak *et al.*, 1998] [Gilbert, 1999]:

- En las sociedades artificiales los agentes emulan comportamientos que tratan de satisfacer un conjunto de requisitos, mientras que en la Teoría de Organizaciones está basada en un conjunto de modelos matemáticos.
- La Teoría de Organizaciones modela funciones de utilidad asumiendo el comportamiento racional de los actores. Sin embargo, los SMA utilizan un modelo de representación donde los actores pueden tomar decisiones racionales, pero también no racionales, dependiendo de la representación que tengan los agentes sobre su entorno.
- La Teoría de Organizaciones representa a los actores de forma homogénea, en cambio, los SMA debido a que se sitúan en entornos abiertos, permitiendo la agregación de una gran cantidad de sistemas con modelos arquitectónicos diferentes.

Todas estas diferencias, invitan a pensar que aunque las sociedades artificiales y las OV están sustentadas inicialmente sobre la Teoría de Organizaciones, en realidad conforman una nueva teoría, acompañada de un conjunto de modelos y metodologías para el desarrollo de sistemas *software* complejos

3.3.2 SOCIEDADES ARTIFICIALES Y ORGANIZACIONES VIRTUALES

Las sociedades artificiales y las OV son términos estrechamente relacionados, a lo largo de este apartado se clarificará la relación entre los conceptos tomando como base la literatura existente. En primer lugar, se presentará el concepto de *Sociedad Artificial*, presente en los resultados de una

gran variedad de autores [Annunziato *et* Pierucci, 2003] [Davidsson, 2001] [Davidsson *et* Johansson, 2006] [Artikis *et* Pitt, 2001]. Así, por ejemplo, [Artikis *et* Pitt, 2001] caracterizan formalmente una sociedad artificial en base a un conjunto de agentes, restricciones en la sociedad, un lenguaje de comunicación, los roles que los agentes pueden desarrollar y el conjunto de propietarios (*owners*) de los agentes. No obstante, la definición más certera desde nuestro punto de vista es el que proponen Annunziato *et* Pierucci [Annunziato *et* Pierucci, 2003]:

Una sociedad artificial está definida como un conjunto de entidades artificiales interrelacionadas e interactuantes, que se rigen bajo determinadas reglas y condiciones

Dentro de la literatura, Davidsson *et* Johansson [Davidsson, 2001] [Davidsson *et* Johansson, 2006] propone una clasificación de las sociedades artificiales basándose en la (i) *apertura*, que es la posibilidad de que un agente pueda unirse a la sociedad; la (ii) *flexibilidad*, que indica el grado de restricción que tiene el agente en su comportamiento debido a las normas de la sociedad; la (iii) *estabilidad*, que es una medida de la previsibilidad de las acciones; y, finalmente, la (iv) *confianza*, que mide el grado en el que el agente puede confiar en la sociedad.

A partir de estas características Davidsson *et* Johansson propone la existencia de cuatro tipos de sociedades. Los dos primeros tipos de sociedades son los siguientes:

- **Sociedades abiertas**. Que son flexibles lo que las convierten en sociedades poco estables y en las que es difícil confiar. En este tipo de sociedades los agentes entran en la organización sin ningún tipo de restricción, para ello tan sólo tienen que interaccionar con alguno de los agentes que la forman. Para que la comunicación entre las diferentes entidades se pueda llevar a cabo deben compartir el mismo lenguaje de comunicación y existir un conjunto mínimo de roles. En el caso de no tener este mínimo conjunto de características (comunicación y roles) serían *sociedades anárquicas*.
- **Sociedades cerradas.** Al contrario que las sociedades abiertas son estables y confiables, sin embargo, esto es así dado que son cerradas y no son flexibles. Este tipo de sociedades pueden denominarse también *sociedad fija o inalterable*.
  Es el tipo de sociedad más utilizada para el diseño de SMA clásicos, ya que en ellas existe un conjunto de agentes que se conocen a priori, cada agente tiene sus propios objetivos (que no tienen que ser comunes) y que interaccionan de forma distribuida para resolver problemas, teniendo en cuenta que pueden confiar entre sí.

Para el desarrollo de sistemas organizativos estas sociedades no son las más apropiadas, debido a sus restricciones intrínsecas, ya sea por la flexibilidad, o la falta de ella. Las OV se basan en otros dos tipos de sociedades más flexibles, permitiendo la entrada de nuevos miembros, donde la estabilidad de la misma se regula a través de algún tipo de mecanismo o institución:

- **Sociedades semi-abiertas.** En la que se introduce el concepto de portero, las entidades externas que pretendan acceder a la sociedad deberán contactar con dicho portero y demostrar que son entidades seguras, o al menos que su grado de seguridad no van alterar la confiabilidad y estabilidad del sistema.
  Adicionalmente, estas entidades también pueden monitorizar a los agentes internos, lo que permite acotar a la propia sociedad y además auditar el estado de la misma
- **Sociedades semi-cerradas.** En este tipo de sociedades, los agentes no están autorizados a entrar. Sin embargo, si que puede existir un agente de la propia sociedad que realice las tareas del agente externo. Igualmente, existe una entidad que interactúa con los agentes externos, y que crea los agentes internos mediadores dentro de la sociedad para realizar las tareas del agente externo.

Una vez que se ha acotado el concepto de sociedad artificial, el otro concepto, el de *organización* puede verse como un conjunto de entidades reguladas por mecanismos de orden social, de forma que habiliten la consecución de objetivos comunes. Al igual que con las sociedades, existe una gran variedad de literatura acerca de las organizaciones de agentes [Gasser, 1992] [Wooldridge *et al.*, 2000] [Rodriguez González, 2010] [Ferber *et al.*, 2004], así por ejemplo, [Wooldridge *et al.*, 2000] enuncian la siguiente definición:

Una organización es un conjunto de roles con cierta relación entre sí, que interaccionan con otros roles a través de unos patrones institucionales sistematizados.

No obstante, desde nuestro punto de vista, la mejor forma de definir una organización de agentes es atendiendo a sus características [Ferber *et al.*, 2004]:

- Una organización está formada por agentes (individuales) que manifiestan su comportamiento.
- La organización puede ser dividida en suborganizaciones que pueden estar solapadas (suborganizaciones o grupos).

- El comportamiento de los agentes está relacionado a nivel funcional con la organización como un todo, definiendo su funcionalidad como un rol a tener dentro de la organización.
- Los agentes se relacionan de forma dinámica.
- Los tipos de comportamiento se relacionan mediante enlaces entre roles, tareas y protocolos.

Dentro de estas características, el concepto clave es el de rol, que es una descripción abstracta del comportamiento de los agentes, incluyendo obligaciones, restricciones y habilidades. Así mismo, el rol debe incluir la descripción de los patrones de interacción entre los diferentes agentes que conforman el sistema. No obstante, tal y como se presentará a continuación los modelos organizativos van mucho más allá, introduciendo una gran variedad de capacidades que permiten el uso de SMA en una gran variedad de contextos.

Este enfoque parece adecuado al problema a modelar en el marco de este trabajo de investigación ya que es posible introducir nuevos técnicas para el control y monitorización de sistemas CC donde la colaboración y las estrategias competitivas sustituyen al modelo jerárquico en la de toma de decisiones. Gracias a lo cuál es posible que los agentes desarrollen modelos de información compartida y distribuida, lo que permite reducir el espacio de datos necesario para llevar a cabo el proceso de decisión, reduciendo por tanto, la complejidad en la toma de decisiones. Además, no sólo se reduce su complejidad, sino que el modelado basado en roles y objetivos permite hacer frente a la autonomía de los individuos que forman parte de la organización, ya que las decisiones de los agentes individuales no sólo atenderán a las intenciones individuales que les permitan satisfacer sus deseos personales, sino que también estarán orientadas a satisfacer los objetivos globales de la organización u organizaciones de las que forman parte. Finalmente, la abstracción e independencia del entorno de la organización y los agentes permite no solo abtraer su complejidad, sino también reducir la dependencia entre los órganos de control y actuación.

3.3.3 MÉTODOLOGÍAS Y MODELOS ORGANIZATIVOS

En los últimos años, el auge de los conceptos organizativos ha experimentado un proceso de rápido desarrollo. Así en el estado del arte se pueden encontrar una gran variedad de modelos y metodologías para su desarrollo [Horling *et* Lesser, 2004], algunos de ellos toman como base las sociedades artificiales. El desarrollo de estas metodologías, cuyo objetivo es facilitar las tareas de diseño software basado en agentes, también ha propiciado la evolución de la propia Teoría de Agentes [Wooldridge *et* Jennings, 1995].

El proceso de desarrollo de un sistema CC mediante el uso de estas novedosas técnicas requiere que inicialmente se realice un estudio de los principales modelos existentes basados en OV de SMA. Esto es así debido a que cada metodología incluye herramientas, técnicas y modelos propios para el modelado, pero en la mayoría de los casos estas metodologías tienen una gran cantidad de rasgos comunes, en lo que diferentes autores han definido como *meta-modelo* del paradigma multiagente [DeLoach, 2009] [Bonjean *et al.*, 2012].

Así, gracias a este estudio previo será posible seleccionar aquella metodología y su meta-modelo asociado que mejor se adapte al problema a resolver. Por tanto, a continuación se presenta una breve descripción de las corrientes metodológicas principales:

- **MAS-CommondKADS** [Iglesias *et al.*, 1998] se basa en la metodología de propósito general *CommondKADS* [Schreiber *et al.*, 2000]. Fue una de las primeras metologías de desarrollo de *software* en el marco de los SMA. La metodología se fundamenta en el desarrollo de siete modelos principales: agente, tareas, experiencia, organización, coordinación, comunicación y diseño.
- **MESSAGE/UML** [Caire *et al.*, 2002] fue una de las primeras metodologías que incluyó el ciclo de vida del desarrollo *software* y el modelado mediante UML.
- **GAIA** [Wooldridge *et al.*, 2000] es, sin lugar a dudas, una de las principales metodologías de desarrollo de SMA y que ha sido ampliamente utilizada en diferentes trabajos y contextos [Moraitis *et* Spanoudakis, 2006] [Moraitis *et* Spanoudakis, 2004] [Spanoudakis *et al.*, 2009] [Carrascosa *et al.*, 2008]. Posteriormente a su nacimiento, la metodología fue revisada para incluir conceptos organizativos, la noción de entorno y un conjunto de herramientas y técnicas que faciliten su uso en entornos abiertos; a esta revisión se la ha denominado **GAIA II** o **GAIAexOA** [Zambonelli *et al.*, 2003]. Finalmente, existe una nueva actualización más reciente que se ha centrado en el diseño de sistemas adaptativos [Cernuzzi *et* Zambonelli, 2006].
- **ROADMAP** [Juan *et al.*, 2002] que partiendo de la metodología GAIA original sigue su misma evolución de forma paralela, ya que añade los conceptos y técnicas necesarias para su uso en entornos abiertos. Posteriormente, se refina esta versión inicial para tener en cuenta la adaptación dentro de estos entornos abiertos [Juan *et* Sterling, 2004].
- **PROMETHEUS** [Padgham *et* Winikoff, 2002] [Padgham *et* Winikoff, 2003] es una metodología que centra sus intereses en cubrir todas las fases del diseño y desarrollo desde el punto de vista de la Ingeniería del *Software*, siguiendo une esquema similar al *software* tradicional. El objetivo es acercar el modelado de SMA a su posterior implementación.

No obstante, el modelo propuesto en Prometheus es limitado, aunque trabajos más modernos han demostrado que puede ser utilizado favorablemente si se combina con metodologías más modernas [Pavón et Gómez-Sanz, 2003] como **INGENIAS** [Gascuena *et al.*, 2009] que se presentarán a continuación.

- **AALADIN** [Ferber *et* Gutknecht, 1998], fue una de las primeras metodologías. Su principal característica es su simplicidad, ya que sólo incluye tres conceptos principales: Agente, Grupo y Rol. Precisamente estos tres conceptos son la base de la segunda versión de la metodología, denominada **AGR (Agent-Group-Role)** [Ferber *et al.*, 2004] que se centra en el diseño basado en modelos organizativos. La última revisión de esta metodología es el modelo **AGRE (Agent-Group-Role-Environment)** [Ferber *et al.*, 2005] que incluye el concepto de entorno como novedad.
- **SODA** [Omicini *et al.*, 2001] propone una metodología de desarrollo de sistemas *software* orientados a Internet, en la cuál se tiene en cuenta tanto los aspectos del propio agente, como de los de la sociedad, y en el que también está presente el concepto de entorno. Para ello, la metodología se centra en las tareas que un agente o conjunto de agentes tienen que realizar. Recientemente esta metodología ha sido adaptada [Molesini *et* Omicini, 2008] para su uso con *SPEM 2.0* [OMG, 2008] que es una metodología de desarrollo *software*.
- **MOISE** también puede ser considerada una de las principales metodologías, además de ser una de las pioneras. Se distingue del resto en que no sólo se centra en los conceptos sociales, sino que también tiene en cuenta las características del agente y las tareas a realizar (misiones) [Hannoun *et al.*, 2000]. Posteriormente a su nacimiento, también ha sido refinada para cubrir las debilidades encontradas en la versión inicial, como la falta de planes globales y la dependencia entre estructura y funcionalidad. Esta nueva versión se ha denominado **MOISE+** [Hübner *et al.*, 2002]. Recientemente también esta segunda versión se ha complementado [Hübner *et al.*, 2010] incluyendo la Teoría de Agentes y Artefactos (**Agents and Artifacts –A&A–**) [Ricci *et al.*, 2006] [Ricci *et al.*, 2007]. Esta teoría (o metodología) facilita la tarea de modelar el entorno, permitiendo encapsular las interacciones del SMA con el exterior a través de un nuevo componente denominado *artefacto*. Esta aproximación ha dado lugar a la metodología **ORA4MAS** [Hübner *et al.*, 2010] que trata de articular la adaptación de la propia organización.
- **TROPOS** [Giunchiglia *et al.*, 2003] [Bresciani *et al.*, 2004] es una metodología que difiere de las anteriores en que propone el desarrollo de organizaciones multiagente basándose en la teoría de organizaciones humanas [Mintzberg, 1992]. En este sentido, aboga por el uso de una de las siguientes topologías *Flat-Structure*, Cadena de valores, Pirámide,

Matriz, *Structure-in-Five*, *Co-optation*, *Joint Venture*, *Bidding*, *Arm's Length* y *Hierarchical Contracting*. La metodología se basa en el entorno de modelado **i*** [Yu, 2011].

Al igual que las anteriores ha sido refinada utilizando para ello las últimas referencias en el marco de la teoría de organización [Kolp *et al.*, 2006].

- **OperA** [Dignum, 2004] es una metodología desarrollada desde su primera versión teniendo en cuenta el concepto de sociedad, objetivos, estructuras y normas que rigen esta familia de metodologías de desarrollo *software*. Su principal aportación consiste en el soporte que ofrece a la dinamicidad y heterogeneidad del tiempo de vida de los agentes en el marco de los sistemas abiertos. La metodología fue extendida en el modelo **OMNI** [Vázquez-Salceda *et al.*, 2005] para integrar el marco normativo de **HARMONIA** [Vazquez-Salceda, 2003] el cuál aporta una mayor potencia semántica a la definición de las normas de una sociedad de agentes.
- **ADELFE** [Bernon *et al.*, 2002] [Bernon *et al.*, 2003] sigue el Proceso Unificado [RUP, 2003] para el desarrollo de SMA incluyendo las fases de requisitos, análisis y diseño. También utiliza la notación UML [Booch *et al.*, 1998] y AUML [Bauer *et al.*, 2001] para la descripción de los sistemas construidos.
- **PASSI** [Cossentino, 2005] es una metodología para el desarrollo *software* que incluye todas las fases en el desarrollo *software* y que además es iterativa. Sin embargo no está adaptada a los conceptos de sociedad artificial, entorno, normas, etc. Su segunda versión [Chella *et al.*, 2006] se centra en la inclusión de los principios de las metodologías ágiles [Martin et al., 2003].
- **OMACS** [Deloach *et al.*, 2008] [DeLoach, 2009] es la que se puede denominar una metodología de nueva generación, ya que no sólo está desarrollada sobre la base que proporcionan las sociedades de agentes, sino que modela la propia reorganización de la sociedad incluyendo cambios estructurales en la misma.
- **MASE** [DeLoach, 1999] [DeLoach *et al.*, 2001] fue una de las primeras metodologías. La principal innovación en el momento de su nacimiento fue la inclusión de las fases habituales del desarrollo *software*. No obstante, pese a ello, el modelo desde un punto de vista de agentes era limitado. Para superar estos problemas, recientemente ha sido refinada mediante la inclusión del metamodelo de una de las metodología más avanzadas como es OMACS [DeLoach, 2009] y mejorando la metodología de desarrollo *software* inherente, ya que se basa en *SPEM 2.0*. El nuevo modelo resultante se denomina **O-MASE** [DeLoach *et* Garcia-Ojeda, 2010].

- **INGENIAS** [Gomez-Sanz *et* Fuentes, 2002] [Pavón *et* Gómez-Sanz, 2003], basada inicialmente en la metodología MESSAGE/UML [Caire *et al.*, 2002]. Propone un modelo basado en ingeniería y que incluye todo el ciclo del *software*, así como un conjunto de herramientas que dan soporte al proceso. Esta metodología se ha ido refinando progresivamente [Pavón *et al.*, 2005] [Fernández-Caballero *et* Gascueña, 2010] [García-Magariño *et al.*, 2011], gracias a lo cuál, en las últimas versiones ya es posible hacer frente a la necesidad de readaptación del modelo de agentes [García-Magariño *et al.*, 2011].
  A partir de esta metodología ha nacido también el modelo **ANEMONA** [Botti *et* Giret, 2008] que basa el diseño de sistemas multiagente mediante Sistemas Holónicos de Fabricación (HMS) [Hopf *et* Schaeffer, 1997].
- **GORMAS** [Argente, 2008] [Argente *et al.*, 2011] es una metodología de última generación basada en las OV para el diseño de SMA, incluyendo tanto las fases de análisis y diseño de sistemas *software*, así como una fase de diseño de la dinámica que facilita la descripción de las interacciones de los agentes individuales. Estas fases permite describir seis modelos de especificación (organización, actividad, interacción, entorno, agente y normativo) que abarcan las características del metamodelo básico de los SMA sociales.
  El modelo descrito por la guía metodológica **GORMAS** extiende a los modelos propuestos en la metodología **ANEMONA** [Botti *et* Giret, 2008], que a su vez es una extensión de **INGENIAS** [Pavón *et* Gómez-Sanz, 2003]. La metodología también está alineada con SPEM.
- **E-Institutions** [Esteva *et al.*, 2001] que asocia el modelado de SMA basándose en instituciones electrónicas que a su vez se basan en instituciones reales. El modelo ha sido refinado para incluir normas [Garcia-Camino *et al.*, 2005] y su aplicación en entornos abiertos [Arcos *et al.*, 2005].

3.3.4 ANÁLISIS DE LOS MODELOS ORGANIZATIVOS

Una vez se han presentando las principales metodologías para el diseño y (en algunos casos) desarrollo de SMA, a simple vista se puede observar que todas ellas tienen un conjunto de rasgos comunes, rasgos que han ido evolucionando a medida que se han ido revisando a lo largo del tiempo. Así, las metodologías clásicas tienen un claro enfoque de diseño centrado en el propio agente (GAIA, MOISE, ALAADIN, MASE entre otras). Mientras que revisiones de estas metodologías han ido evolucionando hasta un modelo de diseño organizativo (GAIAexOA, MOISE+, e-Institutions, AGRE, GORMAS, etc.).

Dentro de este último grupo de metodologías, antes de comenzar el diseño y posterior desarrollo de un SMA basado en OV especialmente orientado a modelar el problema propuesto en este trabajo de investigación (monitorización y control de un sistema CC) resulta necesario realizar un análisis de los conceptos, herramientas, técnicas y mecanismos que incorporan estas metodologías para el desarrollo de SMA. Este análisis, descrito a lo largo de este apartado, permite estudiar los metamodelos propios de las metodologías presentadas. Gracias a lo cuál es posible identificar rasgos comunes y opuestos, lo que por tanto, permite seleccionar la metodología, y por lo tanto la corriente científica, que se adecúa mejor al problema a resolver.

Todas las metodologías están orientadadas al desarrollo de sistemas *software* complejos, por lo que incluyen diferentes modelos de especificación y descripción de requisitos. Sin embargo, resulta contradictorio que muchas de ellas no incluyan un proceso de ingeniería al uso (requisitos, análisis, diseño, implementación y pruebas). Así en este sentido, tres grandes grupos claramente diferenciados en tanto en cuanto a la aplicación de modelos de ingeniería iterativos:

- No incluye ningún modelo de ingeniería fundamentado en ingenieria, como TROPOS, MOISE+, AGRE, etc.
- Incluyen algunas fases de desarrollo, normalmente sólo abarcan el modelo de dominio y de la solución. Es decir, describen como debería ser el análisis y el diseño, pero no tienen en cuenta la implementación. Entre las metodologías que se encuentran en este grupo GAIAexOA o GORMAS.
- Incluyen un proceso de ingeniería completo como PASSI, ADELFE, INGENIAS, etc. Por su parte, metodologías como SODA, O-MASE, GORMAS, recientemente han evolucionado para incluir el proceso de desarrollo de ingeniería SPEM 2.0 [OMG, 2008].

Dentro de estos modelos organizativos, como ya se ha indicado muchos autores hablan del meta-modelo del paradigma de agentes [DeLoach, 2009] [Ferber *et* Gutknecht, 1998] [Juan *et* Sterling, 2004] [Bonjean *et al.*, 2012]. El meta-modelo incluye todos aquellos conceptos, artefactos y herramientas que permiten diseñar un SMA siguiendo una pespectiva organizativa. A continuación en la Figura 12 se presenta el meta-modelo de la metodología OMACS [DeLoach, 2009].

En el meta-modelo de los sistemas organizativos, existen dos conceptos clave, el de **rol** y el **organización** (o grupo), además del propio concepto de **agente**. Estos tres conceptos organizativos, además de definir una metodología por sí solos (AGR [Ferber *et al.*, 2004]), permiten modelar en líneas generales un SMA. Es habitual dividir el estudio de los sistemas organizativos en dos niveles de abstracción [Wooldridge *et al.*, 2000] [Ferber *et*

al., 2004], el *estructural* (o macro) que tiene en cuenta los aspectos dinámicos y de organización (roles y grupos). Y, el nivel *concreto* (o micro) que conforma la definición de bajo nivel de los agentes (tareas, planes, etc.).

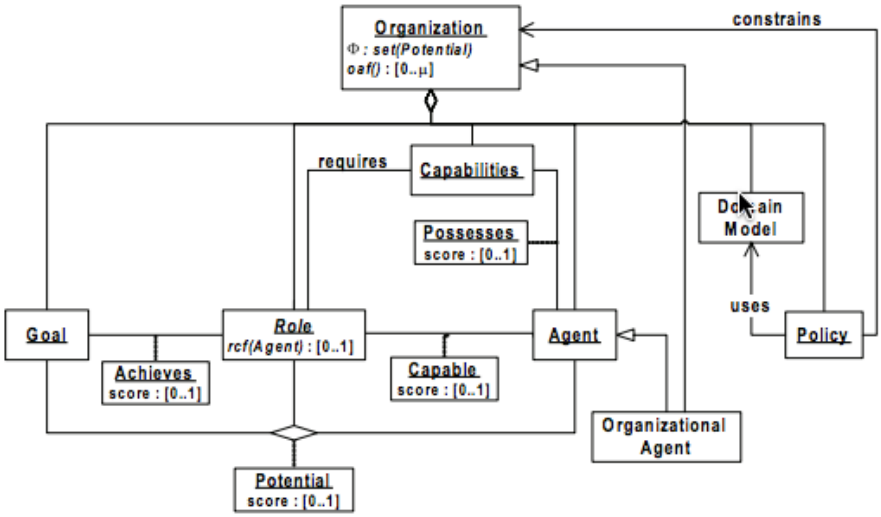

Figura 12.- Meta-modelo Sistemas Organizativos ([DeLoach, 2009])

Por un lado el concepto de rol, presente en la Teoría de Agentes desde sus inicios, define de forma abstracta el comportamiento de las entidades que forman parte del sistema (agentes). Este comportamiento se define por medio de las obligaciones/responsabilidades y requisitos que son necesarios satisfacer por el rol, pero también mediante las capacidades y derechos que se le presuponen al individuo o agente que adquiriera *(playing/enacting)* dicho rol en el sistema, de forma que estas habilidades le permitan satisfacer las obligaciones y requisitos asociados [Ferber *et al.*, 2004] [Wooldridge et al., 2000]. Este concepto de *adquisición* del rol por parte de un agente individual resulta de gran importancia, ya que representa la entidad abstracta (no instanciable), mientras que el agente representa al individuo concreto y con una arquitectura determinada (reactiva, deliberativa, modelo BDI, etc) [Wooldridge, 2002]. En sentido, los agentes también pueden tener una planificación [Georgeff *et* Lansky, 1987] definida mediante una representación actual del mundo, un objetivo y la secuencia de acciones (o misiones [Hübner *et al.*, 2002]) para conseguirlo.

Además, en el marco de los sistemas abiertos, los agentes son entidades que pueden entrar o salir de la organización, pero que para poder jugar un rol específico deben tener unas características concretas definidas por el propio rol, así como seguir un protocolo de entrada en la organización (sociedades semi-abiertas o semi-cerradas) como ya se ha presentado anteriormente [Davidsson, 2001] [Davidsson *et* Johansson, 2006]. En este sentido, existen

metodologías que hablan de roles institucionales y roles externos [Dignum, 2004] en función del tipo de agentes que existan en cada momento de vida del sistema.

Por otro lado, la definición del concepto de organización resulta mucho más compleja, en primer lugar, la organización tiene que describir los *objetivos* para lo que ha sido diseñada, teniendo en cuenta que estos objetivos tienen que estar alineados con los de sus miembros (roles o suborganizaciones) y que en cierta medida aportan racionalidad al sistema en su conjunto. Este diseño, aunque con diferentes nomenclaturas según la metodología, tiende a incluir aspectos sociales, de comunicación, de interacción y normativos [Dignum, 2004]. A continuación se describe cada uno de ellos:

- Los **aspectos sociales** (o estructurales [Ferber *et al.*, 2004]) se refieren a la descripción del conjunto de roles, grupos (asociaciones de roles) y la relación entre ellos. En cuanto a las relaciones existentes entre roles y grupos (recursivamente) destacar que hay autores que han definido un conjunto de estructuras sociales [Horling *et* Lesser, 2004] que permiten modelar las interacciones entre los miembros. Entre las principales estructuras destacan la siguientes: jerarquías, holarquías, coaliciones, equipos, congregaciones, sociedades, federaciones, mercados, matrices y organizaciones compuestas.
  Sin embargo, también existen trabajos en el que simplemente se definen posibles relaciones entre miembros [Dignum, 2004] como dependencia, jerarquía, uso, etc.
- Los **aspectos de comunicación** hace referencia a los medios o el lenguaje que hace posible el intercambio de información. Es decir un lenguaje de representación de conocimiento (habitualmente representado mediante una ontología) y un lenguaje de comunicación. La secuencia de comunicación entre dos agentes se denomina ilocución [Esteva *et al.*, 2001], acto de comunicación [Dignum, 2004] o enlace [Hübner *et al.*, 2002], y puede tener diferentes fines según la filosofía del mensaje [Dignum, 2004]: representación, prohibición, permisos, declaración, expresivos, compromisos o directivos.
- Los **aspectos de interacción** se refieren a como los roles colaboran para alcanzar objetivos comunes. Es decir, dado que pueden existir objetivos que no se pueden alcanzar individualmente, y que requieren de la combinación de varios agentes para la consecuención, es necesario describir una estructura de interacción que permita articular o regular la consecución de los sub-objetivos individuales que a su vez hagan posible la consecución de objetivos de más alto nivel. Destacar que estas secuencias de interacción evolucionan en la medida que avanzan las relaciones entre los roles y grupos. A este conjunto articulado y reglado

de ilocuciones entre varios agentes para conseguir un objetivo común se le denomina habitualmente escena [Dignum, 2004].

- Finalmente, en cuanto a los **aspestos normativos** cabe destacar que es uno de los grandes pilares de los SMA organizativos, e incluso existe una metodología que se basa en este concepto (HARMONIA [Vazquez-Salceda, 2003]). Las normas (o patrones institucionales [Zambonelli *et al.*, 2003] [Ferber *et al.*, 2004]) permiten establecer una relación de confianza entre los miembros de una organización, ya que permiten acotar el libre albedrío de los agentes individuales. Las normas deben ser acatadas por cualquier agente externo que quiera formar parte de una organización. En resumen, definen formalmente las obligaciones, prohibiciones y permisos de los miembros y de las comunicaciones entre estos.

Además de los conceptos que se acaban de presentar (rol, organización, normas y estructuras sociales), los SMA organizativos incluyen de forma habitual otro concepto clave, el de entorno. La Teoría de Agentes, tradicionalmente concibe al agente como una entidad que planifica sus acciones en función de las percepciones del medio. Sin embargo, la complejidad creciente del propio medio en el marco de los sistemas abiertos (dinámicos, heterogéneos e impredecibles) no sólo puede llegar a convertir en impredecible al SMA, sino que las interacciones con él resultan complejas.

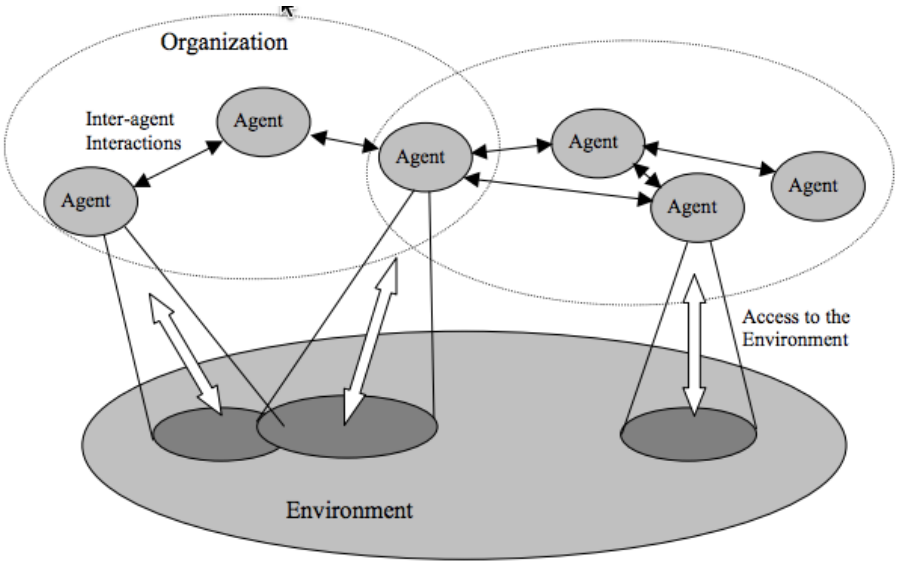

Figura 13.- Concepto de entorno en sistemas organizativos [Zambonelli *et al.*, 2003]

Por ello, las metodologías recientes modelan al entorno dentro del propio SMA, de forma que es posible diseñar el sistema teniendo en cuenta la

complejidad de éste. En este sentido cabe destacar la aproximación de la metodología *Agents and Artifacts* (A&A) [Ricci *et al.*, 2007] que modelan las interacciones con el entorno mediante un mecanismo denominado artefactos organizativos que encapsula la observación de las propiedades y eventos del entorno y articula las acciones de control y uso de los recursos que proporciona el propio entorno.

Todo el paradigma anteriormente presentado supone una evolución de los SMA, de cara a facilitar su aplicación en entornos abiertos, así como los nuevos requisitos que requieren el desarrollo de *software* complejo existente hoy en día. Recientemente se observa que este proceso evolutivo tiene un nuevo hito, el de la readaptación. La readaptación puede definirse como el proceso de cambio en la propia organización para dar lugar a una nueva [Alberola *et al.*, 2011]. Normalmente, esta evolución viene motivada por un cambio en el entorno, o la introducción de nuevos requisitos en el dominio del problema. El proceso de readaptación puede incluir roles, objetivos, servicios, normas, etc. El proceso de readaptación según So *et* Durfee incluye las siguientes fases [So *et* Durfee, 1993]:

- **Monitorización.** En esta fase se define el momento y el motivo en el que la organización necesita evolucionar. El modelo de monitorización puede ser centralizado si se realiza por una autoridad o rol específico, o puede ser distribuido si se delega entre un conjunto de agentes. La dos decisiones (cuándo y por qué) pueden ser tomandas según [Dignum *et al.*, 2004] siguiendo un modelo reactivo o proactivo, en función de si existe un proceso de razonamiento.
- **Diseño.** En esta fase se determina el cómo, es decir, los cambios que deben producirse durante la fase de reorganización para dar soporte a los cambios en el entorno o a los requisitos del problema. El diseño de la reorganización puede ser realizado de forma centralizada o descentralizada, en función de donde se lleve a cabo. Así mismo, en función de cómo se decida la reorganización, las decisiones pueden ser tomadas de forma autónoma o por acuerdo.
  El modelo de diseño puede incluir decisiones acerca de [Alberola *et al.*, 2011]: (i) la dinámica de los agentes y su comportamiento, (ii) modelo de interacción y comunicación, (iii) cambios en el modelo social, (iv) evolución del contexto normativo, (v) el soporte a situaciones de emergencia, y finalmente, (vi) los cambios de población en sistemas abiertos.
- **Selección.** Es la fase en el que se elige uno de los diseños especificados en la fase anterior. Esta decisión también puede ser tomada de forma centralizada o distribuida en función de si la decisión es tomada por una autoridad individual de forma autónoma, o mediante acuerdo, de forma distribuida. El proceso de selección puede tener en cuenta los siguientes

criterios [Alberola *et al.*, 2011]: consecución de objetivos, grado de utiidad, coste global y coste individual.

- **Evaluación.** Finalmente, en esta última fase se evalúa el modelo de diseño que se ha seguido, y si se han alcanzado los objetivos pretendidos con la readaptación. Gracias a este modelo de evaluación es posible tomar mejores decisiones en un futuro.

Aunque los modelos de re-adaptación han sido estudiados por diferentes autores como [So *et* Durfee, 1993] [Alberola *et al.*, 2011], en cuanto a la evolución de normas [Carvalho *et al.*, 2006], estructuras [Dignum *et al.*, 2004] [Hoogendoorn *et* Treur, 2009], así como la inclusión de modelos de readaptación en diferentes metodologías MOISE+ [Hübner *et al.*, 2004], e-institutions [Bou *et al.*, 2007], MASE/OMACS/O-MASE [DeLoach *et* Matson, 2004], etc. Hoy en día sigue siendo uno de los campos abiertos en el marco del paradigma de los SMA.

3.4 CONCLUSIONES

Después de analizar el paradigma tecnológico CC en el Capítulo 2, este capítulo ha tenido como principal interés el estudio de la teoría de agentes y los SMA. Este modelo arquitectónico se basa en la *autonomía*, *habilidades sociales*, *reactividad* y *pro-actividad* de sus componentes individuales, denominados agentes [Wooldridge *et* Jennings, 1995]. Su marco ideal de aplicación son los sistemas complejos, abiertos o ubicuos. El entorno tecnológico del paradigma CC se asocia, precisamente, con estos entornos de aplicación debido a su heterogeneidad, dinamicidad e incertidumbre. Por ello, a lo largo del capítulo se ha analizado la relación existente entre ambos paradigmas distribuidos, obteniendo diferentes conclusiones tal y como se detallará a continuación.

Existe una relación incipiente aunque muy prometedora debido a las sinergias encontradas entre ambos sistemas distribuidos. Se determina que la mejor forma de clasificar los trabajos existentes en el estado del arte es mediante la referencia a los roles del paradigma CC especificados en el modelo arquitectónico propuesto por el NIST [Liu *et al.*, 2011].

A partir de esta clasificación, se observa una gran cantidad de trabajos asociados a los roles *Cloud Consumer* debido a que los sistemas CC proporcionan unas capacidades computacionales que permiten a los SMA maximizar sus cualidades intrínsecas. Así mismo, también se observa un abundante número trabajos en los que los SMA actúan como *Cloud Broker*, y en menor medida como *Cloud Auditor*; debido, principalmente, a su similitud con tecnologías precedentes como la búsqueda, composición y monitorización externa de servicios web. Los modelos, técnicas y herramientas propuestas en estos trabajos solucionan principalmente el problema de la la interoperabilidad entre plataformas, siguiente la aproximación de InterCloud [Buyya *et al.*, 2010a] [Sim, 2013].

Sin embargo, pese a la incipiente presencia de los SMA en estos tres roles, se observa un limitado número de trabajos en el marco del rol *Cloud Provider* del paradigma CC. Es decir, aplicaciones de los SMA en el núcleo de una plataforma CC para permitir su control y monitorización mediante IA distribuida. No cabe duda que este limitado número de trabajos está relacionado con la dificultad para gobernar un entorno CC de última generación, así como su complejidad y heterogeneidad inherente. Por ello, más si cabe, la hipótesis de este trabajo de investigación se considera innovadora.

Para que esta aproximación de aplicación de SMA en entornos CC sea posible, es necesario utilizar las últimas innovaciones existentes en este campo, especialmente aquellos modelos de SMA que se diseñan desde una

perspecitiva social [Davidsson, 2001], destacando sobre todos ellos, el modelo de las OV [Rodriguez González, 2010].

Este modelo de diseño de SMA permite dividir las diferentes responsabilidades de un sistema CC entre grupos de agentes con objetivos específicos y modelos de razonamiento adecuados al problema a resolver. No obstante, pese a modelar el sistema como grupos de trabajo con responsabilidades y objetivos independientes, todos trabajan de forma conjunta para conseguir los objetivos globales del sistema. Así mismo, modelar un entorno CC, mediante el uso de OV de agentes inteligentes permite la auto-adaptación del sistema en función de las condiciones del entorno lo que facilita la adaptación dinámica de la plataforma y no sólo la de sus individuos. Así mismo, también permite la inclusión de agentes especializados, con capacidades de razonamiento avanzadas que permiten modelar el sistema utilizando la base que proporciona la IA. Dentro de las metodologías estudias se ha determinado que GORMAS [Argente, 2008] [Argente *et al.*, 2011] es la metodología de diseño que se considera más adecuada al problema a resolver. En este sentido, se considera que tanto las vistas como meta-modelos propuestos en el marco de la trabajo cubren perfectamente todas las fases de diseño y los conceptos necesarios para la descripción del sistema.

Partiendo de la hipótesis inicial y de los trabajos analizados, en el capítulo siguiente se presentará la arquitectura que se ha propuesto, incluyendo tanto el modelo de la arquitectura, como los modelos de razonamiento avanzado que hacen posible su adaptación. Posteriormente, en el Capítulo 5 se evaluará empíricamente a través de un caso de estudio tanto la arquitectura propuesta como los algoritmos de adaptación que se han diseñado en el marco de este trabajo de investigación.

CAPÍTULO 4
MODELO DE ARQUITECTURA

En este capítulo se presenta el modelo arquitectura basado en OV de agentes inteligentes, para la distribución de recursos computacionales en entornos distribuidos, concretamente en sistemas CC. Este modelo arquitectónico se considera el más apropiado para modelar un sistema CC utilizando SMA. Así, el uso de OV permite aplicar en el diseño de un sistema CC características avanzadas derivadas de la IA distribuida, no sólo en tanto en cuanto a la autonomía, pro-actividad, aprendizaje y razonamiento avanzado de sus componentes [Wooldridge *et* Jennings, 1995]; sino que también será posible la incorporación de métodos y modelos para la auto-adaptación dinámica del propio SMA, lo que facilita la adaptación del sistema en su conjunto a los cambios que se puedan producir en el entorno, principalmete, aquellos motivados por la variabilidad en la demanda de los servicios que ofertados por el entorno CC.

Como ya se han indicado en capítulo precedentes, un sistema CC se asocia a un sistema abierto siguiendo un modelo de diseño basado en SMA. Es un sistema abierto por ser dinámico y heterogéneo. En primer lugar, es un sistema dinámico ya que la población de la organización (los agentes) pueden salir o entrar en la citada organización. Además se puede dar el caso, en que agentes externos ofrezcan servicios dentro de la propia organización, jugando roles específicos dentro de la misma [Davidsson *et* Johansson, 2006]. En segundo lugar, es heterogéneo ya que el diseño de los agentes agentes puede llegar a ser muy diverso, teniendo que interacturar con un entorno también heterogéneo donde las diferentes tecnologías subyacentes complican en gran medida las acciones de los agentes individuales. Finalmente, como se detallará a lo largo de este capítulo, en un sistema CC modelado mediante SMA organizativos, también es un sistema con cierta incertidumbre, sobre todo en la toma de decisiones, debido a que los agentes individuales no tienen un conocimiento total del sistema en su conjunto, sino que tan sólo disponen de un modelo de la realidad parcial, centrado en aquellas áreas que tienen a su cargo y, por lo tanto, tienen que recurrir a la coordinación y negociación con sus semejantes para llevar a cabo las tareas que tiene asignadas.

A lo largo de este capítulo también se detallarán los algoritmos propuestos para la auto-adaptación de la organización en función de la demanda de los servicios por parte de los usuarios. Para ello, como se detallará posteriormente, se han diseñado modelos de razonamiento avanzados que se implementan por los agentes individuales que participan en la organización, según el rol que juegen dentro de la misma. Estos agentes serán capaces de

repartir dinámicamente la cantidad de recursos en función de la demanda en los servicios y los contratos de QoS acordados a través de los correspondientes SLA.

El capítulo se organiza tal y como sigue, a continuación en el apartado 1.1 se formaliza el entorno de aplicación del modelo arquitectónico propuesto. Posteriormente, el apartado 4.2 contiene una descripción detallada del SMA propuesto, el cuál se ha diseñado a partir de la metodología GORMAS [Argente, 2008]. A partir de esta arquitectura, el apartado 4.3 detalla el modelo de razonamiento de los agentes especializados en la distribución de recursos, detallando su modelo de representación del entorno y sus algoritmos razonamiento. Finalmente, en el último apartado (4.4) del capítulo se presentan las conclusiones del mismo.

4.1 CARACTERIZACIÓN DEL ENTORNO CLOUD COMPUTING

De forma previa a formalizar la arquitectura multiagente basada en OV que se propone en el marco de este trabajo de tesis doctoral, resulta necesario formalizar el contexto y el entorno en el que se ejecutará dicha arquitectura. Dada la complejidad asociada a un entorno CC, así como los diferentes componentes artificiales y humanos que están involucrados en este contexto, resulta necesario realizar una caracterización y formalización del entorno, que permita acotar el ámbito de aplicación del modelo arquitectónico que se ha propuesto y que se detallará posteriormente en las secciones 4.2 y 4.3. Esta caracterización que se presenta a continuación, puede abstraer a la mayoría de entornos CC que existen en la actualidad en el ámbito científico y en el mercado.

4.1.1 MODELO DE COMERCIALIZACIÓN DE SERVICIOS

En líneas generales, tal y como se ha descrito en los capítulos anteriores, un sistema CC ofrece a los usuarios un conjunto de servicios computacionales de cualquier tipo, es decir, *software*, plataforma e infraestructura. En cierta manera, este modelo no varía en gran medida con respecto a las tecnologías anteriores; sin embargo, la característica clave de este tipo de sistemas es que los servicios se ofertan siguiendo un modelo de pago por uso. Para ello, tomando las bases de la tecnología *Utility Computing* [Ross *et* Westerman, 2004], el usuario contrata un conjunto de servicios computacionales de forma similar a como se contratarían otros servicios de utilidad en la vida real como el gas, el agua, la electricidad, etc. Cada contrato de prestación de servicios que se realiza con los usuarios deberá incluir información acerca de los recursos contratados, el grado de calidad que se asocia a los mismos, el tiempo de duración del contrato y el coste, el cuál vendrá determinado por la cantidad y calidad de los recursos utilizados.

Una vez que se ha establecido el acuerdo, el usuario es libre de utilizar, o no, los servicios contratados. A diferencia de otros servicios de utilidad tradicionales, el tiempo de vida de los acuerdos SLA establecidos suele ser más corto [Venticinque *et al.*, 2010], lo que determina en que el coste pueda variar a lo largo del tiempo, en función de la carga y la demanda de usuarios existente en cada momento [Alhamad *et al.*, 2010a].

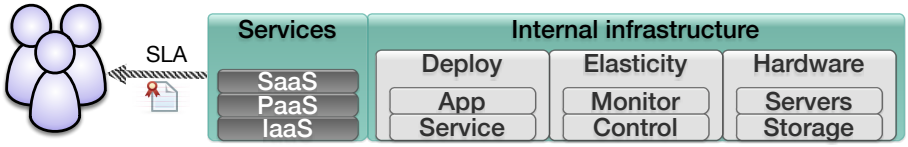

Figura 14.- Modelo de comercialización de servicios en entorno Cloud Computing

Así, de forma general, se ha formalizado el acuerdo de servicio para un usuario cualquiera ($ServA^j$) como la unión del conjunto de acuerdos de uso establecidos sobre cada uno de los servicios de forma individual (SLA_i^j) [Patel *et al.*, 2009] [Emeakaroha *et al.*, 2010]. El acuerdo de uso por servicio concreto ($ID_{i,}$) se determina en función del tiempo de duración del acuerdo (t), las partes implicadas en el mismo (u), el coste asociado (c) y los niveles de calidad acordados (QM_i).

$$ServA^j = \bigcup_{i=0}^{i=m} SLA_i^j \; donde \; SLA_i^j = \{ID_{i,}, u, t, c, QM_i\}$$

A partir de esta sencilla expresión, el reto para conseguir un modelado adecuado del contexto, consiste en determinar la medida de calidad de los servicios que se ofertan. Este proceso de medición es complejo, más si cabe cuándo se ofertan servicios computacionales a diferentes niveles: *software*, plataforma e infraestructura. En el estado del arte es posible encontrar varios trabajos relacionados [Goiri *et al.*, 2010] [Li *et al.*, 2010] [Alhamad *et al.*, 2010a]. Sin embargo, es necesario obtener una medida adecuada al problema a resolver dentro de este trabajo de tesis doctoral, que no es más que la distribución dinámica de recursos computacionales en sistemas CC. Por ello, se ha estudiado el conjunto de métricas existentes, descartando directamente aquellas que no resultaban de interés en el marco de este trabajo de investigación (seguridad, mantenimiento, políticas de internacionalización, gestión del ciclo *software*, etc.) por no estar directamente relacionadas con el problema a resolver.

A continuación se detallarán las métricas relevantes, para ello en primer lugar, se enuncian los productos que se ofertan en un entorno CC, las cuáles se pueden agrupar en tres tipos de productos: (i) *productos hardware*, (ii) *productos software* que siguen la estructura típica cliente-servidor, y finalmente, (iii) *productos combinados* donde se accede a recursos *hardware* mediante el uso de recursos *software*, como por ejemplo, bases de datos o sistemas de almacenamiento con interfaz web.

En primer lugar, se presentan las métricas que permitan medir la calidad de los productos de tipo *hardware*. Principalmente, en este grupo los servicios a tener en cuenta son de almacenamiento, computación y servidores dedicados. En este tipo de servicios es necesario garantizar la disponibilidad de los

mismos, deacuerdo a las características que se hayan acordado a través de un SLA específico:

- Los servicios de almacenamiento según el modelo CC se miden en términos del espacio de almacenamiento (s), número de operaciones (n_r) del tipo añadir, eliminar, recuperar, etc. y el ancho de banda de conexión (BW):

$$SLA_{storage} = \{s, n_r, BW\}$$

- Los servicios de computación y servidores dedicados, en el marco de los sistemas CC, se tienden a enmarcar en la misma categoría, ya que la computación de servicios *grid* se realiza en servidores dedicados en el marco de la tecnología CC [Kibe *et al.*, 2013] [Rings *et al.*, 2009]. Indicar, que en este trabajo de investigación no se contempla la computación de alto rendimiento siguiendo un esquema de paralelización tipo *MapReduce* [Dean *et* Ghemawat, 2008].
  En definitiva, en este tipo de servicios lo que se determina como métricas son las características del servidor dedicado que es contratado por parte del usuario y del cuál, el proveedor debe garantizar su disponibilidad. Así, las principales características son el número de núcleos de computación (n_{cpu}), la cantidad de memoria asignada (s_{ram}), la capacidad del disco duro (s_{hdd}), el ancho de banda (BW) y el sistema operativo ($ssoo$).

$$SLA_{virtual-server} = \{n_{cpu}, s_{ram}, s_{hdd}, BW, ssoo\}$$

Una vez que se han definido las métricas que se utilizan en este trabajo para los servicios *hardware*, a continuación se procede a definir las métricas para los productos de tipo *software*. Las métricas que permitan medir estos servicios resultan más complejas de definir, ya que no sólo hay que asegurar la disponibilidad de los servicios, sino también es necesario evaluar parámetros, que en la mayoría de los casos son dependientes del dominio de uso. Por ejemplo, en productos *software* es necesario evaluar la disponibilidad o no de una funcionalidad específica. Por tanto, una vez más, resulta necesario determinar qué métricas son de adecuadas para los productos *software* teniendo en cuenta el ámbito de esta investigación, la cuál se centra en proporcionar mecanismos que aseguren la disponibilidad y la calidad de los servicios que un usuario haya contratado previamente. Por ello, es necesario establecer un acuerdo de calidad del servicio que atienda a parámetros de calidad directamente relacionados con la forma en que se distribuyen los recursos computacionales subyacentes.

Así pues, en general, las métricas para servicios software están basadas en parámetros como el rendimiento, la confiabilidad, integridad, accesibilidad, robustez, interoperabilidad, seguridad y gestión de errores del servicio [Ladan,

2012]. En la práctica [Goiri *et al.*, 2010], la calidad de los servicios, y por lo tanto los valores de estas métricas están asociados a aspectos relacionados con la ingeniería del *software*, la algoritmia y la infraestructura tecnológica donde se despliegan. Por ejemplo, una correcta arquitectura *software* mejora métricas como la interoperabilidad, la accesibilidad, etc. pero en ningún caso garantiza la eficiencia del *software*. Del mismo modo, unos algoritmos eficientes en la mayoría de los casos permiten incrementar el rendimiento y la robustez del producto *software*, pero no afectan a otras características como la gestión de errores. Por su parte, cuanta más potencia disponga la infraestructura subyacente donde se despliegue la oferta de servicios, mejores valores se obtendrán en cuanto al rendimiento y disponibilidad del *software* o *hardware* ofertado como servicio. Otro problema que existe a la hora de medir los servicios *software* es que, hoy en día, la aplicaciones web modernas y los servicios web son diseñadas siguiendo una arquitectura orientada a servicios [Erl, 2008]. Este modelo es muy flexible y permite que una aplicación pueda dividirse en varios niveles, de forma que cada uno intercambie información con el resto, siguiendo el principio de alta cohesión y bajo acoplamiento [Urgaonkar *et al.*, 2005]. En la Figura 15 se observa un esquema de este tipo de aplicaciones. En este modelo, orientado a la reutilización e integración de componentes puede darse el caso de que haya componentes cuyo rendimiento sea muy alto, pero al mismo tiempo, haya otros componentes cuyo rendimiento no sea tal, lo que degrada el tiempo de respuesta del servicio en su conjunto, desde el punto de vista del usuario final. En conclusión, en este tipo de aplicaciones es necesario que la evaluación de la calidad de los servicios se realice sobre cada elemento atómico, cuya dependencia con terceros no influya en el rendimiento final del usuario.

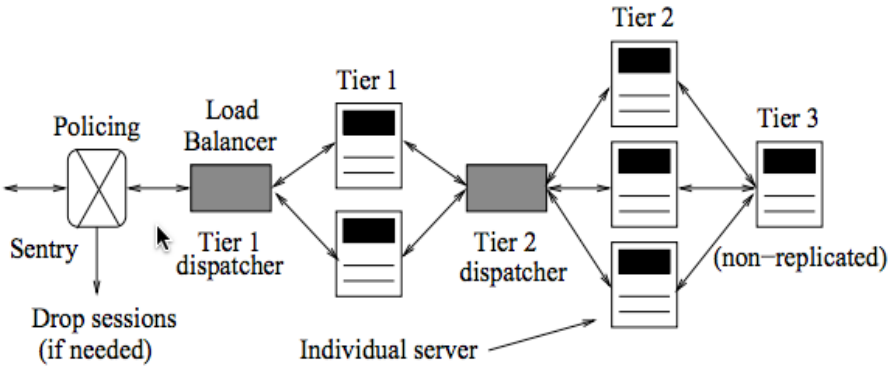

Figura 15.- Varios niveles en una aplicación o servicio web [Urgaonkar et al., 2005]

Así, dentro de este subconjunto de métricas que determinan la calidad de los productos *software* y, que a la vez, están directamente relacionadas con los recursos computacionales subyacentes, también existen diferentes parámetros

a valorar como puede ser el tiempo de respuesta, el número de respuestas en un tiempo determinado, la latencia, etc. De este conjunto de métricas, en el marco de este trabajo de tesis doctoral interesan aquellas que permiten evaluar la calidad del servicio y en este sentido se selecciona el tiempo de respuesta medio de las peticiones. El tiempo de respuesta en las peticiones, viene determinado por el tiempo que tarda una petición en ser atendida. Así, dado un servicio k con un conjunto de métodos (r_i^k) que forman el API del servicio. La calidad de cada una de las peticiones que forman el API del servicio viene determinada por la siguiente expresión de tamaño de la respuesta (s_i^k) y tiempo de la transmisión (t_i^k) de la misma:

$$QoS^k = \{\overline{r_1^k} \cdots \overline{r_i^k} \cdots \overline{r_n^k}\} \; donde \; \overline{r_i^k} = \frac{1}{m}\sum_{i=1}^{m} {s_i^k}/{t_i^k}$$

Junto a la calidad del servicio, también es necesario denotar la disponibilidad (A) del mismo, que en los sistemas computacionales de alta disponibilidad suele ser mayor al 99.5% [Turner IV *et al.*, 2006]. Este nivel de calidad suele venir determinado por el nivel de redundancia de los componentes del entorno *hardware* y de los sistemas de alimentación ininterrumpida. En este sentido, los niveles TIER[35] determinan el grado de redundancia de cada uno de estos componentes. En definitiva, el nivel de calidad de un servicio *software* (SLA_{sw}), se determina en función de su calidad según la métrica que se acaba de definir, el número de peticiones que se pueden realizar (n_r^k), el ancho de banda (BW) y la disponibilidad del mismo (A).

$$SLA_{sw} = \{QoS^k, n_r^k, BW, A\}$$

Finalmente, cuando existen productos computacionales combinados, su caracterización viene determinada por un combinación de las métricas que se acaban de presentar, tal y como se muestra en la siguiente expresión

$$SLA_{sw-hw} = \{SLA_{sw}, SLA_{storage}, SLA_{virtual-server}\}$$

En definitiva, la calidad de los servicios en el marco de este trabajo de investigación viene determinado por la cantidad de recursos computacionales asignados a cada servicio particular. Dada la importancia de la infraestructura en este trabajo, en el siguiente apartado se modelará y caracterizará también la infraestructura subyacente de un entorno CC, que como ya se ha avanzado está compuesta de *hardware* real y *hardware* virtual o abstracto.

[35] ANSI/TIA-942 Telecommunications Infrastructure Standard for Data Centers http://www.tiaonline.org/news-media/press-releases/tia-issues-new-telecommunications-infrastructure-standard-data-center

4.1.2 CARACTERIZACIÓN DEL ENTORNO COMPUTACIONAL

La infraestructura *hardware* de un entorno CC es uno de los sistemas tecnológicos más complejos e impredecibles que existen. En él conviven equipamiento variado como servidores, módulos de red, unidades de alimentación ininterrumpida y un largo etcétera de infraestructura informática de alto rendimiento. El *hardware* existente en estos centros de computación suele ser altamente heterogéneo debido al incesante avance de la electrónica, según dicta la *Ley de Moore* [Schaller, 1997] en el que existe un proceso constante de miniaturización en los componentes, dando lugar a una nueva generación cada 12 ó 18 meses aproximadamente [Barroso, 2005].

Los recursos computacionales a considerar en un entorno CC son principalmente de ejecución y almacenamiento aunque también pueden existir recursos combinados:

- Los **recursos de ejecución** son aquellos que tienen una alta capacidad de procesamiento y memoria para la ejecución de *software*. Aunque también disponen de almacenamiento persistente, éste no es relevante dentro del entorno CC, ya que simplemente es utilizado por el sistema operativo nativo del recurso *hardware*, o el de las máquinas virtuales que puedan contener para la computación de los servicios a los que estén asociados.
  Hoy en día, la administración, monitorización y control de estos recursos puede realizarse de forma dinámica gracias a la tecnología de virtualización [Barham *et al.*, 2003].
- Los **recursos de persistencia** son aquellos que proporcionan al entorno CC capacidad para el almacenamiento persistente y, aunque, también disponen de recursos de procesamiento, éstos no son utilizados como recursos activos en el entorno CC. Su función es la de almacenar de forma duradera y segura la información de los usuarios del entorno CC.
  Los recursos de persistencia suelen agruparse para formar un entorno SAN (*Storage Area Network*) que proporciona un acceso homogéneo a los diferentes componentes de la capa de persistencia [O'connor, 2003] distribuidos en la mayoría de los casos en los diferentes nodos *hardware* (reales o virtuales) del entorno CC.
- Finalmente, también pueden existir **recursos híbridos** que aporten a un entorno CC recursos computacionales y de persistencia. En la práctica este tipo de recursos no suele ser tan habitual.

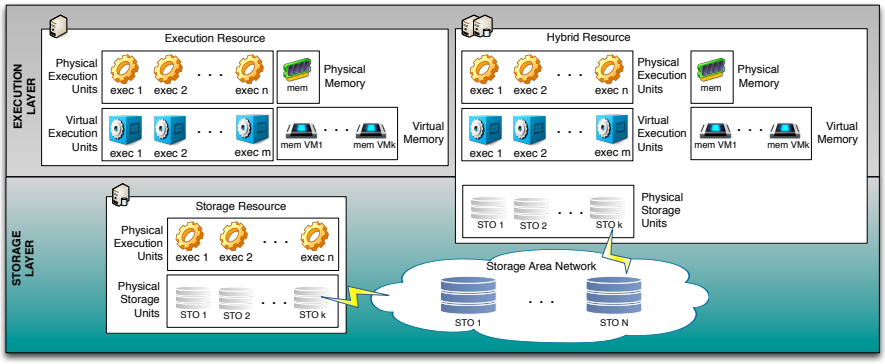

Figura 16.- Recursos computacionales a nivel físico en un entorno Cloud Computing

Dentro de los recursos de un entorno CC también es necesario considerar el equipamiento de red, que hace posible las comunicaciones entre los diferentes elementos del sistema, así como, entre estos con el exterior. Hoy en día, gracias a los sistema de interconexión cableados basados en fibra óptica, este medio de comunicación no constituye un problema a la hora de gestionar la infraestructura. Así mismo, en la actualidad, también es posible virtualizar elementos de red (enrutadores, subredes, balanceadores, etc.) [Niebert *et al.*, 2008]. Sin embargo, dentro del alcance de este trabajo de investigación no se ha considerado esta capacidad debido a la falta de madurez de la tecnología de virtualización subyacente para el control dinámico de este tipo de recursos computacionales [Chowdhury *et* Boutaba, 2009] ya que, en la práctica sigue necesitando la presencia de *hardware* real para que no se limite el rendimiento.

4.1.2.1 EL SISTEMA DE VIRTUALIZACIÓN

La heterogeneidad y complejidad intrínseca a este tipo de entornos computacionales hacen que sean complicados de gestionar y mantener a lo largo del tiempo [Barroso *et* Hözle, 2009]. Por lo que, para ello, se requiere personal altamente especializado. Sin embargo, hoy en día la tecnología de virtualización [Oguchi *et* Yamamoto, 2008] facilita en gran medida esta tarea dentro de este tipo de entornos computacionales. La virtualización permite abstraer la complejidad del *hardware* subyacente en forma de máquinas virtuales. Una máquina virtual es un recurso computacional abstracto que emula una máquina física con unas características *software* y *hardware* concretas.

La gran ventaja de esta tecnología es que la asignación de recursos a cada instancia de ejecución puede reconfigurarse de forma dinámica. Así, es posible, mostrar una vista abstracta o virtual, pero homogénea y controlable del entorno *hardware* real [Barham *et al.*, 2003]. Es decir, a partir del *hardware* existente en un determinado equipo físico (servidor), la virtualización permite repartir y encapsular los recursos físicos entre un conjunto de máquinas

virtuales, siguiendo un esquema similar al modelo *master-slave* [Sullivan *et* Anderson, 1989], en el que un servidor aloja diferentes máquinas virtuales. Así, cada servidor virtualizado, o simplemente máquina virtual, tendrá un conjunto de recursos *hardware* dedicados y será totalmente independiente, siendo posible incluso la ejecución de sistemas operativos diferentes y, por su puesto, *software* totalmente independiente. Esta capacidad que ofrece la tecnología de virtualización permite que los recursos computacionales en la capa de virtualización sean considerados como ilimitados, mientras que en contraposición los recursos físicos o infraestructura *hardware* estén limitados a la infraestructura real existente (número y velocidad de los procesadores, cantidad de memoria RAM, espacio de almacenamiento en disco, elementos de red, etc.).

Los sistemas de virtualización modernos se agrupan en dos categorías [Crosby *et* Brown, 2006], tal y como sigue:

- Los **sistemas de virtualización completa** que emulan un entorno *hardware* entero, de manera que el sistema operativo huésped no tiene conocimiento de estar ejecutándose en una plataforma virtual. Como desventaja, la total encapsulación de las máquinas implica la emulación de una gran cantidad de *hardware* y componentes, lo que reduce en gran medida el rendimiento.
- Los **sistemas de para-virtualización** requieren que el sistema operativo anfitrión y los sistemas huésped sean el mismo o, al menos, compartan ciertas semejanzas. Los sistemas huéspedes deben ser modificados y, habitualmente, ejecutan un *kernel* o núcleo del sistema operativo especial. La pérdida de rendimiento que provoca la necesidad de emulación del *hardware* se reduce en gran medida [Youseff *et al.*, 2006] [Habib, 2008].

Como única desventaja de esta tecnología, se observa que la gestión del entorno virtualizado requiere de un *software* altamente especializado, conocido como *Hypervisor* [Sahoo *et al.*, 2010]. Este módulo es el encargado de gestionar las abstracciones *hardware* en cada nodo físico y para ello consume también recursos [McDougall *et* Anderson, 2010]. No obstante, este consumo computacional depende del modelo de virtualización elegido. Actualmente, este sobrecoste no constituye un gran problema ya que no supera el 2% de la potencia computacional del entorno físico [Che et al., 2008] [Che et al., 2010], aunque para ello requieren *hardware* dedicado, algo que también está ampliamente implantado gracias a las tecnologías como INTEL-VT y AMD-V [Chen et al., 2008].

La virtualización no sólo permite crear esta abstracción en cuanto a la gestión de recursos sino que hace posible una gestión dinámica de los recursos con las siguientes características:

- **Despliegue basado en plantillas.** Las máquinas virtuales pueden ser instanciadas a partir de una serie de plantillas pre-configuradas con los servicios que se desean incluir.
- **Facilidad para la escalabilidad del *hardware*.** El nuevo equipamiento, con independencia de sus características, sólo debe ser configurado inicialmente con el *software* de virtualización asociado. Los servicios a desplegar se añaden a partir de plantillas de máquina virtual.
- **Reasignación de recursos entre máquinas virtuales.** Dentro de un único servidor físico es posible configurar de forma dinámica los recursos asignados a cada máquina virtual, lo que permite realizar un ajuste detallado en cuanto a los recursos computacionales necesarios para la ejecución de un determinado servicio.
- **Migración dinámica de las máquinas virtuales entre servidores.** Además de reubicar los recursos dentro de un único servidor físico, también es posible migrar máquinas virtuales de un servidor a otro, lo que hace posible realizar la distribución de recursos no sólo a nivel micro en cada servidor físico, sino también a nivel macro en todo el entorno CC.
- **Aislamiento entre servicios.** Con frecuencia se busca desplegar cada servicio en máquinas separadas, ya que de este modo se facilita la gestión de la seguridad y se dificulta las interferencias con otros servicios.

Para formalizar un entorno de ejecución basado en virtualización, en primer lugar es necesario caracterizar cada uno de los servidores físicos, es decir, el *hardware* real que alojan las máquinas físicas. Así, cada servidor físico (PR) estará caracterizado por un conjunto de parámetros tal y como sigue:

$$PR = \{hostname, IP, mac, state, M_{max}, vcpu_{max}, M_{min}, vcpu_{min}, benchmark\}$$

Donde existe un conjunto de parámetros, la mayoría de ellos estáticos:

- $hostname$, se corresponde con el nombre del equipo.
- IP, dirección de internet del equipo, habitualmente este tipo de dirección también suele ser estática dentro de la red del sistema CC.
- mac, se refiere a la dirección MAC (*Media Access Control*) del dispositivo de comunicación del servidor, es estática y se necesita para hacer operaciones de arranque del equipo a través del protocolo *Wake on Lan* (WoL) [Liberman, 2006].
- $state$, que determina el estado del servidor ("en ejecución", "reiniciándose", "apagado" y "error").
- M_{max}, se refiere a la memoria física del servidor, en *megabytes* o *kilobytes*.

- $vcpu_{max}$, número máximo de CPUs virtuales del servidor. Normalmente este parámetro no está directamente relacionado con el número de procesadores, o el número de núcleos de cada procesador, ya que suelen intervenir también otros parámetros relacionados con la tecnología de virtualización que determinan el número máximo de procesadores virtuales que se le asignarán a la máquina virtual.
- M_{min}y $vcpu_{min}$, se refiere a la asignación mínima de memoria y CPU, respectivamente, que debe estar disponible para el servidor físico (anfitrión) y que en ningún caso podrá ser asignado a las instancias virtuales en ejecución. Estos recursos se utilizan para el correcto funcionamiento del software de control del equipo físico y el *software* de gestión de la capa de virtualización.
- $benchmark$, que es una medida del rendimiento de la máquina física, con respecto a otros equipos que formen parte del sistema CC.

Por su parte, cada instancia virtual (VM) en ejecución vendrá determinada por un segundo vector con los recursos virtuales que tiene asignados en cada momento.

$$VM = \{IP, state, M, vcpu, M_i, \overline{p_{cpu}}\}$$

Donde existe un conjunto de parámetros, la mayoría de ellos dinámicos:

- IP , será la dirección IP (*Internet Protocol*) asignada al servidor virtualizado.
- $state$, que determina el estado del servidor ("sin estado", "ejecutándose", "reiniciándose", "apagado" y "error").
- M y $vcpu$, se refiere a la memoria y número de procesadores virtuales que tiene asignados el servicio en cada momento.
- M_i, se refiere a la media de memoria utilizada por la instancia virtual en un periodo de tiempo.
- $\overline{p_{cpu}}$, que determina si el uso de la CPU en un periodo de tiempo determinado ha sido ascendente o descendente, se determina a través del análisis estadístico de la regresión lineal de los valores tomados a lo largo del periodo de tiempo.

Cada servicio se construye a partir de una o varias máquinas virtuales, por lo que las características intrínsecas del servicio vienen determinadas por la plantilla de la máquina virtual que ofrece el servicio. En tiempo de ejecución, a partir de esta plantilla será posible instanciar y modificar los recursos asociados a un determinado servicio.Así cada plantilla de nodo virtualizado asociado a un servicio cualquier k (VM_t^k), se describirá a través de un conjunto de propiedades tal y como sigue:

$$VM_t^k = \{ID^k, M_{min}, vcpu_{min}, type, state\}$$

Donde:

- ID^k, se refiere al identificador del servicio correspondiente.
- M_{min}, se refiere a la memoria mínima que tiene que estar asignada a una instancia en ejecución de la máquina virtual.
- $vcpu_{min}$, se refiere al número de procesadores virtuales que tiene que tener asignada una instancia en ejecución de la máquina virtual, como mínimo para que el servicio funcione correctamente.
- $type$, que determina el tipo de servicio al que está asociado, es decir, si es un producto *hardware* o *software*. Este parámetro lo que determina, por ende, es si se pueden modificar los recursos asignados a cada instancia virtual en tiempo de ejecución.
- $state$, que determina si el servicio es balanceable, es decir, si aloja una aplicación con o sin estado. Más tarde en el apartado 4.1.3 se explicará la diferencia.

Cada servidor físico será el anfitrión de un conjunto de máquinas virtuales. Por lo tanto, en cualquier momento, cada máquina física del sistema tendrá asociada una matriz con información relativa a la máquina física, así como a las diferentes máquinas virtuales que aloja en un determinado momento. Esta matriz es una instantánea de cada servidor físico en ejecución, a través del cuál se puede determinar los servicios a los que presta recursos y la cantidad de recursos que presenta a cada uno.

$$exec^e = \begin{Bmatrix} & PR^e & \\ VM_1 & \dots & VR_m \end{Bmatrix}$$

4.1.2.2 EL SISTEMA DE PERSISTENCIA

Los sistemas de ficheros tradicionales no pueden cumplir con los requerimientos de los nuevos y grandes sistemas de almacenamiento, que incorporan grandes cantidades de datos. Estos nuevos tipos de sistemas precisan de nuevos mecanismos de almacenamiento que permitan características como la tolerancia a fallos, la alta disponibilidad, la escalabilidad del espacio de almacenamiento y la velocidad de acceso [Clifford *et al.*, 2007]. Para conseguir estas capacidades es necesario diseñar un sistema de ficheros distribuido que permita la replicación y la partición de datos. La replicación de datos, no sólo en diferentes discos duros, sino también en diferentes nodos *hardware*. Gracias a lo cuál es posible proporcionar un acceso a la información más rápido, que el servicio no se interrumpa por fallos en un nodo y, finalmente, que no se pierda información en el caso de que exista algún fallo en alguno de los componentes. La partición de datos aumenta la velocidad de lectura y escritura de información, y permite aumentar la capacidad del sistema mediante la adicción de nuevos nodos *hardware*.

El espacio de almacenamiento del sistema de ficheros distribuido, tal como se aprecia en la Figura 17, se utiliza habitualmente para tres grandes necesidades de almacenamiento en los sistemas CC [Grossman *et al.*, 2009]: (i) la persistencia de los datos de usuarios y aplicaciones, (ii) los discos duros virtuales de la máquinas virtuales en ejecución y, finalmente, (iii) para el almacenamiento de las plantillas de los servicios.

Hoy en día, los mecanismos de gestión de sistemas SAN permiten una gran cantidad de acciones sobre el entorno de persistencia, que van desde la modificación dinámica de los diferentes espacios de almacenamiento y la asignación de cuotas, hasta la configuración del número de volúmenes distribuidos existente, pasando por la configuración de algoritmos de sincronización e intercambio de información [Elkington *et al.*, 2005].

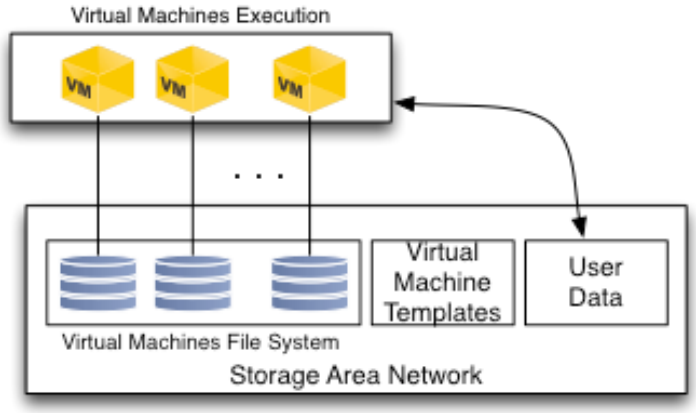

Figura 17.- Despliegue de las máquinas virtuales en el entorno *hardware*

A diferencia de lo que sucedía con los entornos de virtualización, la readaptación de este tipo de sistemas no puede ser tan dinámica debido a las restricciones tecnológicas existentes. Por ejemplo, la modificación de las diferentes réplicas implica la sincronización de datos cada vez que se crease una nueva réplica, lo que redundaría en una pérdida de rendimiento y un incremento de la actividad de red. Así pues, en el marco de este trabajo se limita su aplicación al establecimiento de cuotas de uso por usuario.

4.1.3 MODELO DE OFERTA DE SERVICIOS

La combinación de las capas que se acaban de presentar (ejecución y persistencia) permite abstraer los recursos haciendo posible una gestión dinámica de los mismos lo que, por ende, hace posible ofrecer un salto cualitativo y cuantitativo a la hora de ofertar servicios computacionales. En este sentido, siguiendo un modelo CC como se presenta en la Figura 18, cada servicio *software* de la plataforma, a nivel PaaS o SaaS, puede estar

desplegado de forma simultánea en varias máquinas virtuales (nodos, o *workers*). Gracias a esta capacidad es posible configurar de forma elástica los recursos asignados a cada servicio. La demanda en términos de peticiones de un determinado servicio es balanceada entre las diferentes máquinas virtuales que están asociadas al servicio. Además, de forma dinámica puede variar en tiempo de ejecución el peso que cada nodo virtual tiene en el balanceo. Así, la elasticidad se basa en modificar los recursos (virtuales) asignados a cada servicio de forma dinámica en función de la demanda, siguiendo las siguientes estrategias:

- Variar los recursos computacionales asociados a una determinada máquina virtual (procesamiento y memoria).
- Modificar el número de máquinas virtuales asociadas a un determinado servicio, incrementando o decrementando por consiguiente la cantidad de recursos que se asignan a un determinado servicio.
- Incluso es posible variar el servidor físico en el que se hospeda una determinada máquina virtual. Este proceso se conoce como migración y se puede realizar sin la necesidad de detener la ejecución del máquina virtual que ofrece el servicio, pero para ello tiene el inconveniente de sobre-cargar la red durante el proceso de migración.

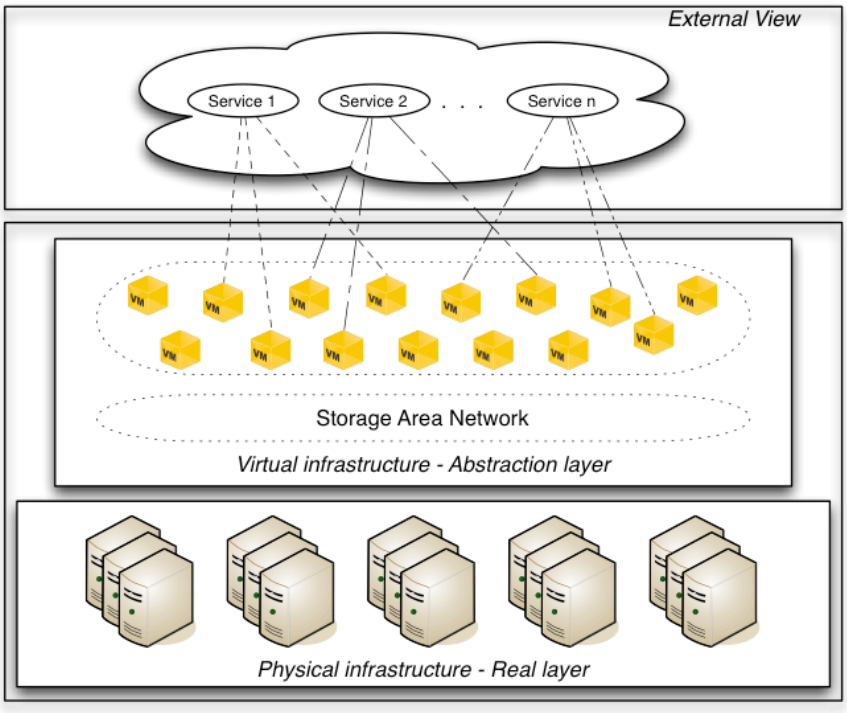

Figura 18.- Modelo de despliegue en CC

Cada nodo trabajador asociado a un servicio concreto tiene unas características particulares que dependen del tipo de servicio y que se almacenan en la plantilla de la máquina virtual que proporciona el servicio, como ya se detalló en el apartado 4.1.2.1. En este sentido, estos nodos pueden tener estado o no, tal y como sigue:

- **Stateless**. Son aquellos servicios que no tienen estado y que por lo tanto pueden existir diferentes nodos atendiendo peticiones de forma simultánea, lo que hace posible su elasticidad. El estado de estas máquinas reside en la capa de persistencia del entorno CC.
  Los nodos de tipo *stateless* son aquellos que de forma conjunta ofertan un servicio concreto de la plataforma. No sólo se tiene que garantizar la disponibilidad del servicio, sino también la calidad del mismo, para lo que es necesario distribuir sus recursos de forma dinámica en función de la demanda haciendo uso de todas las posibilidades que proporciona la tecnología de virtualización.
- **Statefull**. S son aquellos servicios que tienen estado y, por lo tanto, no pueden ser replicados de la misma forma que si no tuvieran estado, ya que obligatoriamente también se tendría que replicar el estado.
  Los servicios *statefull*, suelen asociarse a productos de infraestructura *hardware* donde los usuarios contratan servicios concretos de infraestructura con unas características *hardware* (virtual) determinadas. En este sentido, sólo es necesario garantizar su disponibilidad. También pueden asociarse a productos *software* en los que la persistencia de información no es realizada en una base de datos de tipo CC y, por lo tanto, la información reside en cada nodo individual lo que impide su replicación debido a que no es posible la sincronización efectiva de los discos duros virtuales asociados las copias redundantes. En este caso, la cantidad de recursos asociados al servicio sólo puede alterarse mediante la modificación de los recursos del único nodo en ejecución.

4.2 FORMALIZACIÓN DE LA ARQUITECTURA MULTIAGENTE

El diseño de un sistema de monitorización y control de un entorno tecnológico como el que se acaba de presentar en la sección 1.1 requiere del uso de técnicas de IA para poder incorporar tareas que hagan posible la adaptación dinámica a los cambios y alteraciones en la demanda de los servicios que se oferta. La adaptación dinámica a cambios que ocurren en el entorno requiere de capacidades de aprendizaje, representación distribuida de conocimiento y razonamiento avanzado. En este sentido, se seleccionan los SMA y especialmente aquellos basados en OV para dar soporte al diseño de un *software* de control que sea robusto y fiable. Dichos SMA basados en organizaciones disponen de teorías, modelos, mecanismos, métodos y herramientas que permiten desarrollar sistemas con capacidad de reorganización y que pueden adaptarse de esta forma a futuros cambios en su entorno [Rodriguez González, 2010]. Un diseño basado en OV permite el diseño mediante el uso de aspectos organizativos y sociales, lo que supone una representación más cercana al comportamiento humano. Además, gracias a este modelo basado en este tipo de arquitecturas distribuidas es posible no sólo repartir las tareas de control y monitorización a lo largo de todo el entorno, sino que también es posible el diseño, y posterior implementación, de agentes especializados que tomen decisiones autónomas a partir de un modelo de representación del entorno y un sistema de razonamiento que puede ser más o menos complejo.

Este modelo de SMA basado en OV también permite que agentes externos realicen servicios dentro de la organización lo que facilita la incorporación de nuevas funcionalidades, que no están directamente desarrolladas por el sistema. En este sentido, en los últimos años, los investigadores han llevado a cabo diversos estudios para ofrecer procedimientos y metodologías que permitan diseñar SMA abiertos que admitan la entrada y salida dinámica de sus entidades. Dentro de estos nuevos modelos tienen cabida agentes heterogéneos, con arquitecturas e incluso lenguajes distintos [Zambonelli *et al.*, 2003]. Este tipo de modelos de diseño detallan la estructura del SMA términos de roles y la participación de los agentes en los roles del sistema, junto con normas, protocolos de interacción y objetivos asociados a las tareas. La utilización de este tipo de modelos en entornos CC supone un valor añadido ya que proporciona grandes capacidades de aprendizaje y adaptación. Por otro lado, incorporan aspectos organizativos en la gestión de entornos CC, lo que permite diseñar sistemas abiertos, basados en aspectos sociales, con un comportamiento más cercano a las organizaciones humanas, y que por lo tanto, facilitan la interacción con los humanos.

La arquitectura propuesta en el marco de este trabajo de tesis doctoral se denominada **+Cloud** (*Multiagent System Cloud*) y está basada en OV de agentes inteligentes, gracias a lo cuál es posible hacer frente a las necesidades de adaptación, cambio y entrada y salida de componentes que requieren las plataformas de tipo CC. El principal objetivo de +Cloud consiste en la monitorización de un entorno CC y el control del mismo para que pueda adaptarse de forma automática y dinámica a las necesidades existentes en cada momento. +Cloud tomará datos a lo largo del entorno CC en su conjunto, incluyendo tanto la infraestructura subyacente, como la demanda de los servicios que se proporcionan. Así, gracias a este modelo de monitorización distribuida es posible adaptar los recursos existentes en el entorno CC en función de la demanda de cada servicio existente en cada momento, haciendo posible satisfacer el doble objetivo de cumplir con los acuerdos SLA establecidos y la reducción del consumo energético. +Cloud propone una arquitectura innovadora para la gestión de sistemas CC, proporcionando una nueva visión basada en aspectos organizativos. Entre los aspectos innovadores de +Cloud cabe destacar el diseño de agentes con capacidades avanzadas de razonamiento, las cuáles se detallarán en el apartado 4.3, para la gestión elástica de entornos CC.

A lo largo de este apartado, se presenta la arquitectura +Cloud en su conjunto. Para modelar el innovador SMA que se propone se ha utilizado la metodología de diseño GORMAS (*Guidelines for Organization-based MultiAgent Systems*) [Argente, 2008] que extiende los modelos propuestos en la metodología ANEMONA [Botti *et* Giret, 2008], que a su vez son una extensión de INGENIAS [Pavón *et* Gómez-Sanz, 2003]. Esta metodología se basa en seis meta-modelos (agente, actividad, interacción, entorno, organización y normativo) gracias a los cuáles es posible describir cualquier SMA organizativo a través de cuatro vistas (estructural, funcional, social y dinámica) que permiten especificar de forma detallada cualquier sistema *software* organizativo. Para facilitar esta tarea, la propia metodología propone una secuencia-guía para definir procedimentalmente el análisis y diseño del SMA.

Esta guía metodológica parte de las guías de diseño de organizaciones humanas [Moreno-Luzon *et al.*, 2001] [Mintzberg, 1993] [Galbraith, 1977] y se divide en tres etapas principales: el análisis, diseño del sistema y diseño de la dinámica. Etapas que a su vez se llevan a cabo en las siguientes ocho fases:

- **Fase A. Misión.** Se realiza un análisis de la motivación que se persigue al definir la organización o sistema, es decir, el porqué de la existencia de dicha organización o para qué se crea. Así como los resultados que, en conjunto, se esperan conseguir. También se determina el entorno en el SMA existe, detallando los productos y/o servicios a ofrecer, cuáles son los grupos de interés y su localización.

- **Fase B. Tareas y procesos.** Se analizan con mayor detalle los servicios a ofrecer en el sistema, sus requisitos y los procesos que conllevan. Se detallan también las tareas y objetivos asociados a dichos servicios.
- **Fase C. Dimensiones organizativas.** Donde se analizan las dimensiones de la organización (departamentalización, especialización, sistema decisor, formalización, coordinación), que imponen ciertos requisitos sobre los tipos de trabajo, así como la diversidad e interdependencia de las tareas a realizar.
- **Fase D. Estructura organizativa.** Se determina y selecciona la estructura organizativa más adecuada para la organización, en función de sus dimensiones. Se hace uso de modelos organizativos para especificar los roles, interacciones y normas relacionados con la propia estructura.
- **Fase E. Procesos de información-decisión.** Para cada servicio identificado, se detallan las interacciones (flujos de información y de toma de decisiones) necesarias para llevar a cabo el servicio. Además, se definen los contratos de calidad de servicio a los que se comprometen los proveedores y consumidores cuando éste sea llevado a cabo.
- **Fase F. Dinamicidad del sistema abierto.** Se establece la funcionalidad ofrecida como sistema abierto, que incluye los servicios que se deben publicitar y las políticas de adquisición y liberación de roles. Además, se diseñan los agentes propios del sistema.
- **Fase G. Sistemas de medición, evaluación y control.** Se cuantifican o evalúan las tareas y actividades y se establecen mecanismos para determinar si los objetivos del sistema se cumplen. Así mismo, se revisan las normas de la organización para especificar quiénes se encargan de ellas y las supervisan.
- **Fase H. Sistemas de recompensas.** Se determina el sistema de incentivos, para recompensar a los miembros que avancen en dirección de los intereses de la organización. También se analizan los sistemas de sanción para aquellos miembros que no cumplan con las normas dadas.

Según la metodología, las fases A y B se corresponden con el análisis de la arquitectura, mientras que el diseño se corresponde las fases C y D. Finalmente, el diseño de la dinámica de la arquitectura se realiza en las fases finales (E, F, G y H). En el Anexo A se puede consultar la documentación completa (análisis, diseño y dinámica del SMA), en el que se detalla de forma clara y razonada cuál es la estructura interna, dinámica y las normas que rigen su comportamiento dentro del sistema +Cloud.

A continuación, en las siguientes secciones, se realizará una descripción de los principales componentes de la arquitectura propuesta, identificando los aspectos innovadores de cada uno de ellos.

4.2.1 IDENTIFICACION DE LOS COMPONENTES DE LA ARQUITECTURA

La arquitectura que se propone se basa en aspectos organizativos y, por tanto, es necesario identificar la estructura organizativa a utilizar. Para ello, el primer paso ha consistido en la identificación de sus componentes, lo que permite establecer el modelo de interacción partiendo del análisis de las necesidades de los usuarios potenciales del sistema. Posteriormente, a partir de este análisis ha sido posible deducir los roles de los usuarios y componentes que participan en el sistema y la forma en la que van a intercambiar información.

En este sentido, el desarrollo de un sistema de monitorización y gobierno de un entorno CC siguiendo un modelo de diseño basado en SMA difiere respecto a los modelos tradicionales de control en este tipo de plataformas, donde habitualmente la toma de decisiones se realiza de forma centralizada [Buyya *et al.*, 2010b]. En el marco de esta tesis doctoral se sigue un modelo alternativo, basado en la teoría de agentes y los SMA. En este sentido, se han distribuido las responsabilidades, principalmente de monitorización y toma de decisiones, a lo largo de todos los componentes de la plataforma. Gracias a este modelo es posible que la toma de decisiones se realice allí donde se recoge la información, sobre la base que proporciona el conocimiento local, lo que ha permitido el diseño de procesos de control ágiles basados en información con incertidumbre y la interacción entre semejantes. Esta peculiaridad provoca, incluso, que a medida que el sistema se adapta a la demanda siguiendo el principio de elasticidad de los sistemas CC, parte de los agentes entren y salgan del sistema en función del ciclo de vida de los componentes físicos donde se sitúan. En la Figura 19 se puede apreciar como cada uno de los agentes/roles que participan en la organización están situados a lo largo de todo el entorno computacional.

Siguiendo este modelo de distribución que se indica, en cada servidor físico del entorno CC existe un agente encargado de la monitorización, y otro del control a nivel local, entre ambos tienen la autoridad para gobernar totalmente el servidor físico donde se sitúan, lo que a su vez implica el reparto de recursos en las máquinas virtuales que aloja. Pero, cuando se tenga que realizar una distribución de recursos que conlleve la asignación o retirada de nodos a un determinado servicio entonces, es otro agente especializado y que también está situado en cada uno de los nodos físicos de la infraestructura, el encargado de la toma de este tipo de decisiones que implican a más de un nodo físico de la plataforma CC.

Siguiendo un modelo similar, a cada servicio ofertado a los usuarios se le asocian dos agentes, uno para la monitorización y otro para el control que son los encargados de asegurar el cumplimiento de los acuerdos SLA previamente establecidos. Físicamente están situados en el nodo que realiza el balanceo de

carga entre los diferentes nodos trabajadores lo cuál les permite disponer de información precisa para la correcta toma de decisiones en este nivel. En este sentido, las tareas a este nivel están relacionadas con el balanceo de la carga entre los diferentes nodos, la detección de errores y, principalmente, la monitorización de los parámetros de calidad del servicio.

Finalmente, existen otros agentes situados en el punto de entrada del sistema CC con tareas muy dispares. En primer lugar, dos agentes de control, el primer de ellos que se encarga de controlar la infraestructura *hardware*, su estado, así como el arranque o parada de los nodos físicos en función de la demanda. Por otro lado, otro agente supervisor que es un controlador global que verifica que el resto de componentes y agentes funcionan correctamente deacuerdo a su especificación. Finalmente, también existe un agente encargado de establecer acuerdos de servicio con los usuarios de la plataforma.

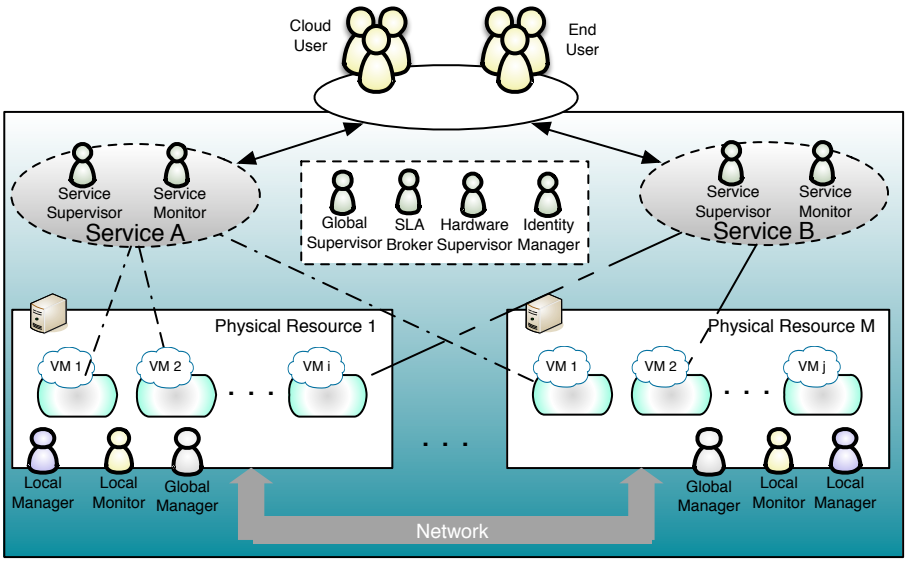

Figura 19.- Distribución roles en la infraestructura

A continuación se describen en detalle los roles que se han identificado después de análisis de la interacción. Cada uno de los roles tiene unas funcionalidades que son capaces de ejecutar y unas responsabilidades dentro de la arquitectura, así como capacidades de interacción con otros roles. Cada rol normalmente será instanciado y ejecutado por un agente individual y dedicado, aunque es posible que un agente pueda ejecutar varios roles.

En primer lugar se comenzará por los roles que están directamente relacionados con los usuarios que hacen uso del sistema. Estos agentes

humanos tienen asociados agentes inteligentes virtuales, que facilitan su interacción con el resto de la arquitectura multiagente:

- **Cloud User.** Representa a usuarios físicos, es decir, personas que utilizan el sistema. Este rol se asocia al tipo de usuario *Cloud user* según el modelo de arquitectura descrito por el NIST [Liu *et al.*, 2011], ya descrito en el Capítulo 2. En este trabajo, dentro de este tipo de usuarios se han definido dos subgrupos:
  - *Desarrollador.* Es un usuario externo que tiene la capacidad de introducir nuevas aplicaciones en el sistema, principalmente aplicaciones en la capa SaaS. Por otro lado, también puede consumir servicios y ofrecer funcionalidades a los usuarios finales.
  - *Manager.* Este usuario puede administrar aplicaciones de la capa SaaS dentro del sistema, configurando parámetros sobre las mismas y negociando con el sistema acerca del uso de los servicios que ofrece la plataforma CC.
- **End User.** Representa al usuario final del sistema, siendo el verdadero consumidor de los servicios y, por lo tanto, también de los recursos de la plataforma a través del uso de las aplicaciones que están desplegadas en el entorno CC. Destacar que estas aplicaciones pueden ser tanto las aplicaciones nativas del propio entorno CC, como las aplicaciones desplegadas por terceros, es decir, usuarios de tipo *Cloud User*, anteriormente descrito.
- **Identity Manager.** Este rol, que no se asocia a una entidad humana, sino a un ente artificial, se encarga de gestionar la entrada y salida de usuarios en la unidad organizativa que representa el sistema. Para ello, dispone de un repositorio con los usuarios del sistema y encapsula los diferentes procedimientos de autenticación que disponga la plataforma. Su labor consiste en iniciar el procedimiento de autenticación cuando otros usuarios (*End User, Cloud User* o *Cloud Admin*) traten de acceder al sistema, de este modo es posible conocer cuál es su tipo de usuario, el nivel de privilegios y el conjunto de acuerdos alcanzados de servicios del sistema en el caso de que existan.

Una vez que se han presentado los agentes humanos y los roles encargados de su entrada en el sistema. A continuación se prestará atención a los roles encargados de la gestión de la infraestructura subyacente, que como ya se ha indicado se corresponde con una de las características clave de este tipo de plataformas. Así pues, se identifican los siguientes roles directamente relacionados con la monitorización y control del *hardware*:

- **Local Monitor.** Es el encargado de recoger datos acerca del estado de los recursos locales de cada servidor físico, incluyendo tanto a la propia máquina física, como a las diferentes máquinas virtuales que pueda hospedar. Su principal objetivo es recoger del entorno la información que otros componentes de la plataforma CC puedan necesitar, realizando el tratamiento previo de estos datos hasta convertirlos en información relevante para la toma de decisiones.

  Para realizar esta monitorización debe existir al menos un agente asociado a este rol en cada servidor físico, el cuál debe tener un conocimiento detallado de cada nodo individual, pero al mismo tiempo tiene un desconocimiento total acerca del resto de nodos de la plataforma CC.

  Dado que existe la posibilidad de que se paren y arranquen nodos de ejecución, entonces también existe la posibilidad de agentes de este tipo entren y salgan del sistema de forma dinámica, ligando su ciclo de vida al del servidor que monitorizan.
- **Local Manager.** Encargado del control y distribución de los recursos computacionales de la máquina física, pudiendo redistribuir recursos entre las instancias en ejecución, lanzar o apagar máquinas virtuales a partir de plantillas de servicio previamente configuradas.

  Su objetivo es, por tanto, la gestión eficiente de los recursos del servidor físico entre los diferentes nodos que hospeda, maximizando su uso, pero sin disminuir la calidad de los servicios que se están proporcionando; y, garantizando que la máquina física siempre dispone de los recursos mínimos indispensables para poder realizar las tareas de coordinación y control, así como para realizar el procesamiento necesario de monitorización.

  Al igual que sucedía con el rol Local Monitor, este rol debe ser adquiridos por un agente en cada uno de los servidores físicos del entorno CC, el cuál trabaja en estrecha colaboración con el agente anterior, es decir el agente *Local Monitor* del mismo equipo, para realizar una monitorización completa y un control y adaptación automática de cada servicio individual. Por lo tanto, ambos roles tiene un conocimiento y control total sobre el estado y reparto de recursos en cada equipo individual.
- **Global Manager.** Es el rol encargado de la toma de decisiones acerca de cómo distribuir los recursos computacionales entre varios nodos de la plataforma CC. El objetivo es la distribución de recursos a nivel global, y no sólo a nivel local como sucedía con el rol *Local Manager*. Como ayuda a la toma decisiones acerca de cómo distribuir los recursos, utilizan una base de conocimiento parcial (proporcionada por el rol *Local Monitor*) y

las experiencias pasadas almacenadas en el repositorio de histórico de uso.

Son los encargados de determinar cómo y con qué características se instanciarán nuevas máquinas virtuales que se asocien a servicios. Así como el proceso de compactación de las máquinas virtuales existentes en un número menor de servidores, para que de este modo sea posible apagar servidores físicos y reducir el consumo energético.

- ***Hardware* Manager.** Es el rol encargado de la gestión de los nodos *hardware* existentes en el entorno CC. Tiene acceso un repositorio con los nodos existentes y el estado de cada uno, siguiendo la caracterización de cada nodo, la cuál ya se ha presentado en el apartado 4.1.2.1.

  Sus tareas son la gestión del estado de cada nodo del sistema, manteniendo el repositorio actualizado, así como el arranque y parada de los equipos físicos según determinen los algoritmos de elasticidad desarrollados.

Otro de los pilares de un entorno CC es la capacidad de ofertar servicios computacionales. Como ya se ha detallado en el apartado 1.1, la gran diferencia de un entorno CC respecto a otras tecnologías anteriores es la capacidad de ofertar los servicios bajo demanda siguiendo un modelo de pago por uso [Armbrust *et al.*, 2010]. Para lo cuál es necesario establecer un modelo de acuerdo de uso en los servicios a través de SLAs específicos. Posteriormente, una vez se ha establecido este acuerdo, la plataforma debe realizar una monitorización precisa del servicio que permita asegurar que se están cumpliendo los niveles de calidad acordados, o por el contrario es necesario tomar medidas que aseguren su cumplimiento. Así en este sentido, se observan los siguientes roles implicados en esta tarea:

- **Service Monitor.** Es el rol encargo de monitorizar cada uno de los servicios ofertados por el sistema. Para ello recoge información acerca las peticiones que estén realizando los usuarios, midiendo parámetros de calidad sobre las mismas, rendimiento, errores, etc. Al igual que sucedía con el rol *Local Monitor*, deberá pre-procesar la información recibida para que pueda estar disponible en tiempo y forma para el proceso de toma de decisiones. Así mismo, este agente también dispone de acceso a un repositorio histórico donde guarda la información relativa a la demanda histórica del servicio que monitoriza.

  Físicamente se sitúa en el nodo que balancea las peticiones entre las diferentes instancias que atienden el servicio, de forma que pueda monitorizar toda la información referente al intercambio de peticiones.

Por lo que existe un agente asociado a este rol por cada uno de los servicios que oferta la plataforma.

- **Service Supervisor.** Es el encargado de tomar decisiones acerca de cada servicio individual, tomando como base la información que le proporciona el rol *Service Monitor*.

  Su principal labor es la de asegurar los acuerdos acerca del uso del servicio que controla, tomando las acciones adecuadas en el caso de que no se estén cumpliendo los acuerdos SLA previamente establecidos. Las acciones posibles cuando se detecta una pérdida del rendimiento que puede incurrir en una violación de los acuerdos establecidos pasan por la solicitud de recursos adicionales al rol *Local Manager* (primero) y *Global Manager* (después). Finalmente, también se asegura de que al menos existen un número determinado de nodos trabajando en equipos físicos independientes, lo que permite asegurar de este modo la disponibilidad del servicio monitorizado.
- **SLA Broker.** Representa al conjunto de entidades encargadas de establecer acuerdos de uso entre la plataforma CC y los usuarios externos que pretenden hacer uso de los servicios ofertados. Esta tarea es compleja ya que implica un proceso de negociación con los usuarios finales, principalmente, con el rol *Cloud User*; así como la recogida de información a lo largo de todo el entorno CC para establecer un modelo de coste de los servicios en función de su calidad y la aceptación o rechazo de nuevos usuarios y/o acuerdos SLA.

  En este sentido, este rol deberá recoger información de los siguientes componentes de la plataforma:
  - Interacciona con el rol *Service Monitor* para recuperar información acerca del uso y calidad de los servicios, lo que permite elaborar un modelo de coste en función de la calidad existente.
  - Interacciona con los roles *Hardware Manager* y *Global Manager* para obtener información sobre el grado de utilización de la infraestructura que permita conocer si existen recursos disponibles para asumir el establecimiento de nuevos acuerdos a largo plazo.
  - Finalmente, interacciona con el *repositorio de histórico de uso* para obtener información relativa al consumo medio del conjunto de servicios en el pasado, de forma que esta variable también influya en el modelo de coste y previsión de acuerdos SLA

  A partir de esta información y del modelo de coste establecido es posible realizar acuerdos con los usuarios de la plataforma a través de la negociación en cuanto al uso y calidad de servicios. En este sentido, es necesario reseñar que esta faceta del paradigma CC se escapa de este trabajo de investigación. No obstante, en el estado del arte se observan

una gran variedad de técnicas y algoritmos, algunos de ellos basados en SMA, con la suficiente madurez en cuanto a su desarrollo para establecer acuerdos con los usuarios [An *et al.*, 2010] [Alhamad *et al.*, 2010a] [Son *et* Sim, 2012] [Sim, 2010] [Venticinque *et al.*, 2011].

Finalmente, una vez que se han presentado los roles que fundamentalmente hacen posible gobernar un entorno CC, a continuación se presenta el último rol que se encarga de la supervisión y control de todo el sistema:

- **Global Supervisor.** Este rol se encarga de supervisar que el resto de roles y componentes de la plataforma funcionan correctamente según sus especificaciones. En el caso de que algo falle, o alguno de los agentes no responda a sus mensajes, es el encargado de tomar las acciones que sean necesarias para devolver el sistema a un estado de funcionamiento óptimo.
  Destacar que este rol puede ser adquirido por un agente que interaccione con un supervisor humano que represente al administrador sistema y cuya labor es la de monitorizar el correcto funcionamiento del mismo. Para ello tiene acceso a los datos de monitorización del sistema, así como a las herramientas que permiten reconfigurar la redistribución de los recursos de forma supervisada.

A partir de esta identificación de roles que participan en el sistema, es posible diseñar la organización de forma unificada, intuitiva y con un alto nivel de abstracción [Agüero *et al.*, 2009] [Agüero *et al.*, 2010]. Mediante esta aproximación se facilita y se simplifica el proceso de diseño de SMA orientado a organizaciones obteniendo modelos de OV que pueden ser implementados en diferentes plataformas. En el siguiente subapartado se presenta en detalle el diseño de la arquitectura que se propone en el marco de este trabajo de investigación.

4.2.2 DISEÑO DE LA ARQUITECTURA

+Cloud es un SMA basado en organizaciones para cuyo diseño se han seguido las pautas propuestas por la guía metodológica GORMAS [Argente, 2008] desarrollada en el marco de los proyectos de investigación THOMAS[36] [Carrascosa *et al.*, 2009] [Giret *et al.*, 2010], OVAMAH[37] [De la Prieta *et al.*, 2009] e iHAS[38]. Esta metodología permite diseñar una organización de forma unificada, intuitiva y con un alto nivel de abstracción. Mediante esta

[36] http://thomas-tin.usal.es/
[37] http://gti-ia.dsic.upv.es:8080/ovamah
[38] http://ihas.usal.es/

aproximación se facilita y simplifica el proceso de diseño de SMA orientado a organizaciones obteniendo modelos de OV que pueden ser implementados en diferentes plataformas. En esta sección se presentarán los aspectos principales del diseño. Dada la extensión del artefactos obtenidos durante el proceso diseño, en este apartado tan sólo se incluye un resumen de los productos principales que permiten describir la unidad organizativa de +Cloud. No obstante, en el Anexo A se presenta de forma detallada y pormenorizada todos los productos obtenidos durante el proceso de diseño.

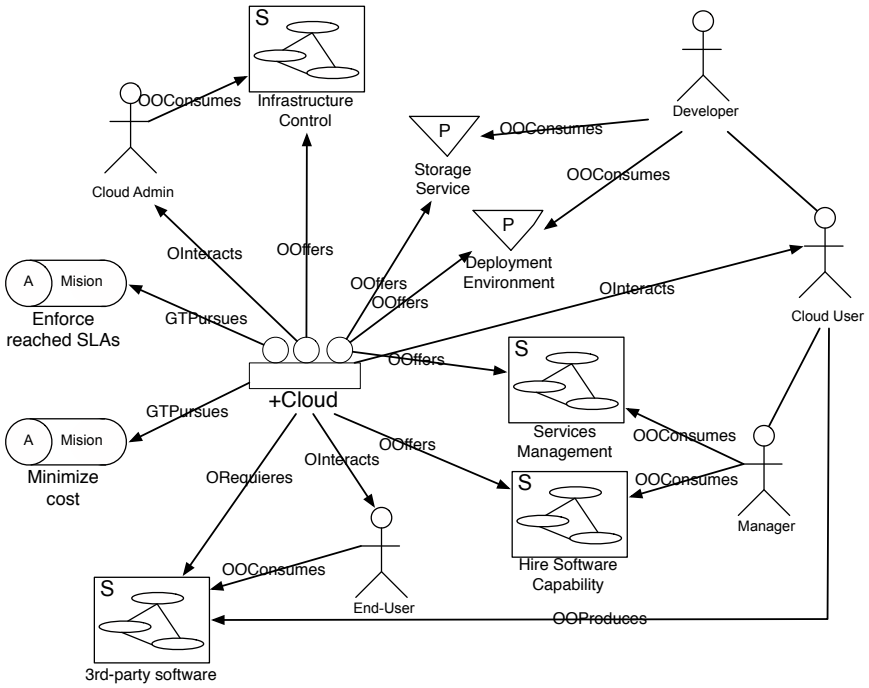

Figura 20.- Vista funcional (misión) del modelo de organización de +Cloud

Siguiendo las pautas indicadas en la guía metodológica GORMAS, una de las primeras tareas es instanciar la vista funcional (*misión*) del modelo de organización, que se presenta en la Figura 20. Esta vista presenta los productos y servicios que ofrece el sistema, los objetivos globales que se persigue (misión y justificación) y los grupos de interés a los que afecta.

Así, en primer lugar, la misión y razón de existencia de la organización será el cumplimiento de los acuerdos de servicios alcanzados con el rol *Cloud User*, pero minimizando para ello los costes asociados a esta misión. En el diagrama mostrado en Figura 20 también se aprecian los tipos de usuarios que hacen uso del sistema (*Cloud Admin*, *Cloud User* y *End User*) y los productos que se ofertan (almacenamiento y despliegue *software*). Para facilitar la interacción de la plataforma también se ofertan los siguientes servicios intrínsecos: gestión de

software, contratación de *software* y control de la infraestructura. Dentro de los servicios ofertados, destacar que la plataforma también ofrece como servicio las aplicaciones que son desplegadas en el sistema por terceros (*Cloud User*), es decir, este tipo de aplicaciones son requeridas por la plataforma para justificar la necesidad de ofertar productos de almacenamiento e infrastructura, sin embargo, dado que una plataforma CC es un simple medio (y no un fín), estas aplicaciones también son servicios que se ofertan a los usuarios finales (*End User*).

Según la metodología GORMAS, en la Fase C se especifican las cuatro dimensiones organizativas (departamentalización, especialización, coordinación y normalización), según el modelo propuesto por Mintzberg [Mintzberg, 1993]. Las dimensiones del sistema +Cloud se presentan en la Tabla 6 correspondiente al Documento C propuesto en la metodología. En primer lugar, en cuanto a la departamentalización se propone un modelo *funcional* agrupando las tareas según la especialización de los agentes que las realizan. La especialización es *horizontal* precisamente debido a este alto grado de especialización, pero también se desea que cada agente individual tenga poder de decisión y responsabilidad sobre las tareas que realiza, lo que conlleva a una especialización donde existe *ampliación vertical*. Por su parte, la coordinación en el sistema se produce de forma *descentralizada,* existiendo *adaptación mutua*, ya que cada agente es el responsable de sus propias tareas pero, para conseguir los objetivos perseguidos con la realización de las mismas, debe interaccionar con otros agentes de la unidad organizativa. Finalmente, en cuanto a la última dimensión, la de formalización, se dice que hay *normalización de resultados* ya que importa la calidad de los mismos, porque los productos tienen que estar dentro de los SLA acordados con los usuarios finales.

A partir de este análisis de las dimensiones organizativas, se analiza la departamentalización y los mecanismos de coordinación existentes dentro del sistema. Para ello, en la metodología se propone un árbol de decisión (disponible en el apartado A.3.2). Sin embargo, a partir del análisis dimensional realizado, si se siguen las diferentes ramas de este árbol de decisión no se obtiene una estructura ideal. No obstante, después de realizar un análisis más detallado, en el que se tiene en cuenta el propio árbol de decisión, el tamaño del sistema y la necesidad de coordinación existente entre tareas, se selecciona la estructura de *coalización* como la más adecuada para modelar el sistema. Se ha seleccionado este patrón de organización ya que las diferentes entidades que forman parte de una estructura de este tipo forman grupos en función de la similitud en cuanto a la funcionalidad, coordinándose entre sí para ofrecer una funcionalidad global más compleja y elaborada, descripción que se ajusta a la perfección con el modelo de coordinación que se perseguía en +Cloud.

C. Dimensiones organizativas
Departamentalización: *Funcional* • Se agrupan las tareas en función de los conocimientos y habilidades de quién las realiza (control de servicios, gestión de infraestructura, etc.). Esto es así, debido al grado de complejidad y especialización de las tareas a realizar, en este sentido se tiene en cuenta que la producción de los resultados es intensiva.
Especialización + Coordinación: *Especialización horizontal con ampliación vertical. Descentralización* • Los roles que realizan las diferentes tareas están altamente especializados debido a la complejidad del entorno con el que se interacciona para satisfacer la funcionalidad que se busca. Este tipo de organización se asocia a la especialización horizontal. Además tienen el control sobre el mecanismo a utilizar para la consecución de los objetivos perseguidos. Sin embargo, también se desea que los roles tengan flexibilidad y capacidad de decisión sobre las tareas a realizar, lo que se asocia a un modelo de organización con ampliación vertical. • La coordinación se realiza de forma descentralizada por lo que se hace necesaria negociación entre los roles para llevar a cabo la toma de decisiones.
Coordinación y Formalización. *Adaptación mutua y normalización de resultados* • Existe adaptación mutua ya que el modo de realización de las tareas es flexible y requiere coordinación entre los diferentes roles implicados. • Existe normalización de resultados ya que importa la calidad de los resultados obtenidos, teniendo en cuenta que se está trabajando en un sistema de alta disponibilidad. El sistema es libre de tomar las acciones que considere oportuno para llegar a alcanzar los resultados previstos.

Tabla 6.- Documento C.- Dimensiones organizativa

A partir del patrón de organización seleccionado, coalización en este caso. El siguiente paso ha consistido en el análisis del proceso de departamentalización, lo que ha dado lugar a la segmentación de la unidad organizativa principal (+Cloud) en las siguientes (sub-)unidades (organizativas especializadas:

- *SLA Negotiation.* Es la unidad organizativa encargada de agrupar las tareas de establecimiento de acuerdos y la negociación de los mismos con el rol externo *Cloud User*. Este proceso tiene que estar totalmente automatizado, así mismo se debe determinar si el sistema tiene capacidad para asumir nuevos acuerdos sin violar los existentes.

- *Service Management.* Es la unidad que tiene asignadas las tareas de monitorización y supervisión de productos que se ofertan y si éstos están siendo ofertados atendiendo a los acuerdos de calidad estipulados en el acuerdo de uso específico con cada usuario. Las tareas de esta unidad organizativa, al igual que las anteriores también tienen que estar automatizadas.
- *Infraestructure Management.* Es la unidad organizativa que controla la infraestructura *hardware* subyacente, es decir, los recursos computacionales (reales o virtuales) del sistema mediante la gestión y monitorización de los mismos. Aunque las tareas también deben estar automatizadas, esta unidad organizativa puede estar supervisada por el A-Agente *Cloud Admin*, el cuál es externo y se asocia al rol interno *Global Supervisor*, cuya funcionalidad ya se ha descrito en el apartado 4.2.1.

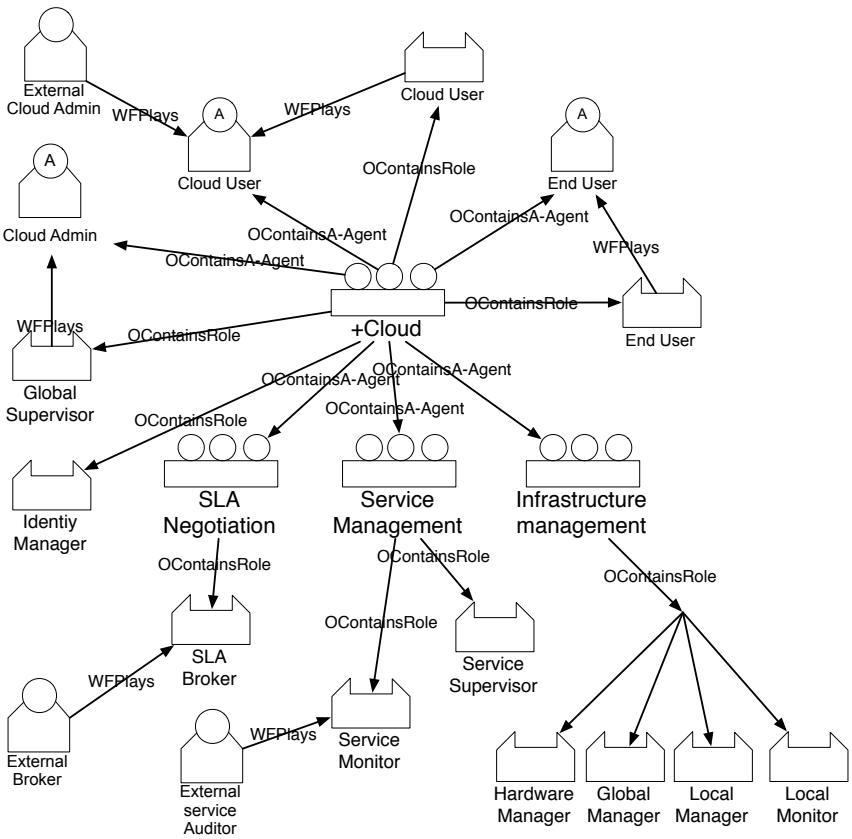

Figura 21.- Vista estructural del modelo de organización actualizado

Siguiendo con el modelo metodológico GORMAS utilizado para el diseño de +Cloud, en la Figura 21 se presenta la vista estructural del modelo de

organización donde se puede apreciar cómo la unidad organizativa principal +Cloud se ha segmentado en las tres unidades que se acaban de explicar. Así mismo, en esta Figura 21 también se aprecia cómo +Cloud se ha diseñado como un sistema abierto, en el que se permite la existencia de agentes externos que ofrezcan dentro del sistema servicios y aporten funcionalidad al sistema. En este sentido, esta vista estructural presenta los diferentes tipos de agentes, ya sean internos o externos, que forman parte del sistema. En este sentido cabe destacar, en primer lugar los roles que están directamente relacionados con agentes humanos los cuáles, han sido claramente identificados previamente. Sin embargo también existen otros dos roles dentro del sistema +Cloud cuyas funcionalidades pueden ser desarrollados por agentes externos. Éstos son el rol *SLA Manager* y *Service Supervisor*, que pueden asociarse a los roles *Broker Cloud* y *Auditor Cloud*, respectivamente, siguiendo el modelo de roles propuestos por el NIST en su arquitectura de referencia [Liu *et al.*, 2011].

Finalmente, como último producto a destacar del diseño de la arquitectura se describirá el entorno en el que se sitúa el SMA organizativo propuesto. En GORMAS durante la fase de análisis se realiza una caracterización del entorno, recogido en el Documento A.3. (Tabla 7). En este sentido, el entorno es dinámico, complejo, con incertidumbre y hostil. Estas características definen al entorno complejo donde los SMA, y especialmente aquellos basados en OV consiguen un mayor grado de efectividad.

A.3- Condiciones de entorno		
Condición	**Valor**	**Justificación**
Tasa de cambio	Dinámico	Ya que constantemente pueden cambiar los acuerdos con los usuarios y el *software* desplegado en el sistema +Cloud.
Complejidad	Complejo	Existe un gran número de componentes en el sistema con múltiples relaciones entre ellos.
Incertidumbre	Alto	Es un entorno dinámico y complejo.
Receptividad	Hostil	Ya que el *software* requerido es desarrollado por terceros y podría contener errores.
Diversidad	Uniforme	Ya que los servicios que se ofertan y se proporcionan son homogéneos (servicios computacionales).

Tabla 7.- Documento A.3.- Condiciones de entorno

El entorno, además de ser el lugar donde se sitúan los diferentes agentes que forman parte del sistema, debe proporcionar los puertos de acceso a la reconfiguración del mismo. Así tal y como se puede aprecia en la Figura 22,

donde se muestra una vista completa del modelo de entorno, los diferentes roles del sistema interaccionan con el entorno para conseguir sus objetivos individuales a través de los puertos que este proporciona, ya sean para acciones de lectura o de modificación:

- Dos aplicaciones que actúan con interfaces con los roles externos del sistema. Es decir, la aplicación que permite monitorizar la infraestructura subyacente del entorno CC, específicamente diseñada para el rol *Cloud Admin* y el escritorio web que dispone de las aplicaciones específicas para el rol *Cloud User*. Los usuarios finales acceden directamente a las aplicaciones de terceros que están desplegadas en el sistema.
- Tres repositorios asociados a la información que debe ser persistente, el almacén de usuario, los acuerdos establecidos con estos usuarios, el *hardware* real disponible y el histórico de uso.
- También se puede apreciar la existencia de diferentes puertos para el acceso a cada uno de los recursos del entorno. En este sentido, por un lado se contemplan los puertos para el acceso a los recursos que ya se han identificado previamente (repositorios e interfaces). Y, por otro lado, destacan otros tres puertos que permiten reconfiguración de la infraestructura subyacente (control del entorno *hardware*, entorno virtual y sistemas de balanceo).

Con la descripción del entorno, termina la descripción de los principales artefactos que se han generado durante el análisis y diseño de SMA basado en OV +Cloud. Una descripción más completa se incluye en el Anexo A de la memoria. La conclusión después de presentar la solución propuesta es que este sistema pretende ser una arquitectura integradora, en la que una organización de agentes pueda gobernar y ejecutar diferentes acciones a lo largo de todo el entorno CC. La novedad de dicha arquitectura radica en el hecho de utilizar un modelo organizativo para llevar a cabo sus objetivos, proporcionando una innovadora solución en entornos altamente dinámicos. En la arquitectura propuesta se plantea una distribución de funciones analizada y diseñada a través de una guía metodológica, quedando abierta la posibilidad de añadir y modificar nuevas funcionalidades de manera sencilla.

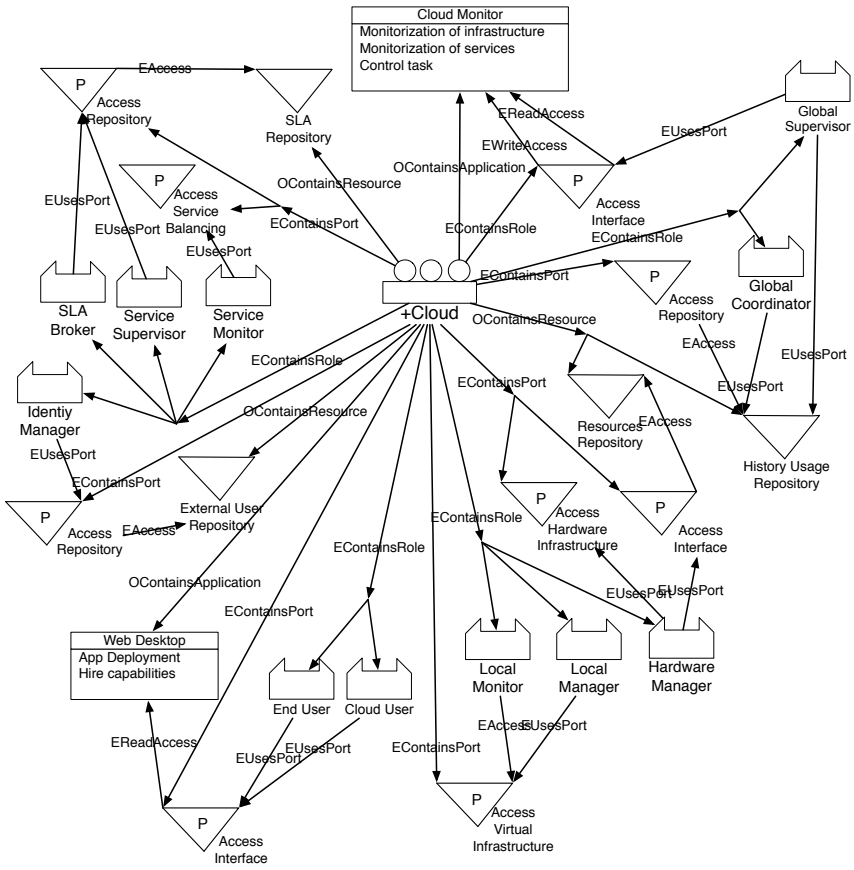

Figura 22.- Modelo de entorno actualizado. Acceso a los puertos del entorno

Una vez que se ha descrito la arquitectura que se ha diseñado, el siguiente capítulo presenta el modelo de razonamiento que siguen los agentes individuales que participan en el sistema para llevar a cabo la distribución de recursos computacionales. Estos algoritmos, siguen la filosofía de los SMA y, basada en la autonomía de los componentes, así como de la toma de decisiones distribuida. Gracias a este modelo, se posibilita no sólo una optimización en el proceso de distribución de recursos, sino también mejorar la disponibilidad del sistema. Este modelo de distribución de recursos es en sí mismo innovador ya que, como se presentará en el apartado siguiente, los modelos actuales son centralizados y, por su puesto, no tienen en cuenta la auto-adaptación dinámica del sistema en función de los cambios que se produzcan en el entorno.

4.3 MODELO DE DISTRIBUCIÓN DE RECURSOS EN UN ENTORNO CLOUD COMPUTING

En el apartado anterior se ha descrito los diferentes componentes del modelo de arquitectura propuesto para el control y monitorización de un sistema CC. Este modelo arquitectónico está basado en OV de agentes inteligentes lo que hace posible la adaptación dinámica del sistema a los cambios que se producen en el entorno [Rodriguez González, 2010]. Para ello, los agentes artificiales, que forman parte de la organización, ejercen el papel de alguno o varios de los roles (y sus responsabilidades asociadas) descritos en el apartado 4.2.1. Estos agentes tienen la capacidad de integrar mecanismos y métodos de razonamiento avanzado, que permiten tomar decisiones de forma individual (pero coordinada) que hace posible no sólo interactuar sobre el entorno, sino también adaptar dinámicamente la propia organización y sus componentes para responder a los cambios que se producen en el propio entorno. El presente apartado tiene como objetivo describir los tipos de agentes especializados en la distribución de recursos computacionales según el modelo propuesto. Algunos de los cuáles, dado que su ciclo de vida está ligado al de las máquinas en la que residen, también permite describir un modelo auto-adaptativo de la arquitectura +Cloud.

Este modelo permite hacer frente a la demanda en los servicios que se ofertan por la plataforma CC, sin violar los acuerdos SLA establecidos. Es decir, gracias a los diferentes agentes existentes con capacidades avanzadas de razonamiento y adaptación que se integra en la arquitectura, es posible distribuir los recursos físicos disponibles entre los diferentes recursos virtuales, y por ende, entre los diferentes servicios que son proporcionados por estos recursos computacionales. En este sentido, el problema reside en cómo distribuir eficientemente los recursos disponibles, haciendo uso únicamente de aquellos recursos necesarios para el correcto funcionamiento del entorno, lo que permite seguir una aproximación que se acerca al modelo de producción tradicional *just-in-time* [Hutchins, 1999]. Con relación a este modelo de distribución, que se presenta a lo largo de esta sección, es necesario destacar que está altamente influenciado por las tecnológicas de virtualización existentes en la actualidad [Fisher *et al.*, 2010].

Tal y como se ha descrito al principio del capítulo (apartado 4.1.2), a nivel general, un sistema CC está compuesto por un conjunto de máquinas físicas que alojan máquinas virtuales, cada una de ellas asociada a un servicio determinado. Cada máquina virtual individual, o nodo (worker) atiende un servicio concreto de la plataforma. Uno o varios nodos pueden atender un único servicio, estos nodos para proporcionar el servicio hacen uso de recursos computacionales. Estos recursos computacionales son ofrecidos al entorno CC por las máquinas físicas que dispone el sistema. La Figura 23 muestra un

pequeño ejemplo del funcionamiento de este modelo en un entorno CC, en él, se puede observar como un servicio tiene asociado tres nodos que se encargan de satisfacer la demanda de peticiones. Estos nodos se encuentran hospedados en diferentes servidores físicos de la plataforma CC. Para que los usuarios finales (rol *End User*) sean liberados de esta complejidad inherente, es necesario que exista un sistema capaz de balancear la demanda de peticiones entre los diferentes nodos que en un momento determinado atienden el servicio.

En cuanto a estas máquinas físicas, es necesario hacer notar que no todas ellas tienen por qué estar en funcionamiento en todo momento. En este sentido, en los sistemas CC modernos los servicios deben activarse o desactivarse en función de la demanda existente. De este modo, se contribuye al ahorro de costes, siguiendo el paradigma computacional *Green Computing* [Hooper *et al.*, 2008] [Bing *et al.*, 2009].

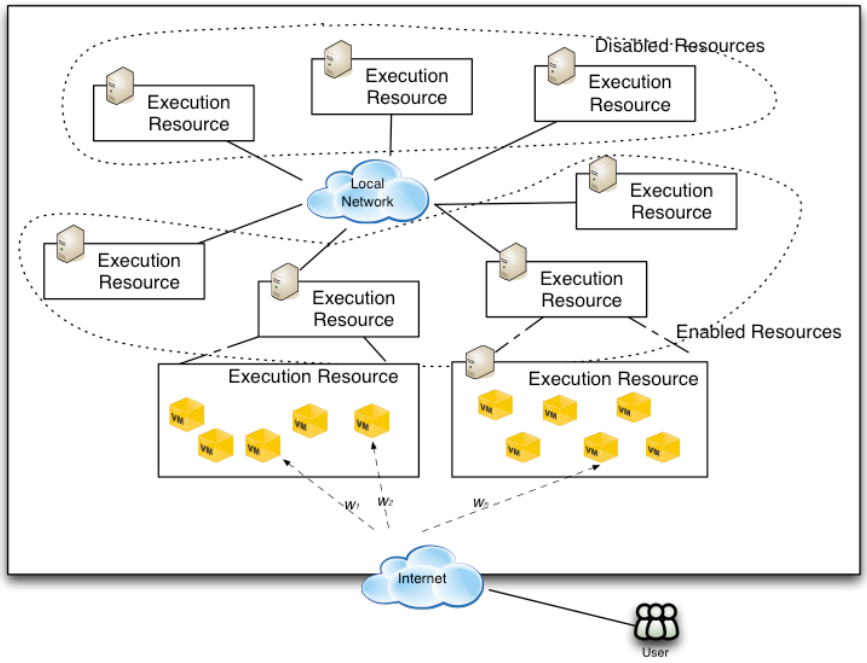

Figura 23.- Distribución de recursos en sistemas Cloud Computing

La distribución de recursos dentro de estos sistemas computacionales es una tarea compleja, donde la infraestructura subyacente ofrece una gran cantidad de posibilidades. Así, es posible distribuir recursos tanto a nivel local, en cada servicio individual o nodo físico, como a nivel global, entre diferentes nodos físicos. En este sentido, el correcto reparto de recursos computacionales en un entorno CC es una de las decisiones clave, debido a que en función de

esta distribución es posible maximizar la eficiencia del entorno y la disminución de los costes asociados. Sin embargo, esta tarea es compleja, ya que requiere tanto del análisis de la demanda actual y futura de los servicios que se alojan, como de los recursos disponibles en cada momento en el entorno CC. Así pues, atendiendo a las capacidades que permite la tecnología subyacente, la distribución de recursos se presenta desde diferentes puntos de vista, tal y como sigue:

- **Distribución a nivel servicios.** A este nivel, el más alto posible, la distribución de recursos consiste en el simple balanceo de carga entre los nodos trabajadores que atienden un determinado servicio. Este tipo de balanceo está altamente extendido desde el nacimiento de los sistemas computacionales de tipo clúster en entornos HPC [Barroso *et al.*, 2003] [Aikins *et al.*, 2007].
  El objetivo habitual en este tipo de reparto de carga computacional consiste en equilibrar la carga de todos los nodos que están ofertando el servicio [Cardellini *et al.*, 1999] [Buyya *et al.*, 2010b].
- **Distribución a nivel infraestructura.** La distribución a nivel infraestructura es un método más complejo y novedoso, que es posible gracias al nacimiento de la tecnología de virtualización [Oguchi *et* Yamamoto, 2008]. De forma previa al nacimiento de esta novedosa tecnología, la distribución de recursos a nivel infraestructura estaba basada en la incorporación de nodos (servidores físicos) al sistema de balanceo, lo cuál en la mayoría de los casos era un proceso manual, por lo que habitualmente la asignación terminaba siendo estática [Beloglazov *et* Buyya, 2010].
  Con el nacimiento de la virtualización, la distribución es mucho más potente, dinámica y, además, puede ser automatizada, ya que los nodos que se incorporan son virtuales y pueden ser situados en cualquier servidor físico gracias al uso de plantillas de servicio para la instanciación, es decir, el arranque de máquinas virtuales de forma dinámica. Incluso, recientemente, con el avance de las técnicas de virtualización y el nacimiento del nuevo *kernel* de Linux[39] es posible modificar los recursos computacionales asignados a una determinada máquina virtual que esté en ejecución.

En el marco de este trabajo, se hablará de distribución de recursos a nivel micro cuando la reasignación de recursos se realice dentro de un servidor físico, y a nivel macro, cuando la reasignación incluya diferentes servidores físicos.

[39] https://www.kernel.org/

4.3.1 TRABAJOS RELACIONADOS

Como se ha venido comentando desde el inicio del capítulo, los recursos de un entorno CC son compartidos entre los numerosos clientes que atienden, lo que implica que sea necesario un ajuste dinámico en función de la demanda. Los modelos existentes de distribución de recursos tienen en cuenta habitualmente tres conceptos básicos [Buyya *et al.*, 2008]: los acuerdos SLA establecidos con los usuarios, el consumo de energía del entorno CC y el propio estado del entorno CC. Este problema, por su complejidad y contemporaneidad está siendo de gran interés por la comunidad científica, la cuál está proponiendo diferentes aproximaciones para hacer frente a este problema abierto en los sistemas CC.

El problema puede observarse desde dos puntos de vista [Buyya *et al.*, 2010b]. Por un lado, la búsqueda del proveedor más barato y eficiente en cuanto al uso de recursos, para lo cuál se sigue un modelo en el que un bróker o gestor de recursos, habitualmente externo, está en permanente contacto con varios proveedores, seleccionando el más adecuado para cada cliente en cada momento. Por otro lado, otro enfoque de mayor interés en el marco de este trabajo, consiste la búsqueda de la eficiencia de recursos dentro de los centros de proceso de datos de un gran número de proveedores de servicios CC. Dentro de este segundo grupo, también se observa el problema desde dos perspectivas, a pequeña y gran escala [Vinothina *et al.*, 2012] [Jung *et al.*, 2011]:

- Un proveedor de recursos CC a gran escala, que disponga de diferentes centros de datos distribuidos a lo largo de todo el mundo, debe dirigir las peticiones de los clientes según dos variables, la distancia al centro de datos y la carga instantánea de cada centro. Así será posible reducir la latencia en la respuesta a los clientes. A este modelo se le conoce como modelo de distribución basado en tiempo (*time-drive adaptive mechanism*) [Jung *et al.*, 2011].
- Nos referimos a pequeña escala, cuando la distribución de recursos se realiza dentro de un centro de datos individual, con independencia de su tamaño. La tarea de distribución de recursos dentro de este medio se realiza por un componente que habitualmente se denomina *Resource Allocation System* (RAS). Dentro de esta perspectiva del problema, en el estado del arte existen principalmente dos aproximaciones [Buyya *et al.*, 2010b]:
    - *QoS-awared based*, u orientados al mercado [Buyya *et al.*, 2008]. Este primer grupo se asocia a un modelo de distribución de recursos orientados al cliente en el que se trata de minimizar los riesgos computacionales con el objetivo de distribuir los recursos

computacionales en función de los acuerdos SLA alcanzados, siguiendo el modelo económico de pago por uso.
Así, según este modelo, las técnicas de gestión de recursos computacionales tienen como objetivo no violar nunca estos acuerdos, proporcionando al usuario final la calidad del servicio que se ha contratado y por lo tanto que el usuario espera recibir. En este sentido, en el estado de arte existen trabajos dentro de esta aproximación como RAS-M [You *et al.*, 2009] que propone un modelo basado en la economía de mercado donde se busca una redistribución de recursos que de lugar a un precio de mercado justo. Otros trabajos utilizan base matemática para la distribución de recursos como la teoría de juegos [Wei *et al.*, 2010a] o la maximización de una función de optimización, basada en técnicas de programación lineal, ya que al fin y al cabo el problema se reduce a la optimización de recursos [Van *et al.*, 2009a] [Van *et al.*, 2009b].

- *Energy-awared based* [Buyya *et al.*, 2010b]. En segundo lugar, la distribución de recursos puede realizarse teniendo en cuenta no sólo los acuerdos SLA preestablecidos, sino también el consumo energético que implica el cumplimiento de éstos. El número de trabajos existente en el estado del arte es menor que en el anterior grupo, y los existentes son más novedosos. Así pues, existe una gran variedad de técnicas como la aplicación de políticas de ahorro de energía en las máquinas físicas o virtuales [Raghavendra *et al.*, 2008], redistribución eficiente de las máquinas virtuales en función del consumo de cada equipo físico [Beloglazov *et al.*, 2012] o modelos que se basan en técnicas de optimización [Kusic *et al.*, 2009]. Todos ellos, se encuentran en un estado de desarrollo inicial que dificultan su aplicación real en grandes centros de cómputo.

El problema, aunque complejo, es simple en su declaración, ya que tan sólo se basa en la redistribución de recursos físicos entre los diferentes nodos virtuales. A partir de estos trabajos que se pueden encontrar en el estado del arte, es necesario destacar que este trabajo de investigación ha centrado el esfuerzo en el desarrollo de algoritmos que tienen en cuenta el consumo energético. En este sentido, el objetivo es reducir el consumo energético necesario para satisfacer los acuerdos SLA que se establezcan con los usuarios de la plataforma. Así mismo, los algoritmos que se han presentado a lo largo de esta sección siguen aproximaciones centralizadas, que utilizan modelos matemáticos y teoría económicas para dar como resultado algoritmos adecuados al problema de resolver.

En este trabajo se sigue una aproximación diametralmente opuesta basada en técnicas de optimización e IA que permiten repartir los recursos siguiendo

un modelo distribuido y escalable que hace posible que el sistema aprenda a lo largo del tiempo. Para ello, se toma como base la arquitectura multiagente basada en OV que habilita la interacción dentro los diferentes componentes de la plataforma, y permite introducir algoritmos de razonamiento avanzados que facilitan el desarrollo de modelos autónomos y distribuidos basados en la auto-adaptación dinámica a los cambios que se puedan introducir en el entorno.

4.3.2 PROPUESTA DE AGENTES ESPECIALIZADOS EN LA DISTRIBUCIÓN DE RECURSOS

La hipótesis de este trabajo de tesis doctoral es que es posible utilizar técnicas de IA, como son los SMA basados en OV, así como los sistemas de aprendizaje automático para elaborar un modelo que permita reducir el consumo energético y, que al mismo tiempo, permita asegurar el cumplimiento de los acuerdos SLA que se hayan alcanzado con los ususarios.

Hasta el momento, se ha presentado el SMA propuesto en el apartado 4.2, sin embargo, esta arquitectura basada en OV es tan sólo una parte de la solución que se propone en el marco de este trabajo de investigación. Esta arquitectura, por ser integradora, hace posible la inclusión de componentes individuales y autónomos (agentes) que incluyen diferentes modelos de razonamiento (agentes especializados) para la toma de decisiones en este entorno complejo y abierto que es un sistema CC. Por lo tanto, a lo largo de este apartado se presentará la segunda parte de la solución. Es decir, estos modelos de razonamiento avanzado y algoritmos que hacen posible la distribución de recursos computacionales dentro de este entorno complejo y abierto.

Los modelos de razonamiento integrados en los diferentes agentes especializados que forman parte de la arquitectura multiagente se describen partiendo de las dos posibilidades de redistribución de recursos que ofrece la tecnología actualmente (servicios e infraestructura) presentados en la introducción del apartado 4.3. Así, como se aprecia en la Figura 24, el algoritmo de distribución sigue un proceso jerárquico de distribución de recursos. En primer lugar, la distribución de recursos se realiza a nivel de servicio, balanceando las diferentes peticiones entre los nodos que se asocian al servicio. Cuando este balanceo de peticiones no es suficiente, y por lo tanto los nodos que atienen el servicio no son capaces de hacer frente a la demanda de peticiones, se realiza una distribución de infraestructura desde una perspectiva micro; en la cuál, se solicita a los diferentes equipos físicos que alojan los nodos de un determinado servicio, que proporcionen más recursos computacionales a cada uno de estos nodos, lo que permite incrementar su rendimiento y, por ende, la cantidad y calidad de peticiones que puede atender por unidad de tiempo. Finalmente, en una tercera fase, cuando tampoco este

segundo proceso de redistribución no ha sido suficiente para hacer frente a la demanda de un determinado servicio, se realiza una distribución de infraestructura a nivel macro. Este último paso, da lugar a la instanciación de un nuevo nodo que puede ser hospedado por una máquina física activa del entorno CC, o por una de las máquinas apagadas que permanecen en estado latente, esperando para ser reactivadas en el momento que se necesiten más recursos.

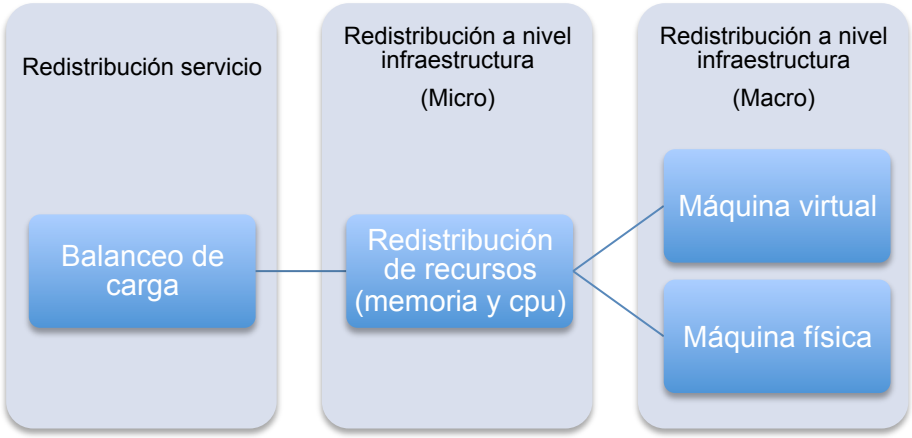

Figura 24.- Secuencia en la redistribución de recursos

En los siguientes apartado se presenta este modelo de distribución de recursos. En primer lugar, en el apartado 4.3.2.1 se presenta la distribución de recursos a nivel servicio. Posteriormente, la distribución de recursos a nivel infraestructura micro y macro se presenta en los apartados 4.3.2.2 y 4.3.2.3, respectivamente. Finalmente, el apartado 4.3.2.4 contiene la descripción del algoritmo que hace posible la compactación de los diferentes recursos computacionales para el ahorro de energía.

4.3.2.1 REDISTRIBUCIÓN A NIVEL SERVICIO

Este tipo de balanceo es el más simple, pero su importancia es clave ya que es el encargado de asegurar que se están cumpliendo los acuerdos SLA relativos a servicios *software* que se hayan alcanzados con los usuarios de la plataforma CC. Dentro de esta redistribución intervienen dos de los agentes presentados en la etapa anterior, que son los agentes de *Service Monitor* y el agente *Service Supervisor*. Tal y como ya se indicó, existe una pareja de estos agentes por cada uno de los servicios y se encuentran situados físicamente en el nodo que realiza el balanceo de carga. A continuación se describe la funcionalidad, tareas y modelo de razonamiento de cada uno de ellos:

- El agente *Service Monitor* es el encargado de controlar el estado del sistema de balanceo. Para ello, implementa algoritmos que permiten

calcular de forma continuada la QoS de cada uno de los nodos que se está balanceando en cada momento

El balanceo de recursos se realiza en base a los pesos de cada uno de los nodos, el objetivo es que todos los nodos tengan un nivel de QoS similar. De este modo, los usuarios perciben que el entorno atiende a sus peticiones de forma homogénea. Para ello, este rol recalcula periódicamente el peso de cada nodo dentro del servicio siguiendo el siguiente algoritmo:

1. Se mantiene un conjunto de grupos de enrutamiento, a cada uno de estos grupos se le asocian diferentes nodos trabajadores. Todos los nodos que pertenecen a un mismo grupo tienen el mismo peso dentro del sistema de balanceo y, por lo tanto, deben tener unos niveles de QoS similares.
2. Para determinar los grupos de QoS se utiliza el algoritmo de agrupamiento del vecino más próximo, basando el cálculo de la distancia entre dos nodos a partir de la distancia euclídea entre los vectores de calidad del servicio en cada nodo $QoS_{n_i}^k = \{\overline{r_1^k} \cdots \overline{r_i^k} \cdots \overline{r_n^k}\}$. A partir de los valores obtenidos, se forman grupos a los cuáles se les asignan el mismo peso con el objetivo de que todos los grupos tengan una tasa de respuesta similar.
3. Para asignar el peso a cada grupo, dado que no puede determinarse mediante un proceso exacto, se realiza siguiendo un proceso iterativo. Así, inicialmente todos los grupos tienen un mismo peso, el cuál se ajusta progresivamente en función de la demanda que sea capaz de admitir cada grupo. Así pues, los grupos con mejor QoS reciben un peso mayor, mientras que los grupos con menor QoS reciben un peso menor. Después de un periodo de estabilización del proceso de balanceo en el que se recalcula el QoS con el nuevo reparto de pesos, el proceso se repite desde el punto 2, volviendo a calcular los grupos afines teniendo en cuenta el nuevo reparto de pesos.

Finalmente, este agente de monitorización del servicio también será el encargado de almacenar periódicamente la calidad del servicio en el repositorio de histórico de la demanda del servicio. Así, para un tiempo t y un servicio k, se almacenará un vector $U_t^k = \{QoS^k, n, timestamp\}$ donde n es el número de nodos que están atendiendo el servicio en el momento $timestamp$.

- El agente *Service Supervisor* del servicio tiene un papel fundamental, ya que a partir de los datos generados por el agente *Service Monitor* debe determinar si es necesario asignar más o menos recursos al servicio.
    - Por un lado, tienen que decidir cuándo retirar un subconjunto de recursos asociados al servicio. En este proceso se elimina uno de

los nodos que están asociados al servicio. Al retirar un nodo de ejecución, el resto de nodos asociados al servicio tendrán que atender las peticiones del nodo que se elimina. Este incremento de carga en cada uno de los nodos lleva asociado un periodo de estabilización en el que el agente *Service Monitor* recalculará los nuevos parámetros de QoS para cada uno de los nodos que todavía atienden al servicio y, posteriormente, se ejecutará el *algoritmo de determinación de los grupos de enrutamiento* y los nodos asociados a estos grupos.

- Por otro lado, asociar más recursos a un servicio se realiza cuando los nodos existentes con los recursos que tienen asignados, no son capaces de atender la demanda de peticiones según los acuerdos SLA establecidos con los usuarios, lo que conlleva que se violen los acuerdos que se han alcanzado con los usuarios de los servicios.

En ambos caso, el problema consiste en determinar cuándo es el momento adecuado para iniciar un proceso de asignación o retirada de recursos a un servicio:

- La retirada de recursos se realiza cuando el grupo de enrutamiento cuya calidad de servicio es más alta, tiene 2 o más nodos asignados, después de que se haya ejecutado el algoritmo determinación de los grupos de enrutamiento n. En este caso, se elimina uno de los nodos, debido a que sino se hiciera se estarían infrautilizando recursos activos. Este proceso de parada de nodos puede dar lugar a la dispersión de los nodos en varios servidores físicos del entorno CC. Este estado de dispersión se produce cuando existen varios nodos físicos con recursos libres, es decir, disponen de recursos que no están asociados a máquinas virtuales. Para asegurar la eficiencia del sistema se realiza un proceso de compactación a través del algoritmo descrito en el apartado 4.3.2.4.
- La solicitud de nuevos recursos, es realizada por el agente *Service Supervisor* cuando un incremento en la demanda de peticiones no hace posible que las peticiones se devuelvan en un tiempo adecuado según los acuerdos SLA acordados. Este proceso implica una redistribución de recursos a nivel infraestructura. Para ello se ejecuta una distribución de recursos a nivel micro en primer lugar, la cuál se describen en el apartado 4.3.2.2. Cuando el problema no se elimina en este primer paso, se procede a realizar una distribución de recursos a nivel macro, cuyo procedimiento se describen en el apartado 4.3.2.3.

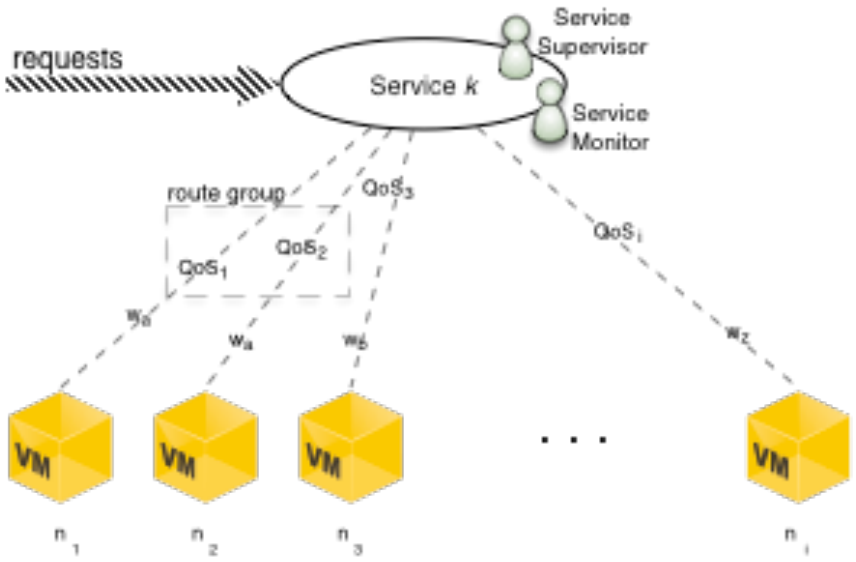

Figura 25.- Redistribución de recursos a nivel servicio

4.3.2.2 REDISTRIBUCIÓN A NIVEL INFRAESTRUCTURA (MICRO)

La redistribución desde un punto de vista de infraestructura a nivel micro se asocia a la redistribución de recursos que se realiza dentro de una máquina física asignando diferentes recursos entre las diferentes máquinas virtuales que aloja y reservando suficientes recursos para el servidor principal. Este proceso está gobernado por dos agentes especializados de la arquitectura multiagente basada en OV, el agente *Local Monitor* y el agente *Local Manager*. Existirá una instancia de estos agentes en cada uno de los servicios físicos de la máquina. El procedimiento es el siguiente:

- El agente *Local Monitor* es el encargado de recuperar información sobre el estado de cada máquina virtual y del servidor donde están alojadas. Sin embargo, no podrá en ningún caso modificar las características de asignación.
- Dependiendo de la tecnología de virtualización subyacente (OpenVZ[40], KVM[41], Xen Server[42], etc.), este agente está formado por un único componente que se aloja en la máquina principal o tendrá subcomponentes en cada una de las máquinas virtuales que hospeda el

[40] http://openvz.org/
[41] http://www.linux-kvm.org/
[42] http://www.xenserver.org/

servicio físico las cuáles periódicamente envían al agente *Local Monitor* el estado de cada máquina virtual.
En definitiva, el agente monitor mantendrá actualizada una matriz con la siguiente información:

$$exec^{e_1} = \begin{Bmatrix} VM_1 \\ \vdots \\ VM_m \\ PR^{e_1} \end{Bmatrix} donde \begin{cases} PR^j = \{hostname, IP, mac, state, M_{max}, vcpu_{max}, M_{min}, vcpu_{min}\} \\ VM_i = \{IP, state, M, vcpu, M_i \overline{p_{cpu}}\} \end{cases}$$

- El agente *Local Manager* es el que tiene poder para gobernar la asignación de recursos dentro de la máquina incluyendo las siguientes funcionalidades: (i) actualizar recursos asignados a cada una de las máquinas virtuales, (ii) instanciar una nueva máquina virtual a partir de una plantilla y, finalmente, (iii) detener la ejecución de una máquina virtual. A partir de estas tres funcionalidades es posible redistribuir los recursos dentro de un servidor físico. Los recursos que se pueden redistribuir son el número de procesadores virtuales (*vcpu*) y la memoria física (M).

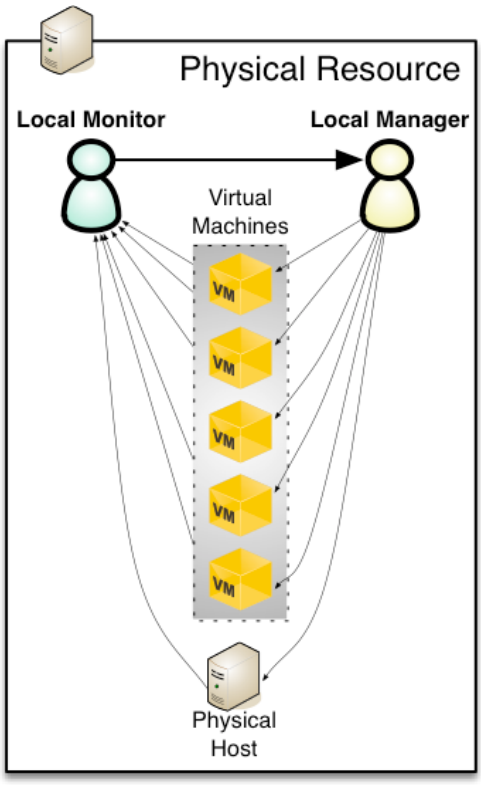

Figura 26.- Redistribución de recursos de infraestructura a nivel micro

El proceso de distribución de recursos tiene en cuenta el tipo de servicio al que pertenecen las máquinas virtuales que existen en ejecución, cuya información se recoge de la plantilla que permite su instanciación, la cuál también determina los recursos mínimos que se le deben asignar. El proceso de reasignación a nivel interno lo inicia un agente de tipo *Service Supervisor* cuando detecta que los recursos mínimos que se le han asignado a su subconjunto de máquinas virtuales no son suficientes. Este agente envía un mensaje, indicando la necesidad de recursos, a cada uno de los agentes *Local Manager* de las máquinas físicas que hospedan los nodos para ese determinado servicio. Cada una de estas máquinas inician un proceso reasignación de recursos en paralelo, que da como resultado un incremento global de los recursos asignados a cada servicio.

Cuando se inicia un proceso de distribución de recursos, el agente *Local Manager* de la máquina configura una estructura de información de asignación de recursos como la que se muestra en la Tabla 8, a partir de la información suministrada por el agente *Local Monitor* de la misma máquina. Donde se recoge información instantánea I_{ei}^{t} del estado completo de la asignación de recursos en un tiempo t, el grado de uso actual de éstos, el servicio y la prioridad que tiene el nodo en el sistema de balanceo del servicio según el cálculo realizado por el agente *Service Monitor* asociado. Estos tres últimos parámetros resultan de importancia debido a:

- $ID_{s,}$, asociado al identificador del servicio que permite determinar si es una máquina virtual correspondiente a un nodo de servicio de tipo infraestructura o *software*. En el caso de los servicios de infraestructura, el usuario establece un acuerdo SLA para unas determinas condiciones *hardware* que no pueden modificarse. En el caso de corresponderse con un servicio de *software*, donde la máquina virtual es un nodo asociado a un servicio, sí que se podrían modificar sus propiedades.
- *Type*, que determina si son servicios de tipo *stateless* o *statefull*, recordar que como ya se indicó en el apartado 4.1.3 las aplicaciones *stateless* pueden tener varias instancias en ejecución de forma simultánea, ya que cada una de ellas no almacena el estado del servicio. Mientras que las máquinas virtuales *statefull* únicamente tienen una instancia en ejecución. Por lo tanto, si este servicio es el que necesita más recursos es prioritario respecto al resto.
- *Priority*, que determina la prioridad de la máquina virtual en el proceso de balanceo de carga. La cuál da una medida acerca del grado de rendimiento de la máquina virtual con respecto al resto de nodos del mismo servicio. Se corresponde con un valor dentro del rango 0 a 1 y se calcula mediante la expresión simple $Priority_i = {}^{w_i}/_{w_{max}}$. Este valor es

calculado por el agente *Local Manager*, mediante la interrogación al agente *Service Monitor* del servicio
Así pues, los valores próximos a o tienen una prioridad baja y, por lo tanto, no están ofreciendo la QoS necesaria, y los valores próximos a 1, tienen un nivel de QoS alta en comparación con otras instancias que atienden el servicio.

	processing			**Memory**				**General**		
	$vcpu_{max}$	**$vcpu_{min}$**	**$vcpu_{used}$**	**M_{max}**	**M_{min}**	**M_{used}**	**$M_{asigned}$**	**ID_s**	**type**	**priority**
VM_1										
VM_2										
VM_m										
PR^{e_1}										

Tabla 8.- Instancia de servidor físcio en la redistribución de recursos de infraestructura a nivel micro

Antes de describir el procedimiento de redistribución de recursos que realiza el agente especialido *Local Manager*. En primer lugar, destacar que éste sólo afecta a las máquinas que atiendan servicios *software*. Aquellas máquinas con características estáticas no se tienen en cuenta dentro del procedimiento. Así, de las n máquinas que hospeda un servidor físico, sólo computan en los algoritmos que se presentan a continuación las m máquinas que atienden servicios *software* (*stateless* o *statefull*) cuyas características pueden ser dinámicas.

El proceso de distribución de recursos es diferente para la asignación de *vcpus* virtuales y de memoria. En primer lugar, para la asignación de *vcpus* se utiliza la máquina de estados que se presenta en la Figura 27. El objetivo de este procedimiento es no disponer de recursos de ejecución infrautilizados y, si éstos existiesen, tan sólo podrían estar asignados a la máquina física. Así el procedimiento de asignación de *vcpus* se inicia verificando si la máquina física tiene *vcpus* que no estén asignadas, es decir:

$$PR^{e_1}(vcpu_{max}) \neq \sum_{1}^{m} VM_i\left(vcpu_{used)}\right) + PR^{e_1}(vcpu_{used})$$

Por lo tanto, puedan asignarse a la máquina o máquinas que necesita recursos. Si no existe ninguna *vcpu* que no esté asignada, se itera sobre cada máquina virtual para comprobar si la cantidad de procesamiento asignado está siendo infrautilizado. Si en algún caso, el valor del parámetro $vcpu_{used}$ es menor que una constante definida por el administrador, entonces se reasigna uno de los núcleos de esta instancia a aquella máquina virtual que necesite un mayor número de recursos.

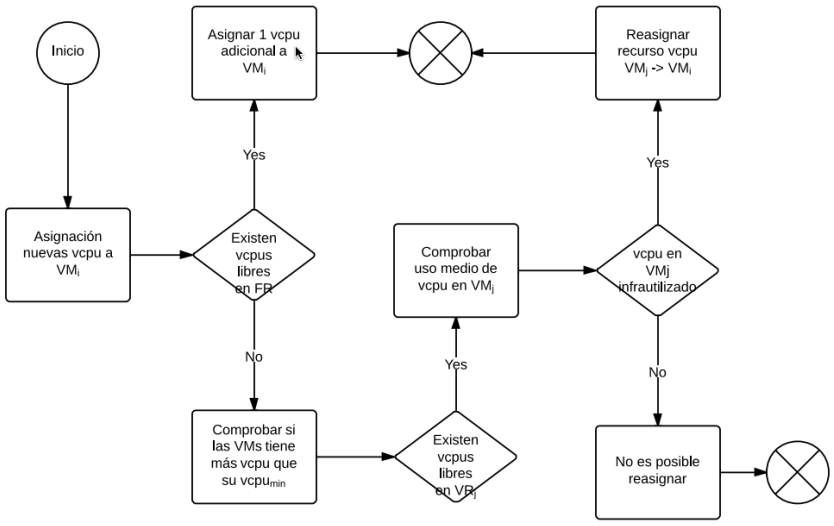

Figura 27.- Reasignación de infraestructura (vcpu) a nivel micro

Por otro lado, el proceso de reasignación de memoria es más complejo y para ello se sigue un modelo de programación lineal que determina la asignación de recursos óptima. No obstante el paso inicial que se realiza es similar al proceso de asignación de unidades de procesamiento virtual (*vcpus*). En este sentido, si el servidor físico tiene memoria libre que puede asignarse, se asigna directamente a la máquina virtual demandante, cumpliendo la restricción de memoria máxima asignable a la plantilla

$$VM_i(M_{asigned}) \leq VM_i(M_{max}).$$

Este proceso de asignación directa se realiza en base a la siguiente expresión donde $VM_i(M'_{asigned})$ se corresponde a la memoria disponible previamente en la máquina virtual y se basa en la prioridad que tiene la máquina virtual en el balanceo de carga. Se ha modelado este procedimiento de este modo para no modificar en gran medida este reparto de prioridades en el balanceo previamente asignado y, además, no asignar todos los recursos libres a una única instancia virtual.

$$\begin{cases} VM_i(M_{asigned}) = VM_i(M'_{asigned}) + VM_i(M_{max}) * priority_i \;\; si \; (VM_i(M_{max}) * priority_i) < PR^{e_i}(M_i) \\ VM_i(M_{asigned}) = VM_i(M'_{asigned}) + PR^{e_i}(M_i) \;\; si \; (VM_i(M_{max}) * priority_i) > PR^{e_i}(M_i) \end{cases}$$

En caso de no existir memoria libre, será necesario reasignar memoria, es decir, retirar memoria de un conjunto de instancias de ejecución y asignársela a la máquina demandante. A nivel técnico este proceso es posible siempre y cuando no se elimine memoria que esté siendo utilizada por las instancias en ejecución. Es decir, es posible redistribuir la memoria que está siendo

infrautilizada siempre dejando un margen de seguridad (M_l) al retirar memoria de un nodo activo.

Así pues, la memoria infrautilizada de cada máquina virtual es la siguiente:

$$VM(M_i) = VM\left(M_{asigned}\right) - VM(M_{used})$$

Y por lo tanto, la memoria infrautilizada de todo el servidor físico vendrá dada por la siguiente expresión:

$$PR'^{e_1}(M_i) = \sum_1^m VM_i(M_{assigned}) - VM_i(M_{used})$$

El problema consiste en cómo redistribuir la memoria infrautilizada entre las diferentes máquinas virtuales y el servidor físico, de forma que se reduzca la cantidad de memoria infrautilizada. El problema es de optimización y se resuelve utilizando programación lineal, de forma que se tienen m variables independientes que se corresponden con la nueva asignación de memoria. De forma que la función que hay que minimizar está compuesta por el sumatorio de la memoria infrautilizada que se asignará a cada máquina virtual por el porcentaje de infrautilización actual, lo que permitirá ponderar cuál es la ganancia en la asignación de memoria en cada caso.

$$min\ PR^{e_1}(M_i) = \sum_1^m x_i \left({VM_i(M_i)}/{PR^{e_1}(M_i)}\right)$$

Este modelo de minimización tiene las dos restricciones siguientes:

- La suma de las variables objetivo será igual a la suma de la memoria infrautilizada inicial.

$$\sum_1^m x_i \leq PR'^{e_1}(M_i)$$

- También será necesario que cada variable x_i sea mayor que el margen de seguridad.

$$x_i, \dots, x_m > M_l$$

Una vez resuelto el problema obtendremos que el valor de memoria asignada para cada máquina virtual está determinada por la siguiente expresión:

$$VM_i\left(M_{asigned}\right) = VM_i(M'_{used}) + x_i$$

Una vez que se han determinado el nuevo reparto de memoria, si los resultados obtenidos en el proceso de cálculo de la nueva asignación son superiores a los recursos que tuviera la máquina del servicio que inicia el cambio, entonces esta nueva asignación se aplica por el agente *Local Manager* y se informa al agente *Service Supervisor* asociado a el servicio que inició el proceso que se han asignado recursos al nodo.

Finalmente, es importante destacar que este proceso puede ser iniciado por varios servicios a la vez, y en este caso es necesario tener en cuenta las siguientes consideraciones para el agente *Local Manager*:

- El proceso de cálculo de nuevos recursos no depende del tipo de servicio, sino de la prioridad de las máquinas dentro de cada servicio y de los recursos que esté utilizando en un determinado momento.
  Por lo tanto, si dos o más servicios solicitan un incremento de recursos a nivel local sólo se aplicarán los cambios propuestos si:
  1. Si, la maquina tiene recursos libres que pueda asignar a los recursos.
  2. Si la máquina no tiene recursos libres, sólo se realizará el cambio de recursos si para todos los servicios que hayan solicitado un incremento de los recursos, el cálculo de la nueva asignación incrementa los recursos que tiene asignados en un determinado momento cada uno de los nodos asociados a los servicios que solicitan más recursos.
  3. Si la máquina no tiene recursos libres, y en el paso dos no se ha encontrado una solución adecuada para todos los demandantes de recursos, entonces se procederá a informar al agente que inicia el proceso, es decir, al agente *Service Supervisor*, que no hay recursos suficientes para que tome las medidas que estime oportunas.
     En este sentido, la solución es aplicar un procedimiento de redistribución de infrastructura a nivel macro, el cuál se describe en el apartado 4.3.2.3.
- En cualquier caso, una vez que se haya aplicado un cambio en los recursos, no se podrá realizar un nuevo reajuste de servicios hasta que se estabilice cada una de las máquinas virtuales y servicios atendiendo a los recursos asignados. El tiempo de espera entre dos reasignaciones será una constante supervisada por el administrador del entorno CC.

4.3.2.3 REDISTRIBUCIÓN A NIVEL INFRAESTRUCTURA (MACRO)

La redistribución de recursos a nivel macro se inicia por un agente de tipo *Service Supervisor* cuando detecta que los recursos asignados al servicio al que se asocia no son suficientes. Solo se podrá iniciar una redistribución a nivel macro, cuando previamente se haya iniciado una redistribución a nivel micro que como resultado de la misma, no se hayan satisfecho las necesidades computacionales del servicio.

La redistribución de recursos a nivel macro se realiza por los agentes *Global Manager* los cuáles se encuentran situados en cada una de las máquinas físicas y tiene acceso a la información que proporciona el agente de *Local Monitor* en la misma máquina en la que se sitúan. Así mismo, también tiene autoridad sobre el agente *Local Manager* pudiendo indicarle a este la necesidad de

arrancar una nueva máquina de un determinado servicio con unas características determinadas. No obstante, el agente *Global Manager* no podrá realizar ninguna acción si el agente *Local Manager* está realizando una adaptación a nivel local.

Este agente *Global Manager* es una agente altamente especializado que implementa una arquitectura deliberativa de tipo CBR-BDI [Corchado *et* Laza, 2003] [Corchado *et al.*, 2008] [Bedia, 2004] [Bajo *et al.*, 2008], [Carrascosa *et al.*, 2008]. Antes de detallar el procedimiento, se hará una breve revisión del modelo de razonamiento basado en caso (CBR, *Case-based Reasoning*). Este modelo utiliza la base de razonamiento del pensamiento humano, en el que se recurre a experiencias pasadas para resolver nuevos problemas [De Mántaras *et* Plaza, 1997]. Así pues, si en un tiempo pasado se decidió resolver un problema utilizando una determinada solución y, una vez aplicada esa solución se obtuvo un determinado resultado, entonces parece lógico que si se presenta un nuevo problema con características similares al que se resolvió previamente en el pasado, se recurra a esta experiencia adquirida para dar solución el nuevo problema. Por tanto, el modelo se basa en la idea de que problemas similares tienen soluciones similares. Sin embargo, carecer de problemas similares no supone que el sistema no sea capaz de proponer buenos resultados, sino que la reutilización de memorias pasadas se convierte entonces en un proceso creativo. Sea cual sea el resultado de este proceso creativo, el individuo aprende una nueva experiencia, ya sea positiva o negativa.

En este modelo de razonamiento, el concepto de caso es fundamental. Así pues, un caso es un fragmento de conocimiento contextualizado que representa una experiencia [Schank, 1982] que no es más que una terna que incluye el modelo de problema, el modelo de la solución y el resultado de su aplicación.

```
Case: <Problem, Solution, Result>
Problem: initial_state
Solution: sequence of <action, [intermediate_state]>
Result: final_state
```

Figura 28 - Modelo de caso en un sistema de razonamiento CBR

El sistema debe estar formado por dos componentes fundamentales: la memoria de casos y el mecanismo de razonamiento. En primer lugar, la memoria de casos no es más que un sistema de almacenamiento dónde se extraen las soluciones anteriores y en ella se almacena lo aprendido, por ello es la encargada de mantener la representación y organización de los casos. Por su parte, el ciclo de un sistema CBR está formado por cuatro procesos secuenciales [De Mántaras *et al.*, 2005] [Aamodt *et* Plaza, 1994] tal y como sigue:

- **Etapa de recuperación.** Es la primera etapa que realiza un sistema CBR. En ella se realiza la recuperación de casos, esto es, el acceso a los casos almacenados que cuentan con una descripción de problema más similar a la del problema actual. Para ello, lo habitual es utilizar un algoritmo que garantice el acceso rápido y eficiente, aplicando una métrica de similitud que determina cuál o cuáles de ellos son los mejores casos [Golding *et* Rosenbloom, 1989].
- **Etapa reutilización.** A partir de los casos más similares, la reutilización o adaptación consiste en trabajar con las soluciones correspondientes a los casos más similares recuperados en la etapa anterior para poder obtener una solución al problema actual. Trabajar con las soluciones significa modificarlas y combinarlas, o simplemente decidir cuál de ellas es la más óptima y, por lo tanto, reutilizarla.
- **Etapa de revisión.** Fase en la que se comprueba la bondad de la solución finalmente aplicada para resolver el problema actual [Leake *et al.*, 2000]. Se comprueba si la solución propuesta en la etapa anterior es apropiada para el caso actual. Para ello se utiliza un sistema experto de conocimiento, o bien una persona experta. En ocasiones, en esta etapa se puede realizar una reparación de los fallos o errores detectados.
- **Etapa de retención y aprendizaje.** En esta última etapa se aprende a partir de la nueva experiencia adquirida. Para ello se almacena el caso actual y la solución aplicada para resolverlo. Además se tiene en cuenta el resultado obtenido en la etapa de revisión para asignar una eficiencia al caso. De esta forma el caso puede ser indexado en la memoria de casos.

Como ya se ha indicado el proceso de distribución de infraestructura a nivel macro es iniciado por el agente *Service Supervisor* asociado al servicio con dificultades para atender la demanda. Este agente alerta a todas las máquinas físicas que hospedan nodos del servicio que es necesario una redistribución de recursos a nivel global para hacer frente a la demanda de peticiones. El mensaje es recibido por el agente *Global Manager* de cada servidor físico que a su vez alerta el resto de agentes *Global manager* del entorno CC. Existe un agente *Global Manager* por cada máquina física activa existente en el entorno CC. Estos agentes, solicitarán al agente *Local Monitor* de la máquina una instantánea del estado del uso de los recursos (I_{ei}^{t}). A partir de esta instantánea, este agente especializado evalúa la cantidad de recursos disponibles, es decir, aquellos que no están asignados a ninguna máquina virtual. Si esta cantidad de recursos fueran mayores que los mínimos indispensables para instanciar un nodo asociado al servicio (información que se especifica en la plantilla del servicio), entonces iniciaría el proceso de razonamiento que se explica a continuación para determinar la cantidad de recursos que se pueden reservar al nuevo nodo de ejecución que demanda el

agente *Service Supervisor* del servicio. Si no tuviera recursos libres o estos fueran menores que los demandados como mínimo por el servicio, entonces el agente *Global Manager* determina que el nodo físico no formará parte del proceso de asignación global. En el caso de que ninguna máquina atienda a la necesidad de incremento de servicio, entonces se solicitará al agente *Hardware Manager* el arranque de un nuevo servidor físico donde se instanciará un máquina virtual en base a las características mínimas definidas en la plantilla del servicio.

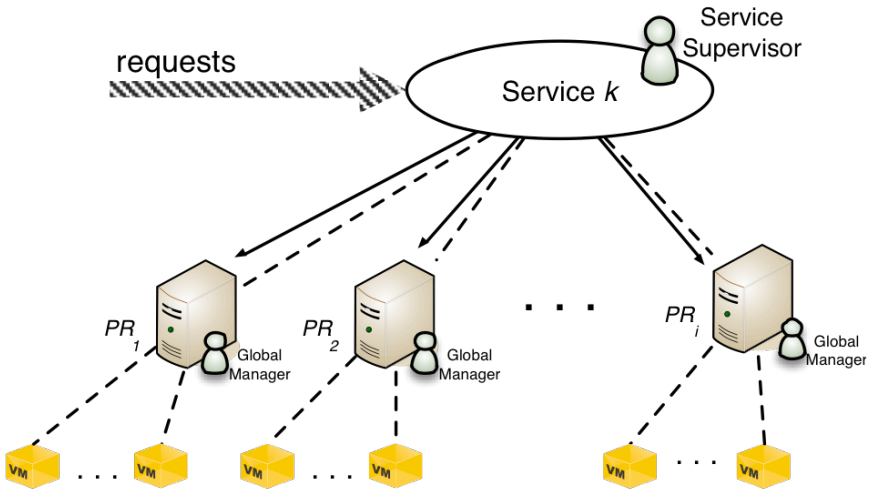

Figura 29.- Inicio distribución de infrastructura a nivel macro

Cada agente de tipo *Global Manager* que atiende la petición de recursos implementa un modelo de razonamiento avanzado basado en un sistema CBR-BDI. Este conjunto de agentes especializados, como están situados en las diferentes máquinas físicas del sistema, ejecutan de forma paralela el algoritmo de razonamiento, que se explica a continuación, determinando de forma individual una solución al problema de demanda del servicio. Estas soluciones individuales se envían al agente *Service Supervisor* que ha iniciado la demanda de recursos, para que seleccione aquella que incluye más recursos para la máquina virtual que se instanciará, priorizando la memoria en detrimento de la asignación propuesta de unidades de procesamiento virtuales (*vcpu*).

El proceso de razonamiento en cada nodo físico, desarrollado por el agente especializado *Global Manager* está basado en un sistema CBR como ya se ha mencionado. Este modelo se basa en la experiencia adquirida en el paso mediante el almacenamiento de casos similares. La memoria de casos es central a todo el sistema CC, de forma que el conocimiento global del sistema pueda ser compartido por cada uno de sus miembros, en este caso el agente

Global Manager. Dado que esta memoria puede crecer exponencialmente como estrategia de mantenimiento se utiliza una base de datos sin esquema de alta velocidad que permite un acceso rápido a los datos almacenados.

En primer lugar, resulta necesario definir el concepto de caso dentro del modelo de razonamiento $C = \{P, S(P), E\}$ donde:

- P se corresponde con la descripción del problema, esta descripción tiene una representación matricial asociada a la instantánea de uso de recursos I_{ei}^{t}, junto con la descripción de la plantilla de servicio que se pretende instanciar y la marca temporal que denota el instante en el que se detecta el problema.

$$P = \{I_{ei}^{t} \quad VM_{t}^{k} \quad timestamp\}$$

- $S(P)$ se asocia a la solución del problema P, que vendrá determinada por una instantánea similar a la del problema donde se incluyan los recursos asociados

$$S(P) = \{M, vcpu\}$$

- Finalmente, la eficiencia (E) se mide desde ambas perspectivas (micro y macro) de la distribución de infraestructura:
  - Por una lado, la eficiencia a nivel micro (E_m) se asocia al grado de eficiencia de la solución propuesta dentro del servidor físico donde se instancia la máquina virtual. Este grado de eficiencia es planteada por el agente *Local Monitor* en función el porcentaje de uso del procesador y de la memoria asignada. Ambos porcentajes están entre o y 1, de forma que la solución es más eficiente a medida que los resultados se acerquen a 1.

$$E_m = \left\{\overline{p_{cpu}}, M_{used} \middle/ M_{assigned}\right\}$$

  - Por otro lado, la eficiencia a nivel macro (E_M) se asocia al grado de eficiencia desde el punto de vista del servicio, determinando si una vez que se ha aplicado la solución propuesta ha sido necesario iniciar un proceso de distribución de recursos de infrastructura a nivel macro. En este sentido, este grado de eficiencia mide el número de nodos adicionales que ha necesitado el servicio.

$$E_M = \{\mathrm{n}\}$$

Por lo tanto la eficiencia viene dada según la expresión $E = \{E_m \quad E_M\}$

El proceso CBR se inicia recuperando los casos anteriores que son similares de la memoria, para ello se seleccionan aquellos casos más similares atendiendo a las siguientes reglas:

1. Se seleccionan los casos de las máquinas físicas con características similares, con un grado de eficiencia superior al 90%. Las máquinas físicas similares se evalúan en función del parámetro *benchmark* que caracteriza cada máquina física. Se recuerda, que este parámetro se almacena en el vector que identifica a cada una de las máquinas físicas (PR) que se detalló en el apartado 4.1.2.1.
2. A partir de este subconjunto de casos recuperados se configura un vector por cada caso que incluye el número de máquinas virtuales existentes en el caso y los recursos libres disponibles $C_i = \{n, M, vcpu\}$. A partir de este vector se evalúa la similitud con respecto al vector que representa el caso actual utilizando para ello la distancia euclídea, lo que permite seleccionar los casos más similares.
3. Posteriormente, de este conjunto de casos se seleccionan aquellos en los que en las experiencias pasadas estuvieran implicado en el servicio que está demandando recursos en este caso durante un periodo de tiempo similar al que se produce el caso, utilizando para ello un patrón de uso de la web en una semana [Abraham, 2003].

A partir de los casos recuperados, durante la fase de reutilización se elabora una solución al problema:

- Si la base de casos no tuviera ningún caso previo similar, entonces la solución al problema se asocia a los recursos mínimos determinados en la plantilla de instanciación del servicio:

$$S(P) = \{M_{min}, vcpu_{min}\}$$

- Si en cambio, si se recuperan casos similares, la solución al problema sería la del caso más cercano multiplicado por la eficiencia del caso:

$$S(P) = \{\mathrm{M}' * (E_m(1) + E_M(1)), vcpu'(E_m(2))\}$$

- Si los valores de asignación de la solución anterior son mayores que los valores asumibles por el equipo, debido a que no tienen tantos recursos disponibles. El resultado del caso serán los máximos recursos disponibles por la máquina.

$$S(P) = \{PR(M_{\max}), PR(vcpu_{min})\}$$

Una vez que se ha calculado la solución al caso, este es enviado por cada uno de los agentes especializados *Global Manager* que se hospedan en servidores físicos con recursos suficientes al agente *Service Supervisor*. Este agente selecciona reactivamente el nodo que ofrezca más recursos a la nueva máquina virtual. Posteriormente durante la etapa de revisión una vez instanciado el nuevo nodo asociado se evaluará su uso desde una perspectiva micro y macro obteniendo el valor de la eficiencia de la solución. Finalmente

durante la última etapa del ciclo CBR propuesto, se guarda el caso junto con su eficiencia para que pueda ser utilizado en posteriores readaptaciones.

El modelo de adaptación propuesto es distribuido lo que permite mejorar la alta disponibilidad del sistema, ya que la toma de decisiones se realiza a lo largo de todo el sistema CC. Además, este modelo permite distribuir la potencia de cálculo que requiere la obtención de la solución, y por tanto, reduce el impacto que la búsqueda de una solución pueda tener en el entorno CC en su conjunto.

4.3.2.4 REDISTRIBUCIÓN A NIVEL INFRASTRUCTURA (COMPACTACIÓN)

A lo largo del modelo de distribución de recursos propuestos no se ha mencionado el uso de una de las características claves de la tecnología de virtualización, que es la migración de máquinas entre servidores físicos. En el proceso de incremento de recursos no se ha utilizado esta característica, ya que la práctica demuestra que exige un coste adicional de recursos y ancho de banda que afectan negativamente al servicio y a los equipos físicos que alojan la máquina a migrar (emisor y receptor).

No obstante, esta característica resulta muy efectiva a la hora de compactar las máquinas virtuales en el menor número de servidores posibles, de forma que se pueda apagar o hibernar el conjunto de servidores físicos que no tengan asignada ninguna máquina con lo que se consigue un incremento de la eficiencia energética. Según el modelo de distribución que presenta el sistema, el procedimiento es simple.

El problema se origina cuando las máquinas tienen una gran dispersión, debido a que cuando un agente de tipo *Service Supervisor* asociado a un determinado servicio, detecta que tiene varios nodos que están en el nivel más alto de prioridad (tienen una alta calidad de resultados), solicita la eliminación de uno de ellos de forma aleatoria. Este procedimiento puede derivar en varios servidores físicos con un número muy bajo de máquinas virtuales y por lo tanto muchos recursos no asignados. Sin embargo el coste energético asociado a estos equipos es similar a que si alojaran una gran cantidad de nodos virtuales. En este caso, se dice que el sistema está *disperso*.

El procedimiento de compactación es simple y se realiza por el agente especializado *Global Manager* tal y como sigue:

1. Aquellos servidores virtuales con un número de máquinas virtuales muy bajo (el número es determinado de forma supervisada por el administrador del sistema), solicita al resto de servidores que acojan sus máquinas virtuales para que de este modo pueda pasar a un estado de hibernación que no consuma recursos.

2. Los nodos con recursos disponibles al recibir la petición evalúan la instantánea del equipo (I_{ei}^{t}) proporcionada por al agente Local Monitor del equipo, para determinar si pueden hospedar una máquina con unas características como la que se pretende migrar.
3. En el caso de que existan recursos disponibles se envía la configuración al *Global Manager* solicitante y se inicia el proceso de migración. Si éste agente recibiera varias confirmaciones simultáneas iniciaría el proceso de migración a una de las máquinas de forma aleatoria ya que desconoce el conjunto de detalles internos de la máquina que acogerá el nuevo nodo virtual.

Gracias a este procedimiento simple, es posible compactar el conjunto de máquinas virtuales sin alterar la calidad de los servicios, ya que no se modifican los recursos individuales de cada máquina. Con lo que el sistema pasaría a estar en estado *compacto*.

4.4 CONCLUSIONES PRELIMINARES

A lo largo de este capítulo se ha detallado el modelo arquitectónico propuesto en este trabajo de investigación. En primer lugar, se destaca que el modelo se sustenta sobre la caracterización del entorno presentada en el apartado 4.1 y que permite formalizar un sistema CC de propósito general. A partir de ahí, se propone un modelo arquitectónico, denominado +Cloud, para el control y monitorización de una plataforma CC mediante el uso de OV de agentes inteligentes. Gracias a este modelo es posible optimizar la eficiencia del sistema, permitiendo que los agentes alcancen sus objetivos, adaptándose de una manera autónoma pero coordinada a los cambios que se producen en el entorno.

El capítulo finaliza, precisamente, con la descripción de los algoritmos avanzados que se integran en los agentes especializados de la OV. Este modelo de distribución basado en algoritmos sociales hace posible la distribución de responsabilidades entre las diferentes individuos y organizaciones que forman parte de +Cloud, lo que por tanto permite la adaptación dinámica de la sociedad en función de los cambios que se produzcan en el entorno, es decir, teniendo en cuenta la demanda de los servicios ofertados en tiempo de ejecución.

Cabe destacar que es un modelo único ya que incorpora un mecanismo propio de reorganización y adaptación social. Los SMA en un paradigma organizacional y el modelo propuesto añade las funcionalidades de reorganización y adaptación necesarios en este contexto de aplicación.

CAPÍTULO 5
CASO DE ESTUDIO Y RESULTADOS EXPERIMENTALES

En el capítulo anterior se ha presentado un novedoso enfoque para distribuir recursos computacionales de forma adaptativa en entornos CC. El modelo dinámico que se propone hace uso de una arquitectura basada en OV de agentes inteligentes, que permite la integración de algoritmos distribuidos que hacen uso de técnicas de optimización y de razonamiento basado en casos para el reparto de recursos computacionales entre los diferentes servicios de una plataforma CC, construyendo de este modo un modelo adaptativo con la capacidad de aprender a partir de las experiencia pasadas

La evaluación del modelo de distribución dinámica que se propone es una tarea compleja, que requiere disponer de un entorno *hardware* y *software* especialmente adaptado a las necesidades. Para llevar a cabo una evaluación de estas características en el estado del arte existen simuladores de entorno CC que pueden ser utilizados como marco de evaluación (Cloudsim[43] [Calheiros *et al.*, 2011], CloudAnalyst[44] [Wickremasinghe *et al.*, 2010], MDCSim[45] [Lim *et al.*, 2009], GreenCloud [Kliazovich *et al.*, 2012], CDOSim [Fittkau *et al.*, 2012], etc.). Una revisión comparativa de estos simuladores puede encontrarse en [Sandhu *et* Kaur, 2013].

Sin embargo, en el marco de este trabajo la evaluación se ha realizado en una plataforma diseñada y desarrollada en paralelo al modelo CC propuesto en el capítulo anterior. Esta plataforma se ha desarrollado, en el marco de este trabajo de tesis doctoral, por el grupo de investigación BISITE[46] e integra el SMA +Cloud. La plataforma expone diferentes servicios computacionales, a nivel *hardware* y *software*. Desde su nacimiento ha sido concebida para integrar el modelo de monitorización y control detallado en el capítulo anterior. Por tanto, las funcionalidades de reorganización y adaptación de los agentes en su comportamiento son necesarias para el correcto funcionamiento de la plataforma.

[43] A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. http://www.cloudbus.org/cloudsim/
[44] A cloudsim-based visual modeller for analysing cloud computing environments and applications
[45] A Multi-tier Data Center Simulation Platform
[46] http://bisite.usal.es

Dada la experiencia acumulada por el grupo de investigación en esta plataforma, así como los recursos necesarios para la implatación del sistema +Cloud en otras plataformas alternativas, se ha optado por su utilización para la evaluación de la propuesta, dejando como trabajo futuro la comparación de los resultados obtenidos en dicha plataforma con los que se puedan obtener en otras plataformas existentes. Además, el uso de este entorno de evaluación permite una aproximación más realista de las tareas de evaluación que se han llevado acabo en un marco de trabajo familiar para el grupo de investigación, lo que ha permitido realizar una evaluación precisa y realista tanto de la arquitectura multiagente propuesta, como de los algoritmos de distribución de recursos computacionales.

Este capítulo se organiza como sigue: en el apartado siguiente (5.1) se presenta brevemente la plataforma CC que se ha utilizado para la evaluación del modelo propuesto. Una descripción más amplia de esta plataforma CC puede encontrarse en el Anexo B. Posteriormente, que en el apartado 5.2 se presenta la evaluación empírica del modelo de adaptación propuesto mediante la realización de diferentes pruebas de rendimiento. Finalmente, en el último apartado (5.3) se presenta una discusión comparativa entre el sistema propuesto y los resultados obtenidos frente al estado del arte.

5.1 EL ENTORNO DE EVALUACIÓN, LA PLATAFORMA +CLOUD

Este apartado tienen como objetivo describir brevemente los detalles de la plataforma CC que integra el SMA denominado +Cloud descrito en el Capítulo 4. Esta plataforma ha sido el entorno de pruebas que ha permitido evaluar tanto la arquitectura multiagente, como el modelo de razonamiento dinámico y adaptativo para el reparto de recursos computacionales que se ha propuesto en el marco de este trabajo de esta investigación. En el Anexo B de esta memoria se presenta una descripción más completa y detallada de la plataforma, por lo que la información recogida en este apartado tan sólo se centrará en facilitar la comprensión de la plataforma desarrollada, así como la base tecnológica que ha permitido validar el correcto funcionamiento del SMA organizativo y los algoritmos de distribución propuestos.

La plataforma CC con la que se integra el SMA +Cloud a nivel *externo*, aúna un conjunto de servicios que hacen posible el desarrollo y despliegue de aplicaciones web. Para ello dispone de un conjunto de servicios PaaS que hacen posible la persistencia de los datos que manejan las aplicaciones desarrolladas por terceros. Estas aplicaciones pueden estar desplegadas en la propia plataforma CC haciendo uso del servicio de despliegue a nivel IaaS, que proporciona servidores privados virtuales a petición de los desarrolladores/gestores, es decir, el agente *Cloud User*. Finalmente, la plataforma también proporciona un punto de acceso unificado y aplicaciones nativas de tipo *software* que se enmarcan en el nivel SaaS del paradigma tecnológico CC.

Por otro lado, a nivel *interno*, la plataforma está gobernada por el SMA +Cloud que se propone en el marco de este trabajo de tesis doctoral. Este SMA tiene fundamentalmente dos ventajas respecto a otros modelos existentes. En primer lugar, dado que se trata de un SMA abierto, permite que agentes externos (tanto humanos, como artificiales) puedan acceder a la organización jugando un rol concreto, de forma que puedan proporcionar servicios y funcionalidades dentro de la propia organización. Y, en segundo lugar, +Cloud es un sistema integrador que hace posible la aplicación de algoritmos avanzados para la distribución de los recursos entre los servicios de la capa superior e interactuar con el entorno, es decir, los sistemas de virtualización, balanceo de carga y persistencia.

A continuación, en el subapartado 5.1.1 se revisará brevemente los servicios externos que ofrece la plataforma (SaaS, PaaS e IaaS). Posteriormente, en el subapartado 5.1.2 se presentará brevemente el contexto tecnológico de la

plataforma y la interacción con el SMA. El diseño interno de la misma ya se ha presentado, en detalle, en el Capítulo 4 del presente documento.

5.1.1 SERVICIOS EXTERNOS

La plataforma CC diseñada propone un modelo de computación que abarca los tres niveles de servicios (capacidades) que propone el NIST en su definición (*software*, plataforma e infraestructura) [Mell *et* Grance, 2011]. En la Figura 30 se presenta una vista global de los servicios ofertados, los cuáles se presentarán en los siguientes subapartados.

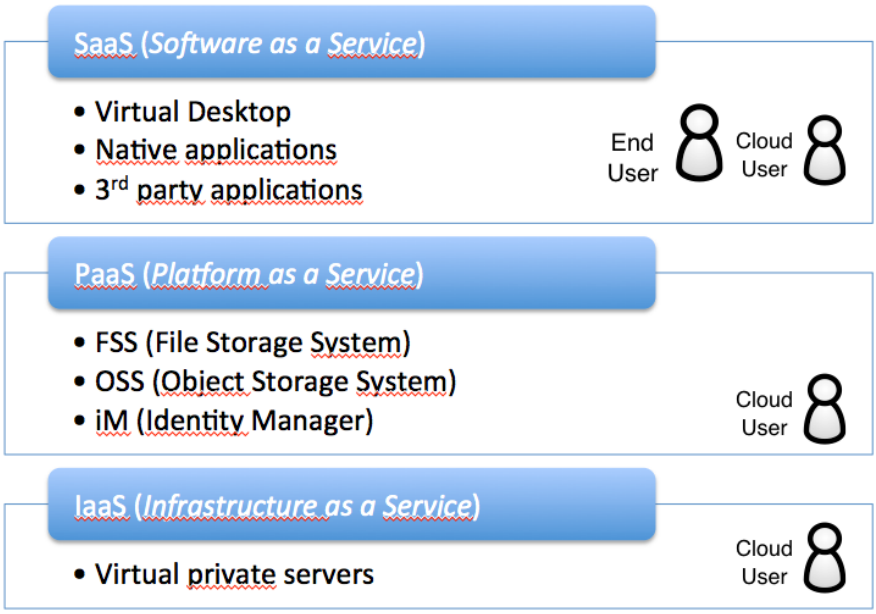

Figura 30.- Servicios externos de la plataforma +Cloud

5.1.1.1 EL NIVEL SaaS

Las capacidades que proporciona la plataforma a nivel *software* están formadas por las aplicaciones web a las que los usuarios finales tienen acceso. Este nivel se ha diseñado para utilizar dos tipos de aplicaciones *software*, las aplicaciones nativas de la plataforma y las aplicaciones desplegadas por terceros.

En primer lugar, se presentan las aplicaciones web nativas que proporciona la plataforma para la administración del propio entorno CC, así como para facilitar el trabajo de los usuarios que hacen uso de la plataforma (*Cloud User*, *End User* y *Administrator User*). En el desarrollo de estas aplicaciones se han empleado las últimas tecnologías en el marco del desarrollo web. De este

modo es posible ofrecer al usuario un entorno de uso avanzado. Así, se han utilizado las últimas novedades ofrecidas por HTML5[47] y CSS3[48], además del novedoso protocolo de comunicación WebSockets [49] que hace posible actualizar los contenidos visualizados por el cliente sin necesidad de recargar páginas o realizar sondeos periódicos. También se utiliza intensivamente bibliotecas como jQuery [50] que facilitan la manipulación del árbol del documento estático de las páginas web, y finalmente, también se integra la tecnología AJAX [51] que garantiza la compatibilidad del código entre navegadores.

A continuación se introducen las principales aplicaciones nativas que se han desarrollando en en el marco de la plataforma CC:

- Un **escritorio virtual**, que es una aplicación web que emula un escritorio similar al de cualquier sistema operativo, así es posible interaccionar con la plataforma de forma semejante a como se haría a través de un escritorio tradicional. Su utilidad es la de servir como punto de acceso unificado a las aplicaciones y funcionalidades disponibles en +Cloud de forma individualizada para cada usuario.
- Una aplicación de **gestión personal,** que incluye herramientas para gestionar la información relativa a cada usuario de forma individualizada incluyendo aplicaciones, gestión de servicios contratados, perfil de acceso, datos personales, etc. Así, en función del rol que cada usuario tiene en la plataforma, tendrá acceso a diferentes funcionalidades. En este sentido, se contempla la existencia de tres roles (Administrador, Desarrollador/Manager y Usuario), de forma que un usuario (real) puede tener asignados varios roles al mismo tiempo dentro de la plataforma. Como se puede apreciar, estos roles están alineados con los roles externos *Cloud Admin*, *Cloud User* y *End User* de la organización virtual principal +Coud.
- Un **catálogo de aplicaciones,** que básicamente es un *market* según la analogía con los mercados tradicionales ampliamente utilizada en las plataformas móviles [Holzer *et* Ondrus, 2011]. A través de esta aplicación es posible consultar información sobre las aplicaciones que se ofertan por terceros, así como adquirirlas para su uso. En función de la configuración propuesta por los desarrolladores de cada aplicación, el usuario para adquirirla deberá aceptar una licencia de uso, después de lo

[47] http://www.w3.org/html/
[48] http://www.w3.org/Style/CSS/
[49] https://tools.ietf.org/html/rfc6455
[50] http://jquery.com/
[51] http://www.w3.org/TR/XMLHttpRequest/

cuál se le asignará un rol específico dentro de la aplicación. El conjunto de roles de cada aplicación puede ser configurado dinámicamente por el *manager*/desarrollador de la misma.

- El **panel de control de la infraestructura** que permite monitorizar y gestionar la infraestructura subyacente, así como los servicios desplegados. Está disponible para el rol *Cloud Admin* principalmente. Incluye dos vistas principales:
  - La *monitorización y control de infraestructura* que hace posible gestionar el estado de la infraestructura interna, distribuyendo sus recursos entre los servicios ofertados a los usuarios.
  - La *vista de servicios* que permite visualizar el rendimiento de cada servicio desplegado en la plataforma.

Además, también existen las aplicaciones desarrolladas por terceros, las cuáles se encuentran desplegadas en el entorno CC y que, por tanto, también se ofrecen como servicio a los usuarios finales, en este caso el rol *End User*. Este segundo grupo de aplicaciones, constituye la razón de existencia de la plataforma CC, y por tanto del SMA +Cloud, según se ha descrito en la vista funcional del modelo de organización (Figura 60, Capítulo 4) de la metodología GORMAS, ya que esta plataforma CC es de propósito general y, por lo tanto, está orientada a proporcionar servicios a terceros.

Para que todas las aplicaciones (nativas y de terceros) sean realmente elásticas, deben utilizar los servicios de la capa inferior de la plataforma para satisfacer sus necesidades de persistencia de información. Gracias a este modelo arquitectónico es posible asegurar la elasticidad de las aplicaciones desplegadas en la plataforma, ya que hace posible efectuar el balanceo de las peticiones y, por lo tanto, el reparto de la carga entre las distintas réplicas de un servicio sin que se pierda información o se produzcan incongruencias en la información disponible. En el siguiente apartado se detallan los servicios de persistencia de información que se enmarcan en la capa PaaS de la plataforma CC.

5.1.1.2 EL NIVEL PaaS

Los componentes de esta capa están orientados a hacer posible la persistencia de la información de las aplicaciones *software* desplegadas en la plataforma, lo que hace posible la elasticidad de las mismas. Gracias al modelo arquitectónico, las aplicaciones pueden desplegarse en máquinas virtuales sin estado, las cuáles pueden ser instanciadas y paradas en función de la demanda de los usuarios.

En esta capa se dispone de un conjunto de componentes cuya funcionalidad se expone en términos de un conjunto de APIs basadas en servicios web sin

estado, implementadas mediante el protocolo REST[52] de servicios web. A continuación se presenta una descripción de los principales componentes:

- El sistema de **gestión de la identidad** que permite validar las credenciales de usuarios y aplicaciones que hacen uso del sistema CC. Para ello, se incluyen dos servicios principalmente:
  - *Autenticación de usuarios* que permite una gestión de la identidad, unificando a todas las aplicaciones y servicios del sistema CC. Para ello, implementa un módulo que proporciona una autenticación de tipo *Single Sign On* [Hursti, 1997]. De este modo, el usuario puede autentificarse una única vez con independencia de la aplicación (nativa o no) que esté utilizando.
  - *Autenticación de aplicaciones* que valida las credenciales de las aplicaciones que han hecho uso de servicios desplegados dentro del sistema CC. Para ello, a los desarrolladores de las aplicaciones se les proporciona un identificador y una clave para cada aplicación. Posteriormente, estas credenciales deben adjuntarse a cada petición lo que permite validar su autenticidad.
- El sistema de **gestión de persistencia de información** hace posible la persistencia de los datos en cada una de las aplicaciones. Para ello, expone dos servicios en función del tipo de información que sea necesario almacenar:
  - S*ervicio de almacenamiento de ficheros,* que proporciona una interfaz en términos de servicios web a un contenedor de ficheros que emula una estructura de directorio de un sistema de ficheros tradicional. Gracias a este servicio, las aplicaciones pueden guardar y recuperar ficheros sin conocimiento acerca de dónde se encuentran físicamente almacenados.

    Este servicio también incluye otras características como un mecanismo de control de versiones, el almacenamiento de metadatos asociados al fichero y la gestión eficiente de subidas y descargas de ficheros grandes.
  - *Servicios de almacenamiento de información,* proporciona un conjunto de operaciones básicas (creación, eliminación, actualización y recuperación) sobre una base de datos orientada a documentos [Han *et al.*, 2011] que es simple y flexible. Así las aplicaciones tienen acceso a un conjunto de servicios web que abstraen la tecnología concreta de persistencia, es decir, la base de datos que se utiliza finalmente para la persistencia.

[52] http://www.w3.org/TR/ws-arch/

Para la implementación de estos componentes se ha utilizado el lenguaje de programación Python[53] debido a su potencia, facilidad de mantenimiento y la flexibilidad para el manejo de estructuras de datos. En este sentido, el formato de intercambio de datos está basado en JSON[54] (*JavaScript Object Notation*). La capa de servicios web se ha implementado mediante el uso del *framework* de desarrollo web Tornado[55], por su capacidad de gestionar un gran número de conexiones de clientes.

5.1.1.3 EL NIVEL IaaS

La plataforma también ofrece un servicio de infraestructura, aunque mucho más limitado que los servicios proporcionados en las capas anteriormente descritas (SaaS y PaaS). Así la plataforma proporciona un servicio de infraestructura de tipo VPS (*Virtual Private Server*) [Beloussov *et al.*, 2008] orientado al despliegue de las aplicaciones de terceros que se ofrecen como servicio en el nivel SaaS. Sin embargo, los VPS proporcionados en este nivel no podrán ser utilizados en ningún caso para otros fines como puede ser la computación de alto rendimiento o la persistencia de grandes volúmenes de información.

Aunque las capacidades de este nivel son bastante limitadas, el usuario puede seleccionar el tipo de máquina virtual que necesita. Estas máquinas virtuales posteriormente serán instanciadas por el sistema de virtualización subyacente a partir de una plantilla previamente creada. Gracias a este modelo del servicio de infraestructura, la plataforma permite desplegar las aplicaciones del usuario en la propia infraestructura de la plataforma lo que facilita la integración en su ecosistema CC desarrollado (aplicaciones nativas, escritorio virtual, etc.) y, también, reduce el tiempo de latencia en el acceso a los servicios de la capa PaaS.

5.1.2 EL ENTORNO TECNOLÓGICO DE LA PLATAFORMA

En el apartado 4.1 del Capítulo 4 ya se ha caracterizado en profundidad un entorno CC de forma general. Este apartado, por tanto, se centra en describir en líneas generales el entorno tecnológico de la plataforma CC que integra el sistema +Cloud. En la Figura 31 se presenta a muy alto nivel los estratos de cualquier plataforma CC que, como se puede apreciar se subdividen en dos grandes grupos, la capa de despliegue *software* y la capa de infraestructura interna *hardware*. En primer lugar, la capa de despliegue *software* incluye los

[53] https://www.python.org/
[54] http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[55] http://www.tornadoweb.org/en/stable/

servicios en los niveles SaaS y PaaS, así mismo, esta capa también incluye las bases de datos que hacen posible la persistencia de la información del entorno tecnológico. Por su parte, la infraestructura *hardware* interna incluye tanto los componentes físicos, como la capa de virtualización que hace posible la gestión dinámica de los recursos disponibles en este estrato.

Este complejo entorno tecnológico dentro de la plataforma es monitorizado y controlado por el SMA +Cloud que se ha presentado en el 4.2 del Capítulo 4. Esta plataforma que está basada en organizaciones virtuales, permite la integración de modelos de razonamiento avanzado que hace posible la distribución de recursos (físicos y virtuales) entre los servicios ofertados al usuario por la plataforma CC. Para interaccionar con el entorno hace uso de puertos, según se describe en la metodología GORMAS. Gracias a estos puertos es posible abstraer el complejo entorno subyacente de los agentes individuales que realizan la toma de decisiones, lo que permite independizar estos algoritmos y modelos desarrollados de la tecnología concreta que se utilice en el nivel de entorno.

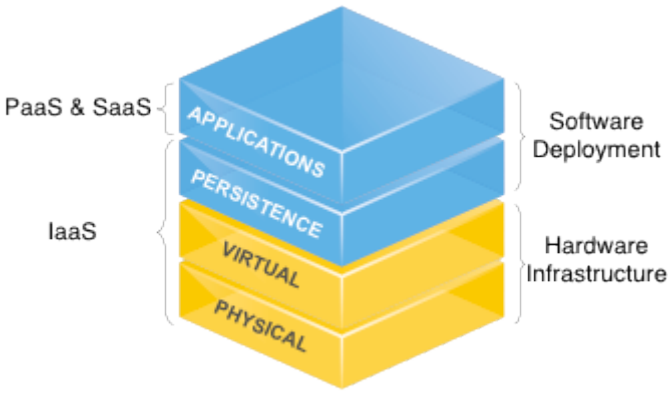

Figura 31.- Estratos a nivel tecnológico de un entorno Cloud Computing

A lo largo de los siguientes subapartados se presentarán los principales rasgos tecnológicos de los componentes internos que hacen posible ofrecer los servicios externos elásticos a la plataforma CC. Así, en el apartado 5.1.2.1 se presentará el modelo y tecnología empleada para la persistencia de la información, tanto de objetos como de ficheros. A continuación, en el apartado 5.1.2.2 se presentará los detalles acerca del sistema de balanceo de carga. Posteriormente, el apartado 5.1.2.3 incluye una descripción del sistema de comunicación que hace posible la interacción entre los agentes del sistema +Cloud, integrado en la plataforma. Finalmente, el último apartado de la sección (5.1.2.4) se presenta la tecnología de virtualización que emplea la plataforma CC, así como una breve descripción de cómo se realiza la interacción con el mismo.

5.1.2.1 LA CAPA DE PERSISTENCIA DE INFORMACIÓN

En esta capa se enmarcan tanto la persistencia de información en bases de datos, como el almacenamiento de ficheros en un sistema de ficheros distribuido. Su cometido es hacer frente a la demanda de almacenamiento en un entorno CC de altas prestaciones. A continuación se presentan detalles sobre estos dos componentes que hacen posible la persistencia en la plataforma CC:

- **Persistencia de información de bases de datos**. Las limitaciones y rigidez del modelo relacional hacen que sea complicado su escalabilidad en entornos distribuidos y, además, dificulta el mantenimiento evolutivo de las bases de datos construidas sobre este modelo de almacenamiento. Para solucionar estos problemas, en los últimos años han aparecido una gran cantidad de propuestas de modelos alternativos al relacional [Han *et al.*, 2011], de los cuáles, el que más éxito ha tenido es el modelo de almacenamiento orientado a documentos.
  La persistencia de objetos en la plataforma CC se basa en la utilización de las bases de datos orientadas a documentos denominada MongoDB[56], que es un sistema gestor de bases de datos de código abierto orientado a documentos, distribuido y escalable que se adapta perfectamente a las necesidades perseguidas en el marco de la plataforma desarrollada.
- **Persistencia de ficheros.** En un sistema CC, en general, la persistencia de ficheros debe realizarse en entornos distribuidos que permitan la replicación y partición de los datos [Clifford *et al.*, 2007], habitualmente se utilizan sistemas SAN (*Storage Area Network*).
  El sistema SAN de la plataforma es altamente escalable, ya que se ha probado y validado su correcto funcionamiento con dos sistemas diferentes. Así, se ha utilizado tanto el sistema tradicional de ficheros distribuido NFS (*Network File System*)[57], como del moderno sistema GlusterFS[58] que ofrece mejores características en cuanto a escalabilidad y mayores posibilidades de configuración y readaptación dinámicas.

5.1.2.2 BALANCEO DE CARGA EN LOS SERVICIOS

Un *proxy* inverso es un componente *hardware* o *software* que recibe peticiones de clientes y las delega a otros nodos que son los proveedores reales del servicio. Por su parte, los balanceadores de carga pueden ser considerados un tipo especial de *proxy* con la habilidad de repartir las

[56] https://www.mongodb.org/
[57] http://nfs.sourceforge.net/
[58] http://www.gluster.org/

peticiones de forma homogénea entre diferentes nodos en tiempo real y, además, permitiendo otras funcionalidades como la modificación de las peticiones, la encriptación y desencriptación, etc.

En la plataforma, como implementación del *proxy* y balanceador de carga se ha optado por Nginx[59] que es un proyecto de *software* libre con un amplio espectro de utilización, que se adapta perfectamente a las necesidades que se perseguían al inicio del trabajo de investigación. La interacción con el sistema de balanceo de carga es realizada por los agentes especializados *Service Monitor* y *Service Supervisor*, los cuáles utilizan un puerto del entorno para encapsular los mensajes correspondientes a la interacción. Este puerto es capaz de procesar en tiempo real el archivo de registro para determinar el estado del balanceado. Al mismo tiempo, también es capaz de variar dinámicamente en tiempo de ejecución la configuración y los pesos entre los diferentes nodos a balancear. Para el balanceo de carga se ha optado por la variación del algoritmo *Round-Robin* que se ha descrito en el 4.3.2.1 del Capítulo 4.

5.1.2.3 COMUNICACIÓN INTERNA DE LA PLATAFORMA

Dentro de la plataforma CC, el sistema de comunicación está formado por una plataforma de mensajería de alta disponibilidad basada en colas de mensajes e implementada a través del sistema RabbitMQ[60], que está basado en el protocolo AMQP[61] (*Advanced Message Queuing Protocol*). Gracias a este modelo es posible la comunicación entre los agentes el SMA +Cloud. Este componente ofrece el soporte necesario para que los diferentes agentes que forman parte de las organizaciones del sistema ofrezcan y descubran servicios. En este sentido, este componente proporciona un mecanismo mediante el cual las entidades autónomas pueden registrar la descripción de servicios como entradas en un directorio y suscribirse a los canales de comunicación que ofrecen el resto de integrantes del SMA.

Dentro del sistema de paso de mensajes basado en colas, se destaca que se ha hecho posible la alta disponibilidad del mismo ya que está replicado en varios servidores de la propia infraestructura del entorno CC. Para ello se ha hecho uso de las bondades que ofrece RabbitMQ. El rol *Global Supervisor* del SMA será el encargado de comprobar que este módulo está funcionando correctamente en todo momento. Si no lo estuviera, se instancia una réplica del servicio en un nuevo servidor físico y se determinarán las causas que produjo la no disponibilidad del mismo.

[59] http://nginx.org/
[60] https://www.rabbitmq.com/
[61] http://www.amqp.org/

5.1.2.4 EL ENTORNO DE VIRTUALIZACIÓN

La virtualización [Barham *et al.*, 2003] es uno de los denominadores comunes de las plataformas CC. Consiste, a grandes rasgos, en la emulación de un entorno *hardware* para un sistema operativo, en este caso virtualizado. De forma que el sistema operativo, en general, no conoce que está siendo ejecutado en una instancia virtual, denominada máquina virtual. Gracias a la virtualización, es posible que en un único servidor físico (anfitrión), pueden existir múltiples servidores virtuales (huéspedes), cada uno de ellos con su propia identidad, compartiendo de manera transparente los recursos del servidor físico.

Ya se describió en detalle en el apartado 4.1.2.1 del Capítulo 4 el modelo de virtualización empleado en este trabajo de tesis doctoral. En cuanto al entorno tecnológico utilizado en la plataforma CC, se ha optado por el uso de entornos con licencias libres, utilizando dos plataformas como son OpenVZ[62] y KVM[63]. En una primera fase del desarrollo se utilizó el sistema de virtualización OpenVZ que está basado en contenedores y, por lo tanto, restringe su funcionamiento a sistemas Linux. Sin embargo, debido al débil aislamiento de OpenVZ, en una segunda fase se migró a KVM que es un sistema de virtualización basado en Qemu[64] [Bellard, 2005]. KVM también permite la migración en vivo de máquinas virtuales entre sistemas anfitrión, pero mediante el uso de un sistema de almacenamiento compartido. La asignación de memoria RAM es menos rígida que en otros entornos de virtualización y funciona correctamente con versiones estables (2.6) del *kernel* de Linux. Se ha desarrollado un puerto del entorno, basado en la librería Libvirt[65], que es utilizado por los agentes especializados *Local Monitor* y *Local Manager* para la monitorización y asignación dinámica de recursos en este tipo de entorno distribuido.

5.1.3 CONCLUSIÓN

A lo largo de este apartado se ha descrito brevemente la plataforma CC que se ha desarrollado en el marco de este trabajo de investigación. La plataforma ofrece un conjunto de servicios externos a nivel *software*, plataforma e infraestructura. Para proporcionar este conjunto de servicios, internamente dispone de un conjunto de componentes que hace posible la virtualización del *hardware*, el balanceo de carga y la comunicación entre componentes.

[62] http://openvz.org/
[63] http://www.linux-kvm.org/
[64] http://wiki.qemu.org/
[65] http://libvirt.org/

Esta plataforma es gobernada por el SMA +Cloud, que se integra con la propia plataforma CC. El sistema +Cloud dispone de un conjunto de agentes que se sitúan estratégicamente en diferentes puntos de la plataforma CC para, en primer lugar, monitorizar el estado de la plataforma y de los servicios que se ofrecen. Posteriormente, a partir de estos datos de monitorización el sistema +Cloud es el encargado de realizar las tareas que sean necesarias para hacer frente a las necesidades computacionales de los servicios que se ofertan, haciendo posible que estos servicios sean elásticos. Estas tareas pasan por la ejecución de los algoritmos avanzados que disponen los agentes especializados de +Cloud y que dan lugar a la autoadaptación del sistema +Cloud, y por tanto, de la propia plataforma CC con la que se integra.

Para evaluar el correcto funcionamiento del sistema y de los algoritmos de adaptación se ha diseñado un caso de estudio, en el que se incluyen diferentes experimentos sobre la plataforma CC y el sistema +Cloud. El siguiente apartado (5.2) incluye los resultados obtenidos durante la realización de este caso de estudio específico.

5.2 CASO DE ESTUDIO

Para evaluar la arquitectura multiagente propuesta en este trabajo de tesis doctoral se han realizado una serie de experimentos que se describen en esta sección. En los casos de evaluación llevados a cabo se ha hecho uso de la plataforma CC descrita en el apartado anterior, que integra el sistema adaptativo +Cloud propuesto en el Capítulo 4 y que es el centro de los experimentos de evaluación. Las capacidades de reorganización y adaptación de los agentes que forman parte de +Cloud son necesarias en esta plataforma CC para hacer frente a la demanda de peticiones en los servicios que se ofertan a los usuarios. A continuación, se detalla el caso de estudio que se ha diseñado para validar el modelo arquitectónico propuesto, así como el proceso seguido por los algoritmos de razonamiento, que integran los agentes especializados en la distribución de recursos computacionales en entornos distribuidos.

En primer lugar, se enuncia el objetivo principal del caso de estudio, que en este caso es doble: en primer lugar, la simulación del comportamiento de los miembros de la unidad organizativa principal y subunidades que forman el SMA, en un caso de adaptación real desplegados sobre la plataforma CC. Posteriormente, a partir de la validación del correcto funcionamiento de la organización dentro de esta simulación en el que se incluyen los principales elementos del SMA +Cloud, se procederá a evaluar el comportamiento de los modelos de razonamiento que hacen posible la adaptación dinámica del SMA organizativo mediante la distribución de los recursos de la infraestructura del plataforma CC entre los diferentes servicios ofertados en función de la demanda observada.

Para este caso de estudio, la plataforma CC, junto con el SMA +Cloud que la gobierna, se ha desplegado en el entorno HPC del grupo de investigación BISITE, formado por 15 equipos de última generación que permiten la virtualización por hardware a través de la tecnología Intel-VT y del sistema de virtualización KVM. El sistema operativo tanto de las máquinas físicas, como de las instancias virtuales es la distribución de Linux CentOS[66] (*Community ENTerprise Operating System*) que es la variante libre de la conocida distribución Red Hat[67].

La descripción de cada una de las subcapas según la caracterización del entorno propuesto en el apartado 4.1.2 del Capítulo 4 es la siguiente:

[66] https://www.centos.org/
[67] http://www.redhat.com/

- La *subcapa de ejecución*, que en el caso de estudio ha estado formada por los procesadores y memoria de 10 equipos físicos, cada uno de ellos con 16Gb de memoria RAM y un procesador de cuatro núcleos que permite la ejecución de 8 hilos de forma simultánea, es decir, se disponen de 8 procesadores virtuales (*vcpu*). Por lo tanto, cada máquina admite un máximo de 8 máquinas virtuales, cada una de ellas con un núcleo de ejecución. Si alguna máquina necesita un número de *vcpus* mayor según su plantilla de servicio, el número de máquinas que podría hospedar el equipo disminuiría en proporción.
- La *subcapa de persistencia* está formada por 4 equipos que conforman un entorno SAN construido mediante la plataforma GlusterFS. Esta capa tiene como objetivo almacenar los discos duros de las máquinas virtuales en ejecución, las plantillas de las máquinas virtuales de los servicios y, finalmente, los datos de las aplicaciones/usuarios gestionados mediante cuotas.

Finalmente, el equipo restante integra de forma única y no replicada (por simplicidad y dado que no se evalúa la disponibilidad de la plataforma) los repositorios que utilizan los diferentes agentes de +Cloud y el sistema de comunicación basado en RabbitMQ. Todos los nodos de ambas capas están unidos mediante un equipamiento de red (*switch*) que permite la comunicación a alta velocidad (*Gigabit*).

5.2.1 DESCRIPCIÓN DEL ESTADO INICIAL

El caso de estudio se basa en la realización de un ataque de tipo denegación de servicio (*Denial of Service*, DoS) [Needham, 1993] sobre una de las APIs que expone la plataforma CC en la capa PaaS, concretamente, sobre el servicio de persistencia de ficheros. Este servicio se presentará detalladamente en el apartado B.1.2.2 del Anexo B.

De todos los servicios web de tipo REST que expone este componente de cara a los usuarios del sistema CC, se seleccionaron los dos siguientes:

- **GetSize**. Obtiene el tamaño en *bytes* de un fichero que se le suministra como parámetro, o de la suma de tamaños del contenido de un directorio.
- **GetFolderContent**. Recupera el contenido de un directorio que se le suministra como parámetro.

El método *GetSize* es una función compleja que utiliza recursividad para calcular la suma de los tamaños de los ficheros contenidos en un directorio. Por su parte, el método *GetFolderContent*, es una función mucho más sencilla que tan sólo devuelve los identificadores de los ficheros o directorios contenidos en la ruta suministrada como parámetro.

En el transcurso de este caso de estudio se procedió a evaluar, a través de una serie de experimentos, en primer lugar la redistribución de la infraestructura a nivel micro, y posteriormente, la distribución de infraestructura a nivel macro. En cuanto a la distribución de recursos a nivel servicio, ésta queda implícitamente evaluada en cada uno de los experimentos que se van a presentar a continuación debido a que este algoritmo es ejecutado por el agente *Service Monitor* asociado al servicio de forma continuada para mantener establecido el nivel de QoS asociado a cada uno de los servicios ofertados por la plataforma. Así, como se pudo observar en los experimentos que se presentan a continuación, una vez que se produce una ejecución de los algoritmos de readaptación en el nivel de infraestructura, el modelo de adaptación de recursos a nivel servicio tiende a equilibrar los pesos de los nodos que proporcionan los servicios para que el nivel de QoS permanezca constante de cara al usuario final. De este modo, el agente *Cloud User* que accede a los servicios tiene una visión de uniformidad en el acceso a cada uno de los productos que se ofertan a través de la plataforma CC, pese a que la adaptación se produce de forma interna en la plataforma.

Todos los experimentos que se presentan a continuación parten, de un mismo estado inicial. En este sentido, se opta por un punto de partida en el que los experimentos que se han realizado durante el caso de estudio se aproximen en gran medida a las condiciones de un despliegue real, en el que la la plataforma CC estuviera en producción sometida a peticiones de usuarios reales. Por tanto, este punto de partida, además de ser didáctico y fácil de comprender, refleja un despliegue habitual de cualquier servicio en un entorno CC.

En cuanto a la distribución de recursos inicial, el servicio de almacenamiento de ficheros está desplegado en dos nodos diferentes (VM1 y VM2), cada uno de ellos hospedado por una máquina física diferente (PR1 y PR2, respectivamente). Gracias a este despliegue el servicio tiene alta disponibilidad (está desplegado en dos servidores), pero al mismo tiempo está situado en máquinas físicas con diferente carga computacional, tal y como sucede en un entorno real, ya que ambas máquinas físicas hospedan otras máquinas virtuales correspondientes a otros servicios de la plataforma CC. En este sentido, el servidor físico PR1 tiene recursos libres, sin asignar. Mientras que el servidor PR2 no tiene recursos libres y las máquinas que aloja, están sometidas a carga computacional alta. En la Figura 32 se representa gráficamente el estado de partida de los casos de estudio que se han realizado. Así mismo, también se puede observar los principales agentes que intervienen en la proceso de readaptación dentro del caso de estudio.

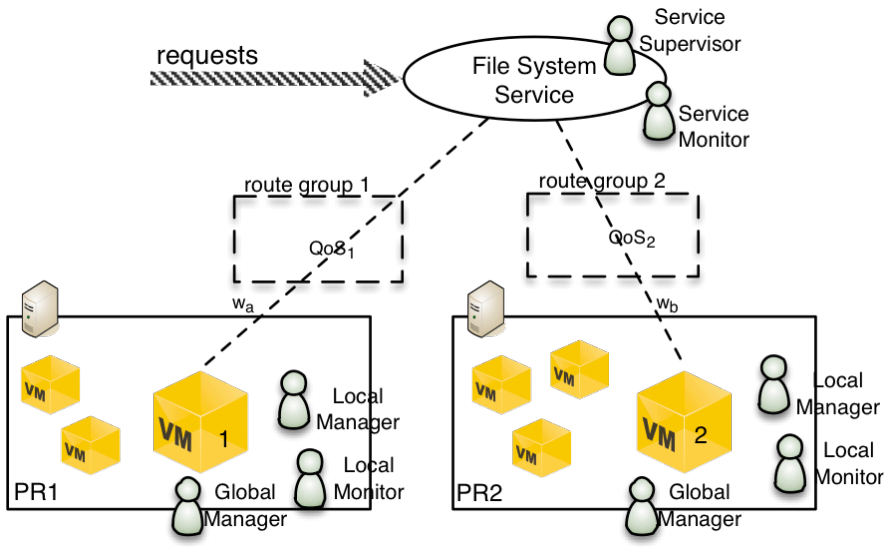

Figura 32.- Estado inicial del caso de estudio de evaluación

Los experimentos que se han diseñado comienzan, realizando peticiones a los dos métodos anteriormente indicados (*GetSize* y *GetFolderContent*) del componente que permite la persistencia de ficheros. El número de peticiones se aumenta progresivamente, aumentando por tanto paulativamente la demanda del servicio y las necesidades computacionales. Debido a este aumento, llega un momento en el que se hace necesaria la readaptación del sistema para hacer frente al creciente número de peticiones.

Destacar en último lugar, que los algoritmos de readaptación en los experimentos que se presentan a continuación, se han ejecutado en el momento en que se detecta un bajo nivel de QoS. En los experimentos se sigue este modelo ya que de cara a la presentación de los resultados se obtiene una vista más clara, simple y ordenada del funcionamiento de los algoritmos. No obstante, en caso de que la plataforma CC estuviera en una fase de explotación con usuarios reales, el proceso de readaptación se debería ejecutar cuando se observa un estado continuado de bajo QoS en alguno de los servicios.

5.2.2 EVALUACIÓN DEL MODELO DE ADAPTACIÓN DE INFRAESTRUCTURA A NIVEL MICRO

Para evaluar la readaptación de infraestructura a nivel micro, se realiza el primer tipo de experimentos en el que se lanza progresivamente, cada segundo, un hilo de ejecución hasta un máximo de 10 segundos (10 hilos). En este caso, cada hilo continuamente consulta el servicio web *GetSize* del servicio de almacenamiento de ficheros. Este método, como se ha indicado

previamente es complejo, ya que realiza consultas internas recursivas para conocer el tamaño del objeto (fichero o carpeta). En el experimento se considera que la QoS del servicio es adecuada cuando el tiempo de respuesta de las peticiones no supera los 2,5 segundos. Por lo tanto, en caso de que se supere este tiempo de respuesta, sería necesaria una readaptación del sistema.

El resultado del experimentado se presenta en la Figura 33 que muestra una gráfica con los tiempos de respuesta del método *GetSize* durante el tiempo que ha durado la prueba. Así pues, el sistema una vez que detecta que se ha reducido el nivel de QoS en el servicio, es decir, cuando los tiempos de respuesta medios son superiores a 2,5 segundos, inicia de forma automática un proceso de adaptación de infraestructura a nivel micro. Este proceso es lanzado por el agente *Service Supervisor*, en función de la información que le proporciona el agente *Service Monitor* acerca del estado del servicio. Estos dos agentes especializados están asociados únicamente al servicio que se considera.

Como se puede observar en la Figura 33, una vez completado este proceso de auto-adaptación y ajustados los pesos de balanceo, el tiempo de respuesta del servicio vuelven a situarse por debajo de los de los niveles de QoS considerados como aceptables (por debajo del límite de los 2,5 segundos).

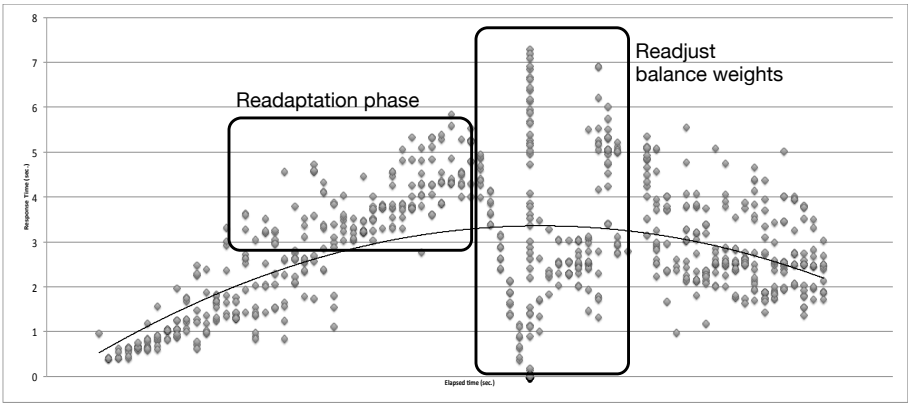

Figura 33.- Experimento 1: Reajuste de recursos de infraestructura a nivel micro (Método: *GetSize*)

A continuación se detalla el proceso que se ha llevado a cabo durante la distribución de recursos de infraestructura a nivel micro. El algoritmo comienza cuando el agente *Service Monitor* detecta que el nivel de QoS asociado al servicio se ha degradado progresivamente superando el mínimo tiempo de respuesta que se considera aceptable (2,5 segundos).

El agente *Service Supervisor*, a partir de estos datos sobre la QoS del servicio, decide iniciar un proceso de redistribución de recursos de

infraestructura a nivel micro, el proceso completo se presenta en la Figura 34. En primer lugar, este agente *Service Supervisor* envía un mensaje (Paso 1, Figura 34) a cada uno de los agentes *Local Manager* de las máquinas físicas (PR1 y PR2) que alojan los nodos trabajadores del servicio para indicarles que el servicio de persistencia de ficheros (FSS) necesita un mayor número de recursos. Junto a este mensaje también se le indica el peso que tiene cada nodo en el proceso de balanceo de peticiones. A partir de esta información cada agente *Local Manager*, de forma independiente, determina la cantidad de recursos adicionales que puede dedicarle al servicio. Para ello, el agente *Local Manager* de cada máquina solicita a su agente homólogo *Local Monitor*, la información acerca del estado de cada máquina, es decir, la instantánea I_{FR}^{t} que caracteriza al equipo físico.

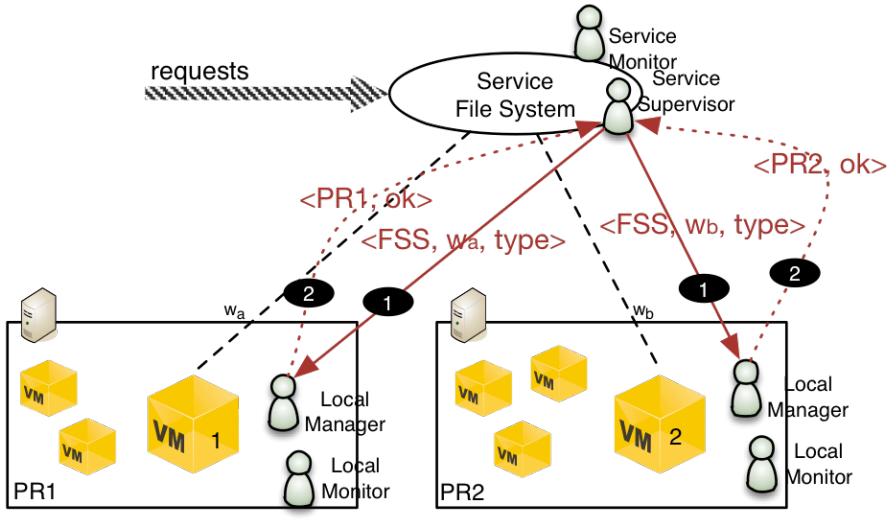

Figura 34.- Experimento 1: Intercambio de mensajes en la distribución de infraestructura a nivel micro

Una vez que el agente *Local Manager* ha recuperado toda la información local que utiliza para llevar a cabo el proceso de toma de decisión, el siguiente paso consiste en determinar mediante su proceso de razonamiento interno la cantidad de recursos (*vcpu* y memoria) que puede provisionar para el nodo que hospeda, el cuál demanda una cantidad mayor de recursos para hacer frente a la demanda existente. Concrétamente, en el experimento se han dado las dos opciones posibles atendiendo a la cantidad de recursos libres de cada uno de los servidores que hospedan los nodos virtuales:

- En el caso del servidor PR1, como dispone de recursos libres que no están asignados a ningún otro nodo, la asignación se produce de forma directa. Así pues se le asigna una *vcpu* adicional al nodo que demanda recursos, y más memoria en función de su prioridad dentro del sistema

de balanceo, según la expresión que se detalló en el apartado 4.3.2.2 del Capítulo 4.

- En el caso del servidor PR2 hospeda un número mayor de máquinas virtuales todas ellas con una mayor carga computacional, las cuáles están asociadas a otros servicios de la plataforma. Por lo tanto, no dispone de recursos libres, por lo que evalúa si el conjunto de nodos que hospeda tiene recursos infrautilizados:
  - En cuanto a la evaluación de unidades de cómputo (*vcpu*), el servidor PR2 no dispone de ninguno libre. Ya que de sus 8 *vcpus*, todos ellos están asignados a otras máquinas virtuales que hospeda en el momento en el que se ha realizado experimento, así como al servidor físico. Sin embargo detecta que una de las máquinas virtuales que tiene asignados 2 *vcpus* está infrautilizando uno de ellos, por lo que decide retirar este recurso de computación a esta máquina y asignárselo al nodo virtual del servicio de almacenamiento de ficheros que ha demandado recursos.
  - En cuanto a la provisión de nueva memoria tampoco dispone de recursos libres, por lo que determina la memoria infrautilizada por las máquinas existentes mediante el proceso de optimización basado en programación lineal. Después de realizar esta evaluación el modelo de optimización determina que no es conveniente reasignar nuevos recursos, ya que no existen recursos infrautilizados, por lo que no se provisiona memoria para el nodo virtual que demanda recursos.

  Después del proceso de razonamiento, el agente *Local Manager* en la máquina PR2 le asigna al nodo del servicio FSS mayor cantidad de recursos en términos de unidades de cómputo (*vcpus*), pero no así en términos de memoria.

Una vez finalizado este proceso de razonamiento individualizado en cada servidor físico, ambos agentes *Local Manager* situados en PR1 y Pr2 notifican (Paso 2, Figura 34) de forma individualizada al agente *Service Supervisor* el resultado del proceso de provisionamiento de servicios. Una vez se reciben las notificaciones, el agente *Service Supervisor* no realiza ninguna acción adicional, ya que al menos uno de los nodos físicos ha provisionado más recursos para el servicio FSS en su conjunto (en este caso ambos nodos físicos). Por su parte, el agente *Service Monitor*, realiza el equilibrado de pesos para ajustar la demanda a las capacidades de los nodos atendiendo a los nuevos recursos. En función de este ajuste en los pesos de cada nodo, si se reduce el tiempo de respuesta medio del servicio (situándolo por debajo del umbral de los 2,5 segundos, como así ha sucedido), el agente *Service Supevisor* no realizará ninguna acción más. Sin embargo, en el caso, de que el problema persistiera se realizaría un proceso de adaptación a nivel macro.

5.2.3 EVALUACIÓN DEL MODELO DE ADAPTACIÓN DE INFRAESTRUCTURA A NIVEL MACRO

Una vez que se ha presentado la distribución de recursos de infraestructura de recursos a nivel micro, a continuación en el segundo tipo de experimientos se procede a evaluar el nivel macro. En este segundo tipo de experimentos, por claridad sólo se evalúa la adaptación de infraestructura a nivel macro, sin considerar de forma previa el nivel micro que se acaba de evaluar en el apartado anterior. Es decir, cuando el agente *Service Supervisor* una vez que detecta una pérdida de rendimiento ejecuta directamente el proceso de adaptación de infraestructura a nivel macro.

Este tipo de adaptación se produce cuando la demanda en el servicio es mucho mayor, por lo que en este tipo de expermientos, el incremento en la carga computacional del servicio no se realiza tan progresivamente como en el experimento de adaptación anterior, sino que se realiza un incremento más agresivo de la carga. Así, en este caso se lanzan 10 hilos cada tres segundos, hasta un máximo de 40 hilos. Estos hilos al igual que en el caso anterior realizan consultas concurrentes al servicio de almacenamiento de ficheros. Concretamente los hilos consultan continuamente a los dos métodos ya mencionados: *GetSize* (método complejo) y *GetFolderContent* (simple).

La Figura 35 y la Figura 36 presentan el resultado de la ejecución del experimento cuando no se ha producido ningún tipo de adaptación. En ambas gráficas, en primer lugar, se puede observar un mayor número de peticiones, ya que al existir un mayor número de hilos en ejecución, el número de peticiones es mucho mayor. Como también se puede apreciar, en el caso de que no se produzca adaptación, el tiempo de respuesta se incrementa en gran medida, por lo que se reduce proporcionalmente el nivel de QoS aceptable. En este experimento, el nivel de QoS aceptable para la función *GetSize* sigue siendo 2,5 segundos, mientras que el nivel umbral de QoS para *GetFolderContent* se ha establecido en 0,5 segundos.

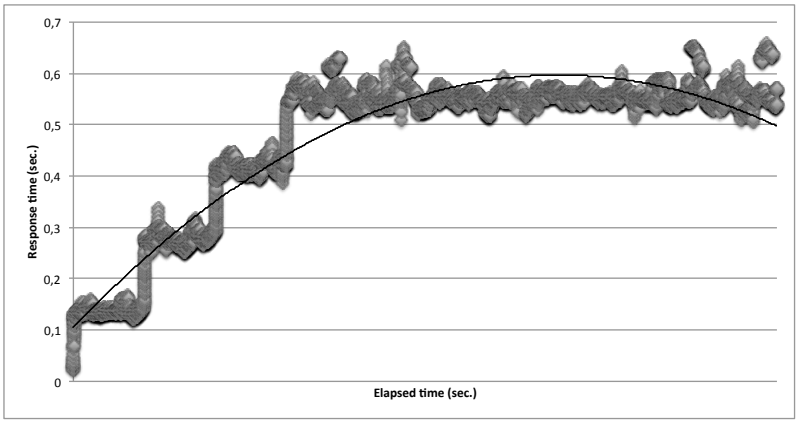

Figura 35.- Experimento 2: Reajuste de recursos de infraestructura a nivel macro, sin adaptación (Método: *GetFolderContent*).

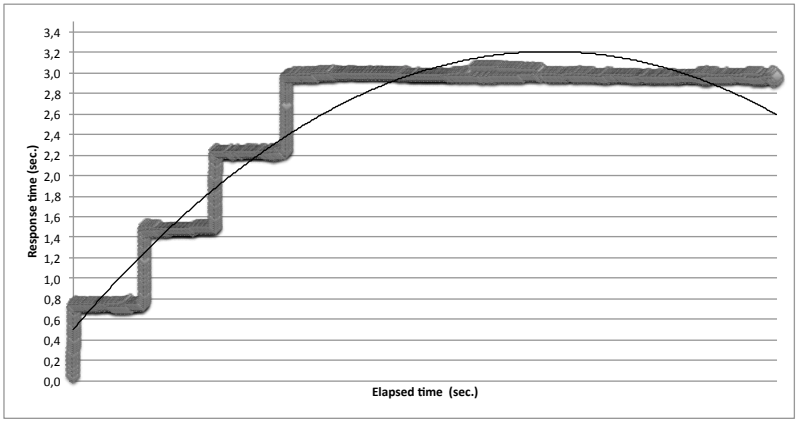

Figura 36.- Experimento 2: Reajuste de recursos de infraestructura a nivel macro, sin adaptación (Método: *GetSize*).

El proceso de interacción entre agentes para la adaptación de infraestructura a nivel macro se presenta en la Figura 37. Este proceso de adaptación se inicia de forma similar al proceso de distribución de recursos de infraestructura a nivel micro, pero en este caso, el agente especializado *Service Supervisor*, que inicia el servicio, envía la alerta (paso 1 , Figura 37) al agente *Global Manager* de cada una de las máquinas físicas que alojan los nodos del servicio. Se recuerda, que este agente *Global Manager* es un agente especializado que utiliza un proceso de razonamiento de tipo CBR-BDI [Corchado *et* Laza, 2003] [Corchado *et al.*, 2008] encargado de la distribución de recursos a nivel macro. Estos agentes, una vez que reciben la alerta inicial, reenvían este mensaje de alerta al resto de agentes *Global Manager* del sistema CC (paso 2, Figura 37).

El proceso que sigue se realiza de forma paralela en cada uno de los nodos físicos del sistema CC. Cada uno de los agentes *Global Manager*, que reciben el mensaje de alerta, solicita al agente *Local Monitor*, de la máquina en la que residen, la instantánea de estado del sistema (I_{FR}^{t}). A partir de esta información, determinan si existen recursos libres para instanciar un nuevo nodo asociado al servicio que demanda recursos. Si la máquina a partir de esta instancia detecta que el servidor físico no tiene recursos libres, entonces no realiza ninguna acción. Pero, en el caso de que sí tuviera recursos libres inicia el proceso de razonamiento que se detalló en el apartado 4.3.2.3 del Capítulo 4. Este proceso es realizado de forma paralela en cada una de las máquinas que dispone de recursos libres.

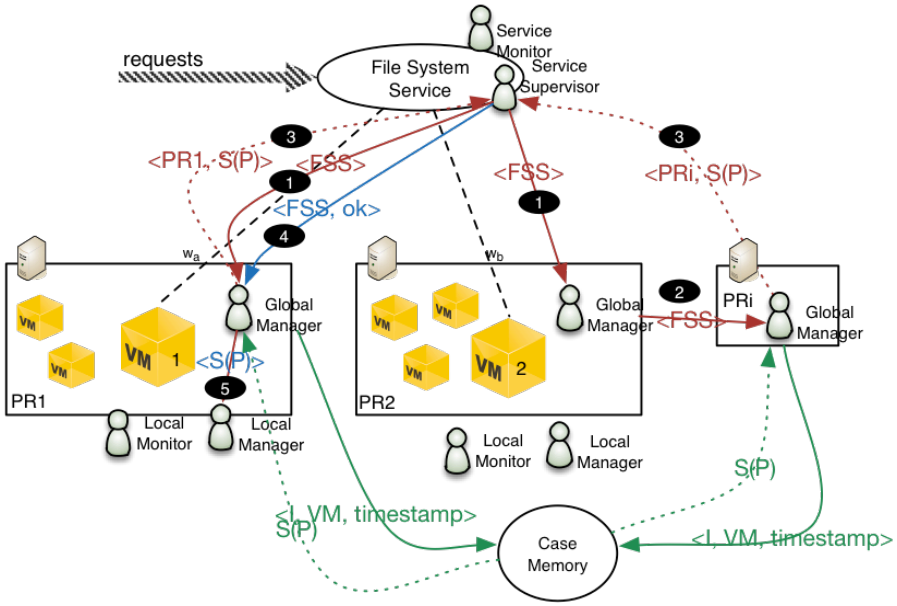

Figura 37.- Experimento 2: Intercambio de mensajes en la distribución de infraestructura a nivel macro

El proceso de razonamiento se inicia definiendo el problema, P, en cada máquina física. La caracterización del problema se construye a partir de la instantánea de uso que proporciona el agente *Local Monitor*, los recursos mínimos del nuevo nodo que se desea instanciar y la marca de tiempo actual. A partir de esta descripción del problema se recuperan los casos similares de la memoria de casos. Cada agente *Global Manager* de forma independiente determina los recursos que puede conceder al nuevo nodo que se acaba de instanciar. Esta nueva provisión de recursos es, en sí misma, la solución del problema para el caso actual, $S(P)$.

El conjunto de soluciones que proponen cada uno de los agentes CBR-BDI de cada servidor físico con recursos libres se envía al agente *Service Supervisor* que ha iniciado el proceso debido a sus problemas con el rendimiento (paso 3, Figura 37). Este agente *Service Supervisor* encargado de controlar el servicio, una vez que haya recibido el conjunto de soluciones propuestas por los diferentes agentes CBR-BDI de la organización, envía un mensaje de aceptación a aquel agente *Global Manager* que haya ofertado un mayor número de recursos para el nuevo nodo que se tiene que instanciar (paso 4, Figura 37). Finalmente, el agente *Global Manager* que recibe la solicitud, solicita al agente *Local Manager* (paso 5, Figura 37) de su misma máquina, que instancie un nuevo nodo virtual a partir de la plantilla de la máquina virtual asociada al servicio según la solución del problema que se ha propuesto.

Posteriormente, una vez que se lanza el nuevo nodo de ejecución, resulta necesario evaluar la solución propuesta, esta evaluación se realiza por el agente *Local Manager*, en función del grado de recursos infrautilizados del nodo que se acaba de instanciar. En el caso de que sea necesario un nuevo proceso de distribución de recursos a nivel macro, será el agente *Service Supervisor* el que complete la evaluación realizada por el agente *Local Manager* para así penalizar dicha solución. En ambos casos, se evalúa la eficiencia de la solución propuesta en función de la cantidad de recursos infrautilizados del nuevo nodo instanciado.

El caso de estudio se ha repetido en numerosas ocasiones lo que ha permitido almacenar en la memoria de casos un buen número de experiencias pasadas. A partir de este proceso se detecta que el proceso de razonamiento está altamente influenciado por el contenido de la memoria de casos, así en función de la cantidad y calidad de los casos existentes, correspondientes a experiencias pasadas, se obtienen mejores resultados en cuanto a la adaptación de infraestructura a nivel macro. Por ejemplo, en la Figura 38, y posteriormente, en la Figura 39 se presenta una distribución de recursos de infraestructura a nivel macro donde la memoria de casos estaba vacía y, por lo tanto, se instanciaron máquinas con una cantidad de recursos igual a la cantidad de recursos mínimos que puede tener el servicio, según su plantilla. Como se aprecia, en ambos casos, la adaptación de recursos soluciona el problema.

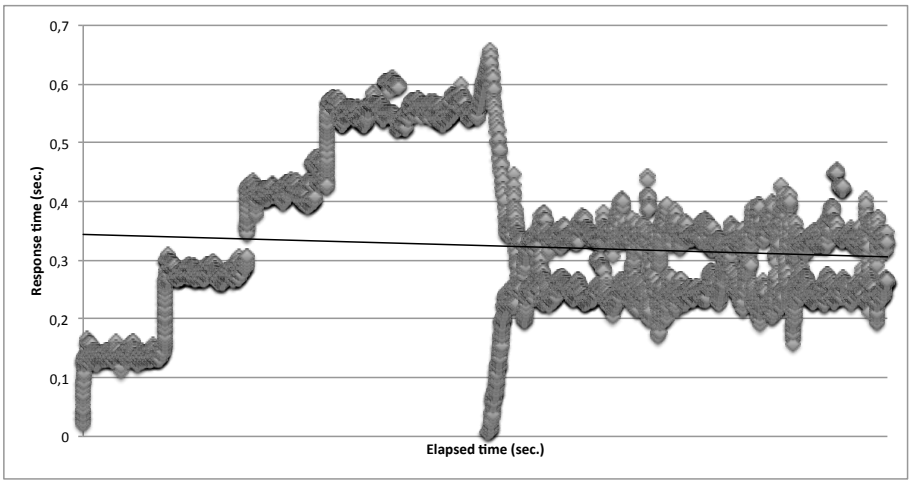

Figura 38.- Experimento 2: Reajuste de recursos de infraestructura a nivel macro, adaptación 1 (Método: *GetFolderContent*)

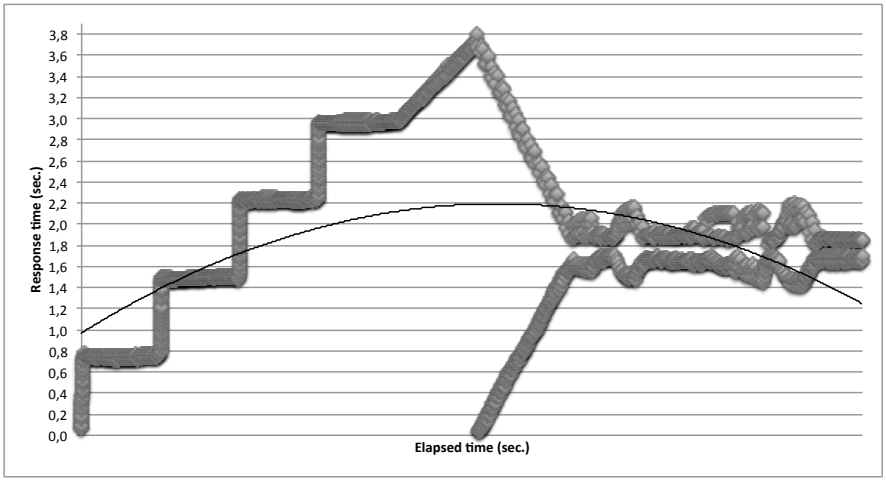

Figura 39.- Experimento 2: Reajuste de recursos de infraestructura a nivel macro, adaptación 1 (Método: *GetSize*).

Aunque en los dos casos anteriores el problema se soluciona, el nivel de QoS se aproxima al margen establecido como de baja calidad (0,5 en el caso de *GetFolderContent* y 2,5 en el caso de *GetSize*). En cambio, como se presentan en la Figura 40 y en la Figura 41, cuando se dispone de una gran cantidad de casos en la memoria, de forma que exista un buen número de experiencias pasadas similares a la actual, se observa que los resultados de adaptación son mejores debido a que el nivel de QoS es inferior. Sin embargo, en este caso, también se observa como desventaja, que el tiempo de razonamiento es mayor, debido a que existe un mayor coste computacional asociado a la recuperación de los casos y la búsqueda de la mejor solución a partir de estos.

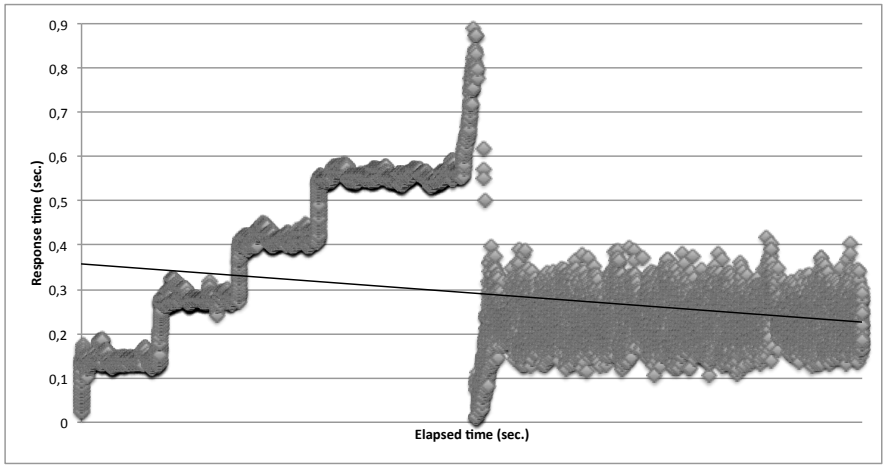

Figura 40.- Experimento 2: Reajuste de recursos de infraestructura a nivel macro, adaptación 2 (Método: *GetFolderContent*).

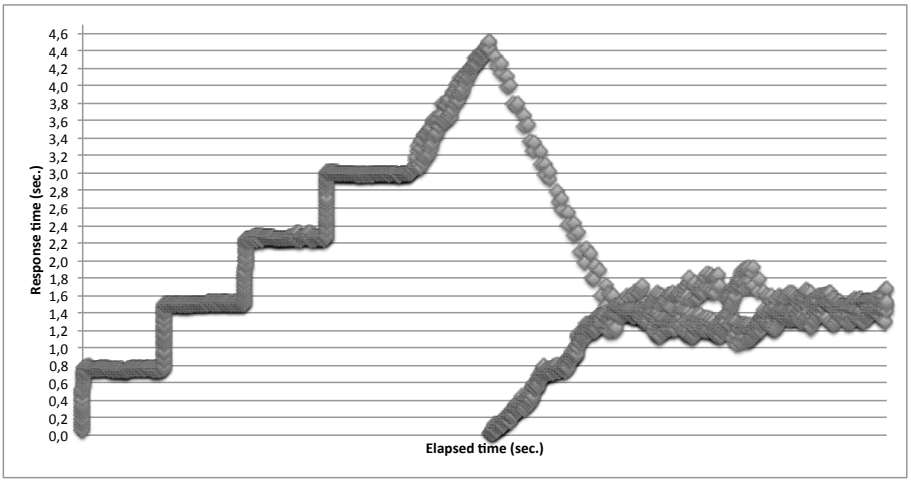

Figura 41.- Experimento 2: Reajuste de recursos de infraestructura a nivel macro, adaptación informada (Método: *GetSize*).

Posteriormente a estos experimentos, también se realizaron otros conjuntos de experimentos de distribución de recursos de infraestructura a nivel macro en los que se buscaban varias ejecuciones consecutivas del proceso de adaptación, de forma que, con una única ejecución del algoritmo de adaptación no fuera posible satisfacer la demanda en los servicios. Este tipo de pruebas, desde el punto de vista del modelo de adaptación fue positiva, ya que éste funcionó perfectamente dentro de los límites del caso de estudio. Sin embargo, tal y como se presenta en la gráfica de la Figura 42 el tiempo de respuesta después de 2 ó 3 procesos de readaptación, dependiendo del método que se esté evaluando, la QoS no se reduce como en las primeras

readaptaciónes, y el tiempo de respuesta sufre una gran dispersión. En este sentido, se determina que esta dispersión no está derivada de los algoritmos de distribución de recursos propuestos, sino de la saturación debido a la demanda de recursos en las capas inferiores que contienen la persistencia de los datos (bases de datos y sistema de almacenamiento distribuido).

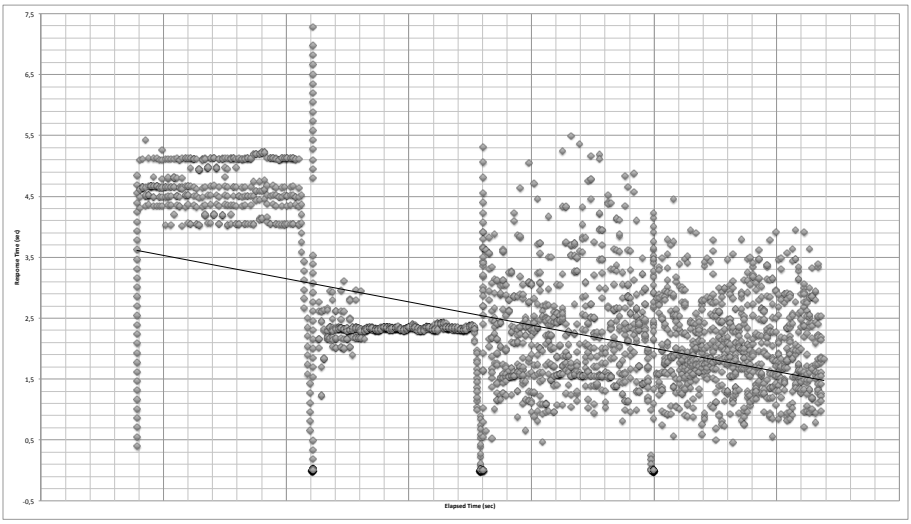

Figura 42.- Experimento 3: Reajuste de recursos de infraestructura a nivel macro, adaptaciones consecutivas (Método: *GetSize*).

5.2.4 CONCLUSIÓN DEL CASO DE ESTUDIO

El modelo de adaptación dinámico propuesto se enmarca dentro de las propuestas que buscan incrementar la eficiencia energética, según la clasificación descrita en el apartado 4.3.1 del Capítulo 4. Este tipo de modelos hoy en día están todavía en un estado de desarrollo e inmadurez, debido a que las grandes plataformas utilizan la otra gran aproximación para la distribución de recursos computacionales, la cuál es predominante y está guiada por los intereses económicos [Buyya *et al.*, 2008] [You *et al.*, 2009], en detrimento de los intereses energéticos y de eficiencia. Sin embargo, en el marco de la investigación, los trabajos analizados señalan que los esfuerzos se dirijen a la búsqueda de soluciones a la distribución de recursos computacionales en entornos CC que sean eficientes, es decir, en línea con el modelo que se ha propuesto en este trabajo de investigación. De estas forma es posible asegurar los niveles de QoS acordados mediante acuerdas SLA con los usuarios, mientras que al mismo tiempo sean capaces de ser eficaces energéticamente asegurando una consumo de recursos energéticos adecuado a la demanda de los recursos [Buyya *et al.*, 2010b].

La comparación empírica del modelo propuesto con respecto a otras aproximaciones existentes en el estado del arte no es posible, ya que resulta difícil recrear los entornos de computación o/y simulación en el que han sido evaluados. Sin embargo si que se puede realizar una comparación teórica de la aproximación propuesta con respecto a otros trabajos existentes en el estado del arte. En primer lugar, se observa que el modelo propuesto sigue una aproximación distribuida para resolver el problema lo que lo diferencia totalmente de los trabajos existentes en el estado del arte [Raghavendra *et al.*, 2008] [Beloglazov *et al.*, 2012]. Esta aproximación, la cuál se ha demostrado válida para la distribución de recursos computacionales en este tipo de entornos CC, tiene ventajas con respecto a su disponibilidad, ya que no existe un único componente encargado de la distribución de recursos, sino que es el propio sistema (sociedad) el que se reorganiza en su conjunto mediante la adaptación individual de sus componentes (agentes).

En las aproximaciones existentes en el estado del arte, tal y como se detalla especialmente Goudarzi *et* Pedram [Goudarzi *et* Pedram, 2011] la ejecución de los algoritmos de asignación es una tarea compleja que require una gran cantidad de potencia y tiempo computacional. Sin embargo, en el modelo propuesto se simplifica la búsqueda de una solución adecuada al problema, ya que (i) se distribuyen las necesidades computacionales entre diferentes nodos; (ii) el espacio de valores a considerar es menor, ya que cada nodo sólo debe considerar los datos sobre sus recursos propios y, además, no se necesita un conocimiento global de la plataforma; y, finalmente, (iii) cada nodo puede aplicar de forma autónoma una solución parcial al problema, eliminando las necesidades de coordinación a nivel global de la plataforma. Por otro lado, en cuanto a los algoritmos concretos de adaptación, en la propuesta se utilizan técnicas de optimización, las cuáles han sido utilizadas previamente [Kusic *et al.*, 2009], pero nunca siguiendo una aproximación distribuida.

La otra gran diferencia de este trabajo con respecto a otras aproximaciones existentes en el estado del arte es la unidad minima de distribución. Mientras que habitualmente en el estado del arte esta unidad mínima es la máquina virtual [Buyya *et al.*, 2010b] [Beloglazov *et al.*, 2012], en este trabajo se considera como unidades mínimas el número de *vcpus* y la memoria virtual asignada a cada máquina virtual de la infraestructura. Gracias a esta aproximación es posible distinguir entre nivel micro y macro en la distribución de recursos de infraestructura lo que permite solucionar problemas de demanda sin la necesidad de instanciar máquinas virtuales, lo que constituye en sí mismo una solución energéticamente eficaz y maximiza el uso de los recursos computacionales.

Finalmente, la clara ventaja que se observa del modelo propuesto con respecto a otros trabajos, es su capacidad para el aprendizaje. En el modelo de

distribución de infraestructura a nivel macro se ha utilizado como algoritmo de adaptación basado en un sistema de razonamiento basado en casos que se integra en un agente especializado de la organización +Cloud. Gracias a este enfoque, tal y como se ha demostrado, es posible que el sistema aprenda a partir de las experiencias pasadas, consiguiendo una mayor eficiencia en las adaptaciones a medida que aprende, memorizando tanto las experiencias positivas, como negativas. En el estado del arte no existe ninguna aproximación similar en la que la distribución de recursos computacionales se realice en base a los resultados obtenidos en procesos de adaptación pasados.

5.3 DISCUSIÓN

En el apartado anterior, se ha evaluado de forma empírica a través de un caso de estudio la arquitectura multiagente y los modelos de adaptación propuestos en el marco de esta investigación. Gracias al conjunto de experimentos que se han realizado se ha podido validar varios aspectos tal y como se detallará a continuación.

En primer lugar, después de evaluar el modelo propuesto se puede afirmar que una arquitectura multiagente basada en OV resulta adecuada al problema a resolver (monitorización y control de un entorno CC) dado que puede ser incorporada a en un entorno masivamente distribuido y que, además, permite la integración de diferentes modelos de razonamiento avanzado que permiten satisfacer las necesidades de distribución dinámica de recursos computacionales entre los diferentes recursos ofertados.

Por otro lado, el diseño basado en OV permite realizar un modelado de alto nivel, de forma que es posible abstraer completamente el diseño de la implementación posterior. Por tanto, esta aproximación permite asegurar la independiencia entre los estratos software donde se realiza la toma de decisiones y aquellos en los que se realiza la ejecución de acciones en función de las decisiones tomadas. En un entorno CC, esta separación de responsabilidades es particularmente importante ya que, como se ha determinado durante las primeras fases del trabajo de investigación, en las plataformas existentes hoy en día se observa una alta dependencia del entorno tecnológico (herramientas de virtualización, balanceadores de carga, sistemas de archivos distribuidos, etc.). Esta dependencia, constituye una gran limitación y dificulta su evolución, ya que un cambio en alguno de los componentes *hardware* o *software* obliga también a alterar los algoritmos y técnicas que hacen posible la elasticidad del sistema.

En el caso de la arquitectura propuesta +Cloud, dado que utiliza puertos para la comunicación con el entorno esta dependencia se limita a la implementación del propio puerto, es decir, a la interfaz de comunicaciones con el entorno. No cabe duda que un cambio en las capacidades que ofrece la tecnología subyacente también obligaría a modificar los modelos de razonamiento propuestos, como en una aproximación de diseño tradicional. Sin embargo, para estos casos, los modelos organizativos también ofrecen una respuesta adecuada a esta dificultad. En un sistema como el que se propone, los agentes que forman parte de la sociedad, juegan uno o varios roles concretos. Cada rol define de forma abstracta los objetivos, responsabilidades y privilegios del individuo que lo adquiere. A partir de esta definición de alto nivel, en el caso de que la tecnología proponga nuevas capacidades, el trabajo de adaptación consistiría en modificar al individuo o individuos que realizan las

tareas concretas, los cuáles juegan roles en la organización. Por lo tanto, dado que no sería necesario alterar la sociedad en su conjunto y sólo las entidades individuales; la arquitectura propuesta también está por encima de otras plataformas existentes en el mercado.

En segundo lugar, se discuten las capacidades de +Cloud como sistema abierto. Gracias a este modelo de construcción de sistemas software es posible permitir que agentes externos a la organización accedan a la misma para proporcionar cierta funcionalidad, a cambio de que asuman y cumplan un conjunto de normas que garanticen la estabilidad y supervivencia de la sociedad. En el marco del trabajo propuesto se ha diseñado el sistema para que los agentes sean o puedan ser externos al sistema inicialmente diseñado. Dentro de este trabajo, por supuesto, los agentes humanos que participan en la organización (*Cloud User*, *End user* y *Cloud Admin*) son externos, pero también el agente es el *SLA broker* se ha diseñado de este modo. Debido a que el contexto de comercialización de un sistema CC pueda cambiar en el futuro. El sistema +Cloud de este modo está adaptado a estos nuevos modelos desde su concepción, ya que el diseño de este rol como externo permite que agentes accedan a la organización para proporcionar la funcionalidad de negociación y contratación de servicios, con independiencia del modelo de comercialización en el que se basen. Gracias a este modelo de diseño se contribuye a la interoperabilidad entre plataformas, ya que se construye el sistema facilitando un modelo abierto de conexión entre plataformas (Intercloud [Sim, 2013]). Por otro lado, los SMA que siguen un proceso de diseño basado en modelos organizaciones y sistemas organizativos, como es el caso del modelo arquitectónico propuesto tiene claras ventajas en el dominio de aplicación de los sistemas CC, ya que permiten la modificación en el diseño mediante la introducción de nuevos roles dentro de la sociedad, para lo cuál tan sólo es necesario definir sus responsabilidades, objetivos y normas.

Finalmente, en el apartado 3.2 del Capítulo 3 se presentó un estudio detallado acerca de las sinergias entre el paradigma CC y los SMA, determinado que en el estado del arte existe un incipiente número de trabajos en los que los SMA aportaban ventajas a los sistemas CC, dentro del grupo denominado por Talia como *Cloud using agents* [Talia, 2011] que posteriormente ha dado lugar a un concepto más amplio: *Agent-based Cloud Computing* [Venticinque *et al.*, 2010] [Talia, 2012] [Sim, 2012]. Este conjunto de trabajos analizados se determinó que podían organizarse según los roles de uso de los sistemas CC en la arquitectura de referencia del NIST [Liu *et al.*, 2011]. Pero además de esta estructuración inicial, también se identificó un conjunto de ámbitos de aplicación de los SMA en el marco de los sistemas CC (seguridad, busqueda de proveedores, negociación, monitorización, etc.).

En este trabajo, el modelo propuesto es la primera aproximación conocida de los SMA basados en organizaciones virtuales dentro del paradigma CC. Este enfoque, junto con el diseño de la organización como un sistema abierto permite considerar el entorno computacional CC, el cuál es complejo, tanto desde una perspectiva interna, como externa. Es decir, teniendo en cuenta tanto las capacidades y funcionalidades que se le ofrecen a los usuarios (negociación, acuerdos SLA, capacidades, etc.); como las propias características internas del entorno de computación (monitorización, gestión de infraestructura, monitorización, etc.). En este sentido, en la Tabla 9 se presentan los ámbitos de aplicación principales del los SMA en el marco del rol *Cloud Broker* del paradigm CC. Como se puede apreciar, siguiendo un modelo de diseño como el del sistema +Cloud, en el paradigma CC no sólo se obtienen ventajas de la aplicación de los SMA desde una perspectiva externa, sino que también se obtienen ventajas debido al control de las características internas de plataforma.

Usos principales	**Nivel interno**				**Nivel externo**			
	Moni.	**Cont.**	**Seg. Priv.**	**Gest. Infr.**	**Neg. SLA**	**Ofer. serv.**	**Com.**	**Eval. Serv.**
Búsqueda de proveedores								
Buscador de servicios	X	X	X			X	X	
Interoperabilidad					X	X		
Modelo de confianza en SLA	X	X			X		X	
Contratación de servicios		X	X		X		X	
Negociación de acuerdos SLA								
Negociación de condiciones		X				X		
Monitorización de acuerdos	X	X	X		X		X	
Composición de servicios								
Búsqueda de servicios compatibles						X	X	
Interoperabilidad	X		X			X		
Composición Horizontal y Vertical	X	X	X			X		

Tabla 9.-Tabla resumen SMA y CC, rol *Cloud Broker* en el entorno +Cloud

Destacar, finalmente, que tras revisar el estado del arte existente en el momento de la realización de este trabajo de tesis doctoral también se observó un limitado número de trabajos en los que los SMA jugaban el rol *Cloud Provider* del paradigma CC. El trabajo constituye una de las primeras aproximaciones como aplicación de los SMA dentro este rol. Finalmente, del mismo modo, en la Tabla 10 se presentan las sinergias entre el nivel externo e interno del uso de SMA organizativos para el rol *Cloud Provider* del paradigma CC. Este rol en el caso de seguir una aproximación tradicional tendría tan sólo ventajas a nivel interno.

Uso principales	**Nivel interno**				**Nivel externo**			
	Monit.	**Contr.**	**Seg. Priv.**	**Gest. Infr.**	**Neg. SLA**	**Ofer. serv.**	**Com.**	**Eval. Serv.**
Seguridad y privacidad								
Monitorización de infraestructura	X				X			X
Modelos de autenticación			X				X	
Privalegios de acceso			X				X	
Seguridad de la información			X			X		X
Almacenamiento seguro			X			X		
Oferta de servicios								
Servicios sensibles a la calidad	X	X			X			X
Oferta de servicios						X		
Gestión de infrastructura								
Gestión de recursos	X				X			X
Optimización de costes		X		X	X			X
Gestión de flujo de trabajo				X		X		

Tabla 10.- Tabla resumen SMA y CC, rol *Cloud Provider* en el entorno +Cloud

CAPÍTULO 6 CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

A lo largo de este trabajo de tesis doctoral se ha presentado un innovador enfoque basado en OV de agentes inteligentes para el control y monitorización de plataformas CC. Gracias a este modelo arquitectónico, implementado en el SMA +Cloud, ha sido posible modelar la elasticidad de los servicios que ofrece una plataforma CC siguiendo un enfoque distribuido y haciendo uso de técnicas derivadas de la IA. Gracias a lo cuál se ha podido introducir en los sistemas CC características como la *autonomía*, *habilidades sociales*, *reactividad* y *pro-actividad*; que tradicionalmente estaban asociados a los SMA. Posteriormente, esta arquitectura propuesta (+Cloud) se ha sometido a un intenso proceso de evaluación a través de un conjunto de experimentos realizados en el marco de un caso de estudio diseñado específicamente, lo que ha permitido validar tanto el modelo, como las diferentes técnicas y algoritmos propuestos, en el marco del proceso de investigación, para proporcionar elasticidad al sistema CC.

Una vez que el proceso de investigación ha terminado, el presente capítulo tiene como objetivo enunciar las conclusiones alcanzadas, así como la contribución de esta investigación al estado del arte (apartado 6.1). Posteriormente, a partir de este hito en el proceso de investigación, en el apartado 6.2 se enunciarán las líneas de trabajo futuro que guiarán el proceso de investigación a medio y largo plazo.

6.1 CONCLUSIONES

Este trabajo se planteaba, en sus inicios, como una de las primeras aproximaciones de los SMA, y más concretamente aquellos que siguen una orientación basada en OV, en el marco de los sistemas de control y monitorización de un entorno CC. Como se ha analizado ampliamente, tanto en el Capítulo 2, como posteriormente en el Anexo C de esta memoria; hoy en día el desarrollo de este tipo de plataformas CC está fundamentalmente dirigido por los intereses empresariales de las grandes compañías tecnológicas. Sin embargo, el trabajo realizado viene a ratificar que en este ámbito de investigación no tiene por qué estar únicamente ligado a la innovación dirigida a la creación de productos comercionales, sino que también hay cabida para la investigación fundamental, a través de la búsqueda de nuevos modelos y algoritmos que permitan hacer más eficiente el ecosistema de los sistemas CC y mejorar la interacción con los usuarios finales.

En este marco de investigación, se ha propuesto un nuevo modelo arquitectónico, basado en OV de SMA, que tiene un carácter claramente integrador. Dentro de esta arquitectura se ha desarrollado, evaluado y validado un conjunto de algoritmos para la distribución de recursos computacionales en entornos CC. Su principal novedad se basa en la capacidad de auto-adaptación dinámica del sistema propuesto en función de la demanda. Así, no sólo se realiza una distribución de recursos en función de la demanda de los servicios, sino que es el propio sistema el que se adapta dinámicamente en función de los cambios que se produzcan en el entorno. Así mismo, este esquema de adaptación dinámica se basa en modelos distribuidos lo que reduce la cantidad de información en la toma de decisiones, permitiendo el reparto de responsabilidades y subobjetivos entre los diferentes miembros del sistema.

En conclusión, la plataforma CC que integra la arquitectura propuesta, denominada +Cloud, a tenor de los resultados (empíricos y teóricos) presentados a lo largo de este capítulo permiten validar que la hipótesis de partida de este trabajo de investigación: "*es posible modelar el sistema de control y monitorización de una plataforma CC mediante el uso de OV de agentes inteligentes que se auto-adapten y reorganicen en función de las necesidades del contexto. Así pues, los agentes individuales de forma autónoma gestionarán los recursos computacionales y los servicios disponibles, coordinándose con el resto de agentes para proporcionar un modelo de adaptación dinámico y distribuido en función de los intereses del usuario final*".

Por lo tanto, una vez que se ha validado la hípotesis de partida, se concluye que también se ha satisfecho el objetivo principal, y los subobjetivos

asociados. Así, en el siguiente subapartado (6.1.1) se presentarán las principales contribuciones de este trabajo de investigación. Posteriormente, en el subapartado 6.1.2 se presentarán las contribuciones de este trabajo a los proyectos de investigación en el que se enmarca, así como las tareas de difusión realizadas.

6.1.1 CONTRIBUCIONES A LA INVESTIGACIÓN

El trabajo de investigación que se ha detallado a lo largo de la presente memoria, ha permitido contribuir al estado del arte en los ámbitos de la teoría de agentes, las organizaciones virtuales y los sistemas distribuidos inteligentes. A continuación se describen las principales contribuciones de esta investigación:

- **Se ha estudiado las necesidades de un entorno CC.** A partir del análisis del estado del arte, incluyendo tanto estudios y trabajos previos, como las plataformas existentes; se ha elaborado un marco de aplicación del trabajo de investigación propuesto, lo que ha permitido detectar las debilidades de la tecnología y la identificación de las áreas susceptibles de mejoras.
- **Se ha analizado la relación entre la teoría de agentes y el paradigma CC.** Durante el trabajo de investigación que se ha llevado a cabo se ha analizado exhaustivamente el estado del arte con el objetivo de descubrir sinergias entre ambos sistemas distribuidos (SMA y CC). De este modo, se ha podido demostrar que en el estado del arte existe suficiente trabajo previo como para sostener la hipotésis inicial de este trabajo de investigación.
- **Se han analizado los modelos organizativos de agentes inteligentes para el diseño de sistemas *software* complejos.** Durante el análisis del estado del arte se ha determinado que el modelo más adecuado para diseñar una arquitectura orientada a la gestión, control y monitorización de un sistema CC es mediante el uso de modelos organizativos de agentes inteligentes.

  Partiendo de esta premisa se ha procedido a analizar los diferentes modelos para el diseño de sistemas abiertos y complejos, como es el caso de un sistema CC. Así pues, como resultado de este análisis, se ha determinado que la metodología GORMAS [Argente, 2008] [Argente *et al.*, 2011] es la que más se ajusta a los intereses perseguidos, debido a que su amplio meta-modelo permite caracterizar correctamente un sistema *software* complejo. Además, esta metodología está basada en un proceso de diseño iterativo, alineado con SPEM [OMG, 2008].
- **Se ha realizado un amplio estudio de las tecnologías complementarias**. Este trabajo hace uso de un buen número de

tecnológicas que completan el trabajo de investigación realizado. En las fases iniciales, se ha realizado un estudio de todas ellas. Por un lado, se ha hecho un *análisis del contexto tecnológico en el entorno CC*, lo que ha permitido seleccionar el conjunto de tecnológias que han formado parte del entorno CC desarrollado (balanceo, virtualización, persistencia, etc.). Así mismo, también se ha realizado un *estudio de técnicas, algoritmos, métodos y modelos de IA* que han constituido la base de los modelos de razonamiento avanzado y aprendizaje desarrollados.

- **Se ha caracterizado el entorno de computación.** A partir del análisis del estado del arte de los diferentes ámbitos que abarca este trabajo de investigación, se ha procedido a formalizar el entorno de computación distribuido sobre el que posteriormente se ha aplicado el modelo de adaptación propuesto. Esta caracterización del entorno no sólo ha constituido el punto de partida para la formalización de la arquitectura que se ha propuesto, sino que también forma parte de los resultados del trabajo realizado, sirviendo como base para diferentes líneas de investigación que se realicen en el futuro.
- **Se ha diseñado una arquitectura multiagente auto-organizativa adecuada al problema a resolver.** Se ha diseñado un modelo arquitectónico específico para el control y monitorización de un sistema CC a través de un modelo organizativo de agentes inteligentes. Este modelo se ha evaluado y validado como apropiado para este contexto. Esta arquitectura integradora permite incluir diferentes agentes especializados con capacidades de razonamiento avanzadas para la distribución de recursos computacionales en un entorno distribuido.
  Para el diseño de esta arquitectura se ha seguido una aproximación basada en sistemas abiertos, lo que hace posible que en el futuro se puedan incluir nuevos modelos, técnicas y algoritmos como parte de la organización diseñada. Así, este trabajo constituya un resultado extrapolable a futuras investigaciones.
- **Se ha diseñado un modelo de adaptación dinámica mediante un sistema de razonamiento basado en casos y algoritmos de optimización.** La distribución de recursos en entornos computacionales es, básicamente, un problema de optimización. Este punto de partida inicial, derivado del estudio del estado del arte, se ha complementado mediante la inclusión de un motor de razonamiento basado en casos [Corchado *et al.*, 2008] que permite que los algoritmos propuestos aprendan de las experiencias basadas, obteniendo mejores resultados con cada nuevo proceso de adaptación.
- **Validación del modelo arquitectónico propuesto en un entorno real de aplicación.** En el contexto de los proyectos de investigación asociados a este trabajo de tesis doctoral se ha desarrollado una plataforma CC, la cuál ha integrado el modelo arquitectónico propuesto

basado en OV (+Cloud). Gracias a lo cuál, se ha podido evaluar el modelo propuesto en un entorno de aplicación real y un contexto de explotación. Gracias a esta posibilidad se ha podido evaluar extensivamente el modelo propuesto.

6.1.2 CONTRIBUCIÓN A PROYECTOS Y DIFUSIÓN DE RESULTADOS

Además de las contribuciones al estado del arte que se acaban de presentar en el apartado anterior, este trabajo de tesis doctoral también ha permitido sentar las bases del desarrollo de un importante proyecto de investigación fundamental no orientada, a nivel nacional:

- **iHAS: Intelligent Social Computing for Human-Agent Societies** (TIN2012-36586-C03-0), otorgado por el Ministerio de Ciencia e Innovación y los fondos FEDER. http://ihas.usal.es

Por otro lado, este trabajo de tesis doctoral, también ha sido fundamental para el desarrollo de otros proyectos de investigación y desarrollo, que se han llevado a cabo en colaboración con diferentes empresas a nivel nacional y regional:

- **Cloud-IO: Plataforma Cloud Computing para la Integración y Despliegue Rápido de Servicios sobre Redes Inalámbricas de Sensores** (IDI-20111471), otorgado por el Centro para el Desarrollo Tecnológico e Industrial (CDTI) del Ministerio de Economía y Competitivida y los fondos FEDER.
  http://cloud-io-project.com/
- **SmartMedical: Plataforma Cloud Computing para gestión de historiales clínicos en aseguradoras privadas** (IDI-20120794), otorgado por el Centro para el Desarrollo Tecnológico e Industrial (CDTI) del Ministerio de Economía y Competitivida y los fondos FEDER.
  http://smartmedical.es/
- **DoyFE.ES: Verificación y prevención de fraude en contenidos digitales** (IDI-20120798) otorgado por el Centro para el Desarrollo Tecnológico e Industrial (CDTI) del Ministerio de Economía y Competitivida y los fondos FEDER.
  http://www.doyfe.es/

Algunos de estos proyectos, que todavía no han finalizado (iHAS y DoyFe), permitirán continuar el desarrollo de esta línea de investigación, a través del desarrollo tanto de la plataforma CC, como del SMA +Cloud. Gracias a lo cuál se permitirá extender el rango de escenarios de aplicación del modelo construido.

Finalmente, resaltar que también se ha realizado un gran esfuerzo para intercambiar conocimiento con distintos investigadores y expertos en los ámbitos de investigación de este trabajo de tesis doctoral. De esta forma se ha podido obtener una opinión crítica y constructiva, que ha constituido una pieza clave, en términos de realimentación en las diferentes etapas iterativas del desarrollo del modelo propuesto.

Así mismo, también se ha realizado un intenso esfuerzo en la difusión y diseminación del trabajo mediante la publicación de los resultados en revistas internacionales y la asistencia a conferencias, congresos y foros especializados. Parte de estas contribuciones y actividades de difusión se han realizado en colaboración con grupos interesados en el trabajo de investigación realizado, como es el caso del GTI-IA[68] - *Grupo de Tecnología Informática - Inteligencia Artificial* (UPV, Universidad Politécnica de Valencia) y el *Laboratoire d'Informatique de Grenoble* (CNRS, *Centre National de la Recherche Scientifique*). Finalmente, en cuanto al intercambio de conocimiento y difusión con expertos en el ámbito de la investigación, también se ha buscado la colaboración con el tejido empresarial en el marco de la TIC, prueba de ello son las publicaciones en congresos internacionales realizadas conjuntamente con expertos de empresas como INSA – Ingeniería del Software Avanzado S.A.[69], Flag Solutions S.L.[70] y Nebusens S.L.[71]

[68] http://www.gti-ia.upv.es/
[69] http://www.insags.com/
[70] http://www.flagsolutions.net/
[71] http://www.nebusens.com/

6.2 LÍNEAS DE TRABAJO FUTURAS

El trabajo de investigación que se ha detallado a lo largo de esta memoria constituye un hito plausible en un largo proceso de búsqueda e indagación de nuevas soluciones, técnicas, herramientas y modelos. Sin embargo, no cabe duda de que este hito constituye el punto de partida de una investigación con un recorrido en el tiempo mucho mayor, debido a que se centra en un ámbito que hoy en día está en clara expansión. Por ello, se han establecido un conjunto de líneas de trabajo a desarrollar que permiten continuar fortaleciendo la base que ya se ha alcanzado con el desarrollo de este trabajo de investigación. Gracias a esta planificación en la investigación futura, será posible inferir nuevo conocimiento orientado solventar problemas y corregir debilidades en el marco de este contexto tecnológico.

Este marco de búsqueda e indagación de nuevas soluciones y modelos, indudablemente también estará altamente influenciado por los objetivos e intereses de los proyectos que sirvan como base a este trabajo de investigación. Concretamente, este trabajo está estrechamente enlazado con el proyecto iHAS, el cuál centra sus intereses en la investigación en mecanismos, algoritmos, herramientas y modelos que permitan la creación de máquinas sociales en los que convivan e interactúen agentes virtuales y humanos de forma transparente en un entorno totalmente integrado. Denominamos este tipo de sistemas sociedades humano-agente (*human-agent societies* HAS).

Dentro este proyecto, el modelo arquitectónico propuesto en este trabajo de investigación está alineado con el primer demostrador del proyecto, que pretende desarrollar un modelo de computación social en el marco de los entornos CC. Por lo tanto, el trabajo desarrollado en el marco de esta tesis doctoral satisface los objetivos marcados para el desarrollo del citado demostrador, ya que se han (i) estudiado las necesidades de un entorno CC; se (ii) ha diseñado un sistema CC utilizando la base que proporcionan las OV de agentes inteligentes; a partir de este diseño, se (iii) ha implementado un prototipo funcional que, posteriormente, se (iv) ha evaluado pormenorizadamente en un entorno de explotación real.

Aunque, como se indicado, ya se han alcanzado los objetivos propuestos, no sólo en el marco de la tesis doctoral, sino también dentro del proyecto al que se asocia este trabajo de investigación; partiendo de esta base, es posible proponer las siguientes líneas de trabajo que se acomentarán a corto y medio plazo como complemento a los objetivos perseguidos inicialmente:

- Comparación del modelo propuesto con plataformas existentes, evaluando la posibilidad de integración del sistema +Cloud con otras

plataformas CC y evaluando el rendimiento de los modelos de monitorización, control y adaptación en diferentes sistemas CC libres.

- Definición y construcción de un modelo de negociación de acuerdos SLA, que integre diferentes modelos de razonamiento avanzados derivados de la IA para guiar el proceso de negociación, lo que permitirá integrar en la plataforma desarrollada, el servicio de contratación automatizada de los productos que se oferten.
- Definición de un modelo de interoperabilidad de servicios en función de su tipo y ámbito (*software*, plataforma e infraestructura) lo que permitirá definir un protocolo que facilite su composición con independencia del proveedor de los mismos.
- Definición de un modelo de coste que permita determinar en función de la demanda, el modelo de producción y el gasto energético, un modelo de coste que permita establecer precios variables en función del estado concreto del sistema CC.
- La combinación de las tres líneas de trabajo anteriores, permitirá dar soporte a la interoperabilidad entre plataformas a nivel horizontal y vertical de los productos ofertados en una plataforma CC. Para ello, se pretende contribuir al desarrollo del concepto denominado como Intercloud [Buyya *et al.*, 2010a] [Sim, 2013] mediante el desarrollo de nuevas técnicas, herramientas y modelos que hagan uso de las capacidades los SMA para participar en este modelo de interoperabilidad
- Extensión de los algoritmos de distribución de recursos computacionales con dos objetivos principales. Por un lado, se pretende adaptar el modelo de auto-adaptación dinámico propuesto para así poder incluir todas las capas *software* de un sistema CC, incluyendo el estrato de persistencia. Para ello, será necesario actualizar el modelo propuesto de forma que también integre las peculiaridades referentes a la elasticidad de las bases de datos y los sistemas de almacenamiento distribuido en red que se han analizado y caracterizado a lo largo del trabajo de investigación ya realizado. Por otro lado, también se pretende ampliar el modelo de adaptación propuesto para que pueda abarcar otros productos de infraestructura, especialmente aquellos que hagan posible la computación de altas prestaciones orientadas al análisis masivo de datos.
- Finalmente, como última línea de trabajo se pretende investigar en el desarrollo de nuevos modelos de interacción de los usuarios con el sistema CC, a través de los últimos dispositivos y modelos disponibles. De esta forma, es posible transformar las plataformas del paradigma CC en máquinas sociales que se adapten a las necesidades de los usuarios

no sólo en términos de rendimiento de los servicios, sino también en función de las interacciones con los usuarios.

CAPÍTULO 7 RESEARCH OVERVIEW

The technology industry is presently making great strides in the development of the paradidgm of *Cloud Computing* (CC). As a result, the number of both closed and open source platforms has been rapidly increasing. They all have a similar architecture. From an external point of view, the three most widely known services are Software, Platform and Infrastructure. From an internal point of view, the services generally offered are considered elastic services due to the high number of underlying technologies (virtualization, server farms, web services, web portals, etc.) which have reached their prime. Although at first glance this may appear to be simply technological paradigm, reality shows that its rapid progression is primarily motivated by economic interests that surround the purely computational or technological characteristics. The competitive advantage associated with knowing and dominating this technology is key to understanding the rapid increase of a technology that as little as five years ago did not even exist. Businesses that have a Cloud environment will have an advantage over others in their sector.

The main objective of this study is the development of a new generation CC systems that can provide any type of resources (service, platform and infrastructure), and whose core is composed of a multiagent architecture based on virtual organizations. This will allow the self-adaptation in order to manage the elasticity by using a knowledge model with elements of uncertainty, which will facilitate its implementation independent of the size of the data center. To this end, we will need to take into account not only the underlying infrastructure, but also the services that are offered to the end user. This model will make it possible to create a model of elasticity that allows autonomous, dynamic and automatic adaption, and learns from past experiences, with the aim of offering computational services of any type.

This chapter is a summary of the research presented in this thesis work. The chapter is structured as follow. Section 7.1 is a brief introduction of the study. Section 7.2 describes the primary characteristics of the agent organizations and the state of the art for Cloud Computing paradigm. Section 7.3 presents the structure of the proposed architecture and the algorithms for allocation resources over the servicies. Section 7.4 demonstrates how the model can be used in a specific case study; to this end, the context, the concepts and the model of the system will be explained. Finally, Sections 7.5 and 7.6 will emphasize the results and conclusions obtained.

7.1 INTRODUCTION

The technology industry and the scientific community have taken great strides in recent years toward implementing the technological paradigm of CC. This has resulted in the rapid growth of platforms, both private and public [Fisher *et al.*, 2010] [Lou *et al.*, 2011] [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012]. There is not little doubt that the general social acceptance of this paradigm [Leavitt, 2009] has in large part led to its development, due to the economic interests of large technology companies that underlie the purely technical aspects [Buyya *et al.*, 2009] [Armbrust *et al.*, 2010]. From an external point of view, the CC platforms offer three well-known types of services (software, platform and infrastructure) [Mell *et* Grance, 2011], where novelty, with respect to previous technologies, lies in the ability to offer any type of computing ability as a service through the Internet.

The marketing model used in the CC paradigm is also innovative, as it is based on a pay-as-you-go concept [Armbrust *et al.*, 2010], just as with any other traditional public service (light, water, gas, etc.). In this sense, CC follows in line with the model proposed by the abstract *Utility computing* paradigm [Ross *et* Westerman, 2004] in which users must negotiate and previously establish a Service Level Agreement (SLA) in order to access services [Alhamad *et al.*, 2010a]. Once this contract for computing goods has been established, both the users (through regular payments) and the CC system (by maintaining the service) are obligated to follow through with their agreement.

This novel marketing model contractually requires the CC platforms to maintain the quality of services as previously agreed upon, which forces the analysis of the internal architecture that makes the marketing model possible. In this regard, novelty is determined by the innovative spectrum of underlying technology (virtualization, service farms, web services, etc.), which have recently reached the point of allowing the services to be offered with the same level of quality, regardless of existing user demand [Liu *et al.*, 2011] [Wang *et al.*, 2008] [Zhang *et al.*, 2010].

New possibilities at a technological level lead to the birth of a new concept, elasticity [Chiu, 2010]. This concept is based on the *just-in-time* production method [Hutchins, 1999], which references the manner in which the services (computational) and the resources they require are produced. Thus, the services produced within the framework of CC only receive the amount of resources they need to maintain a uniform level of quality while immediately responding to demand [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012].

As a result, the main objective within this context is to maintain the level of quality of services (QoS) [Alhamad *et al.*, 2010a], regardless of demand, which implies a need to dynamically manage the computational resources assigned to each service offered to the user. In order to do so, these platforms predominantly use a management system (monitor and control) for the underlying computational infrastructure. These systems are often centralized and do not necessarily stress optimization of hardware resources, which consume vast quantities of energy and require high refrigeration [Song *et al.*, 2009].

Elastic algorithms should not only provide an elastic model (increasing and decreasing), but should do so following a model that can also reduce operational costs in the CC environment. However, existing research in the state of the art are based on methods that use centralized algorithms based on mathematical and heuristic models [You *et al.*, 2009] [Raghavendra *et al.*, 2008] [Kusic *et al.*, 2009], neither of which can ensure the efficiency of the system or, even less so, its availability in the event of a system failure. Therefore, this type of algorithms should evolve toward a model in which different agents are represented within an uncertain environment, which forces them to interact and share information with their equals. This would allow the algorithms managing the resources to be distributed throughout the system, which would therefore facilitate their implementation regardless of the size of the data center.

In this context, the theory of agents and multiagent systems (MAS) [Russel *et* Norving, 1995] [Wooldridge *et* Jennings, 1995] can provide a new model for managing CC systems based on the distribution of responsibilities, flexibility and autonomy. Managing the functions of the nucleus of a CC system through a agent-based model allows the resulting platforms to be much more efficient, scalable and adaptable than they currently are. However, joining both computational models (MAS and CC) is a great challenge, given the difference between the two models. However, since the CC system is considered an open system and the application of MAS in open systems is a recognized challenge in which there has already been a notable rate of success [Zambonelli *et al.*, 2004] [Reitbauer *et al.*, 2004] [Weyns *et al.*, 2004] [Capera *et al.*, 2003] [Razavi *et al.*, 2005] the union of the two models is an achievable challenge within the framework of this research study.

MAS that have been designed through organizational models are considered especially effective within the framework of this project. This is due to the fact that they can already provide advanced and innovative solutions [Dignum, 2004] that make it possible to exploit differentiating elements when it comes to providing flexibility, capability and response speed as part of a strategy defined by the provision of customer satisfaction [Schertler, 1998].

Finally, the use of MAS enables continuing research in techniques, tools and methodologies that allow incorporating intelligent characteristics, such as autonomy or experiential learning, on a CC platform.

Thus, the initial hypothesis is that *it is possible to model the control and monitoring system of a CC platform by using a VO of intelligent agents that self-adapt and reorganize according to the needs of the surrounding environment. Individual agent will autonomously manage the resources and services available and coordinate with the other agents to provide a dynamic and distributed adaptation model according to the needs of the end user.*

7.2 CLOUD COMPUTING AND MULTIGENT SYSTEMS

It is now possible to identify certain weaknesses in current CC platforms, weaknesses that appear when efficiently managing resources, responding autonomously during peak demand, automating the decision-making process, responding to system failures, etc. [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012]. In order to respond to these current problems, we can identify the MAS [Russel *et* Norving, 1995] [Wooldridge *et* Jennings, 1995] according to the hypothesis of this thesis as the key to providing a possible solution to the identified problems, thus allowing the evolution of the CC paradigm through the incorporation of new characteristics within the framework of distributed artificial intelligence [Talia, 2012] [Kang *et* Sim, 2011].

Due to the complexity of the CC systems, in which there is a broad range of technologies, components, systems, etc., this section will provide a detailed analysis of the characteristics of the CC, identifying strengths and weaknesses, as well as specific needs. The synergy between both distributed systems will also be analyzed. Finally, an appropriate design model will be defined based on the MAS organizational models.

7.2.1 CLOUD COMPUTING AND RELATED TECHNOLOGIES

Historically speaking, Cloud Computing was first introduced as a term by Professor R.K. Chellappa [Chellappa, 1997], who suggested that the computational model of the future would be much more closely tied to economic interests than to the limitations imposed by technology. Although almost a decade has passed, this notion could be considered too utopic given that it was the continual advances in technology that drove market and business models. Today, it seems clear that, despite the many obstacles yet to be overcome, technologically speaking, the rate of technological innovation is determined by the macroeconomic interests imposed by the large technological companies, as the Cloud Computing paradigm has shown since its birth in 2007 [Lohr, 2007]. These economic interests have led to the concept of CC being as closely tied to a business plan as to research.

This rapid growth is to a large extent tied to the more sophisticated developments that have been reached by related technologies. These technologies have contributed from different points of view (hardware, software and business models), as shown in Tabla 11.

	Hard.	Soft.	Buss.	Influence
Grid Computing	X			High value add to the end user services
Utility Computing			X	Origin of the business model for the CC paradigm, based on the provision of computational services on demand.
Autonomic Computing		X		Define the basis of self-monitoring and automated control of the computational environment
Virtualization	X			Underlying software environment that allows the rapid provision of services
High Availability	X	X		Grouping data centers in clusters and high availability techniques
Service Oriented Architecture		X		Model the provision of computational services at the software/hardware level.

Tabla 11.- Cloud Computing and related technologies

The increasing importance of this paradigm has led to a vast number and variety of definitions [Mell *et* Grance, 2011] [Vaquero *et al.*, 2009] [Wang *et al.*, 2008] [Foster *et al.*, 2008]. The most generally accepted definition, and in our opinion the most precise from both a technical and functional point of view, is proposed by NIST (National Institute of Standards and Technology [Mell *et* Grance, 2011]. In this definition, Mell *et* Grance propose that *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.*

According to NIST, the services must contain the following characteristics [Mell *et* Grance, 2011]:

- Services on demand, meaning that services, regardless of their type, must be provided automatically and without human interaction according to user demand.
- Availability of services through the Internet, meaning that clients should access the services through the internet and providers, as a result, must use this medium to provide their services.
- Availability of resources, meaning the provider must be able to offer services independently of their demand, using physical or virtual hardware resources assigned dynamically to each resource and reassigned according to demand. In this respect, there are authors such as [Zhang *et al.*, 2010] [Buyya *et al.*, 2009] who speak directly of

high availability services, technology closely related to high availability computing.
- Elasticity, meaning that the different resources should be provided elastically and even automatically according to demand.

Together with the definition, NIST proposes four deployment models and three types of services (*Capabilities*). The types of services (capabilities), which can be any type of computational resource, are presented first:

- *SaaS. Software as a Service.* These services are available when the provider offers its applications to the consumer. The applications are executed directly on the cloud infrastructure. Although this model includes advantages such as ubiquity and the use of light clients, it also entails a set of weaknesses directly related with the fact that the consumer loses control of the infrastructure.
- *PaaS. Platform as a Service.* The provider provides the tools that the users need to create their own applications. These services include programming, libraries, tools, etc.
- *IaaS. Infrastructure as a Service*. The type of capability provided to the consumer is a type of hardware, such as processing, storage, network, etc.

In line with the definition proposed by NIST, the last step required to characterize the CC systems consists of identifying the four proposed deployment models. First is the *Private Cloud* model, which is used when the CC infrastructure is used by a single organization, which can in turn include different consumers. Second is the *Public Cloud*, which is associated with infrastructures for open use by the general public. Based on these two models (public and private), various authors propose the existence of a hybrid models named *Virtual Private Cloud* [Krautheim *et al.*, 2009] [Zhang *et al.*, 2010]. In which private CC are built over the infrastructure services of the public platforms. As a result of this hybrid model, there are multi-holdings [Bezemer, 2010][Zhang *et al.*, 2010] when various CC environments share the same public infrastructure.

As for the other two models, the third is named *Community cloud* and is available when the CC infrastructure is used by a specific group of consumers or organizations that share a common interest. Finally the *Hybrid Cloud* model permits the combination of any of the other three models presented. In this case, we can talk about the interoperability between platforms [Ortiz, 2011], which are generally complex since each CC platform is closed.

7.2.2 REFERENCE MODEL CLOUD COMPUTING

A CC system is undoubtedly complex and is part of an open environment composed of different technologies, users and economic interests that give

way to a new computational model that has revolutionized the way in which services are offered through the internet. For all of these aspects to coordinate and work together to achieve common objectives, bearing in mind any existing technical limitations, complicated architectures have been developed.

NIST also proposes a reference architecture [Liu *et al.*, 2011], shown in Figura 43. The details of the architecture, focusing on the main roles in the paradigm, are presented below:

- **Cloud Provider.** This role is in charge of (i) *Coordination of Services* (IaaS, PaaS, or SaaS), which are provided to third parties and therefore require an underlying infrastructure. This role must also (ii) *facilitate the management of the services* offered through a support layer for marketing and business. Finally it is responsible for (iii) *security* and (iv) *privacy*.
- **Cloud Auditor.** This agent is capable of monitoring the service and ensures that the agreed-upon requirements are being met.
- **Cloud Broker.** This agent acts as an intermediary between consumers and providers. It aims to seek and provides the services best suited for the consumer's objectives, regardless of the provider.
- ***Cloud Carrier.*** Provides connectivity between provider and consumer
- ***Cloud Consumer.*** Who is the end user of the services provided by the CC platforms

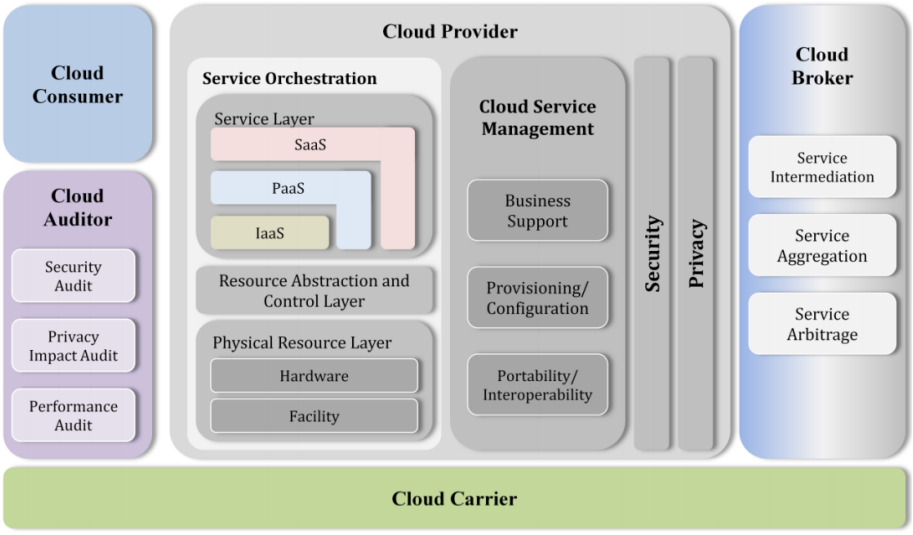

Figura 43.- Reference Cloud Architecture for NIST[Liu *et al.*, 2011]

7.2.3 MULTIAGENT SYSTEM AND THE CLOUD COMPUTING PARADIGM

As noted above, the CC computational paradigm has grown strongly in recent years; its development has led to the advancement of a large number of platforms, both public and private. According to Gartner[72], its wide acceptance within the business world [Mitchell *et al.*, 2012], as well as its easy and rapid integration with traditional technology architectures [Desisto, 2013] has led to the rapid development thus far. Likewise, the pay-as-you-go marketing model [Armbrust *et al.*, 2010] similar to traditional utility products has also been a key factor in its rapid evolution.

The definition provided by NIST [Mell *et* Grance, 2011] stands out among the others since it not only defines the paradigm, but also its characteristics, services and deployment models as well. Nevertheless, in spite of the quality and breadth of this definition, more than a few others [McKendrick, 2012] have noted that the definition is not sufficiently ambitious. There is a vast number of technological innovations that have emerged within the scope of this paradigm [Low *et al.*, 2011] [Azodolmolky *et al.*, 2013] that are nevertheless not a part of existing platforms. The majority simply focus their efforts on providing hardware infrastructure services through the use of the underlying virtual technology, without considering the capabilities of the higher levels such as platform and software.

Given these weaknesses and vulnerabilities, as well as the limitations of CC platforms, and in line with the hypotheses proposed in this study, a MAS framework based on VO has been selected to deal with these obstacles. Although one may initially consider these two distributed systems (MAS and CC) to be incompatible, a detailed analysis demonstrates that they are in fact not only complementary, but share considerable synergy between them. First of all, CC environments can cover the computational needs for persistence of information and the computing potential that MAS require for different applications such as data mining, management of complex services, etc. Additionally MAS can be used to create a much more efficient, scalable and adaptable design for the CC environment than what is currently available. Finally, the use of MAS in the framework of the design for CC systems provides this paradigm with new characteristics such as learning or intelligence, which makes it possible to develop much more advanced computational environments in all aspects (intelligent services, interoperability among platforms, efficient distribution of resources, etc.).

[72] http://www.gartner.com/technology/home.jsp

The number of studies that can be found on the state of the art relating CC with agent technology is actually quite low. However, this tendency is changing and it is becoming increasingly common to find studies and applications focused on this field. Despite the limited number of studies on the matter, **Agent-based Cloud computing**, or the **Agent-based Cloud platform**, is becoming a common concept, mentioned by various authors in recent years [Talia, 2011] [Talia, 2012] [Kang *et* Sim, 2011] [Sim, 2012] [Venticinque *et al.*, 2010].

The present study proposes a new classification from the point of view of CC, based on the reference architecture proposed by NIST [Liu *et al.*, 2011] and the different responsibilities of each of the roles that participate in the CC paradigm as identified in the architecture: *Provider, Consumer, Broker, Carrier and Auditor*. The role of Cloud Carrier is not included in this classification since it is the agent responsible for providing the transportation of information and does not provide any possible functionality between MAS and CC. Figura 44 presents a general overview of the synergy between both paradigms

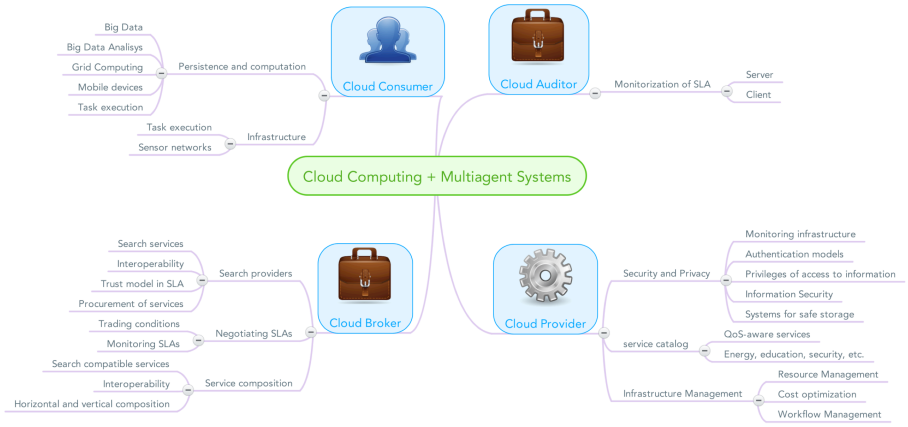

Figura 44.- Cloud Computing and multiagent systems

As shown in Figura 44, when a MAS takes the role of Cloud Consumer, the CC environment offers high performance technology that can be used by the MAS [Talia, 2012]. These services allow and facilitate the application of MAS in a wide variety of complex applications due to the ability to expand the reasoning and knowledge model of traditional MAS. The temporary restrictions are reduced and different high performance storage models are available [Russom *et al.*, 2011] [Decraene *et al.*, 2010] [Chen *et al.*, 2013] [Leitao *et al.*, 2013].

When MAS takes the role of *Cloud Auditor*, or *Cloud Broker*, it serves as a third party (intermediary) that can intervene in the existing business relationship between user and consumers. This ability is widely applied

primarily in the *Cloud Broker* role, significantly highlighting the search features offered by MAS [Sim, 2012a], the selection of services [Alhamad *et al.*, 2010b], and the automated and simultaneous negotiation of agreements with different providers [Aversa *et al.*, 2010] [Venticinque *et al.*, 2010]. An incipient, rapid growth is seen in the use of MAS when arranging cloud services (*Cloud manufacturing*) [Gutierrez-Garcia *et* Sim, 2010].

Finally, when a MAS takes the role of *Cloud Provider*, its main contribution is related to the security and privacy of data, due to the ability of the agents to monitor, provide reasoning skills and respond proactively to the changes in the environment [Venkataramana *et* Padmavathamma, 2012] [Talib *et al.*, 2012a]. However, despite the different related studies that offer CC type services [Yang, 2012] [Chao *et* Sun, 2013], it is only possible to find promising and incipient studies in the state of the art related to the quality of services and provision or computational resources [Wei *et* Blake, 2013] [Nuñez *et al.*, 2012].

The latter group, which includes the most interesting studies for the scope of this doctoral thesis, includes those in which MAS is applied to existing problems or weaknesses in the CC system. These studies are based on the experience from related or preceding studies and have a direct application to CC environments. However, it is also possible to observe that these studies solve partial or specific problems, but do not address the issue of controlling a CC environment or dealing with existing challenges comprehensively (management of infrastructure, SLA agreements, orchestration of services, etc.). As the current state of the art does not include references in which social models are applied in a CC context, the present study will pioneer the application of MAS in the development of a CC platform.

7.3 ARCHITECTURE MODEL

Taking into account the needs and shortcomings detected in the review of the state of the art, this doctoral thesis proposes a new model of a multiagent architecture based on VO and especially designed for the management of CC environments. Prior to formalizing the proposed architecture, it is necessary to formalize the context and the environment in which the proposed architecture will be executed. Given the complexity associated with a CC environment, as well as the different artificial and human components involved in this context, it is necessary to define how the services will be offered at a technical level. For this reason, and following the CC model as presented in Figura 45, each software service for the platform, at the PaaS or SaaS level, can be deployed simultaneously on various virtual machines (nodes or *workers*). This ability makes it possible to elastically configure the resources assigned to each service. In terms of requests for a specific service, the demand is balanced among the different virtual machines that are associated to the service. Additionally, the weight of each virtual node on the scale can vary dynamically in execution time. Therefore, the elasticity is based on modifying the (virtual) resources that have been assigned to each service dynamically according to demand.

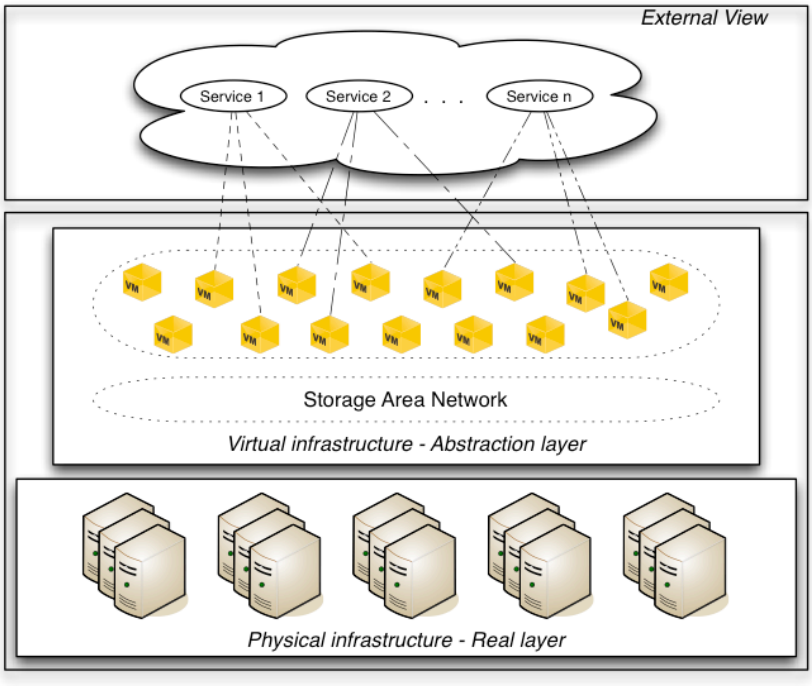

Figura 45.- Deployment model in Cloud Computing

The design of a monitoring and control system in a technology environment, such as that shown in Figura 45, requires the use of AI techniques to be able to incorporate the tasks that allow the dynamic adaptation to the changes and alterations in the demand of the services offered. The dynamic adaptation to changes that occurs in the environment requires learning capabilities, distributed representation of knowledge, and advanced reasoning models. In this sense, the MAS based on virtual organizations allow the incorporation of theories, models, mechanisms, methods and tools that facilitate the development of systems with reorganization capabilities and those that can adapt automatically to future changes in their environment [Rodriguez González, 2010]. Furthermore, this design model permits the external agents to perform services within the organization, which facilitates the incorporation of new functionalities that are not directly developed by the system.

The architecture proposed within the scope of this doctoral study is called +Cloud (*Multiagent System Cloud*) and is based on virtual organizations (VO) of intelligent agents, which in turn allows for the provision of new solutions required by CC platforms for the components to adapt, change, enter and exit. The main objective of +Cloud is the monitoring and ability to control a CC environment, allowing it to automatically and dynamically adapt to the needs at any given time. +Cloud gathers data from the entire CC environment, including the underlying infrastructure as well as the demand for the services it provides. This distributed monitoring model makes it possible to instantly adapt existing resources to the CC environment according to demand for each service, which in turn meets the double objective of complying with the established SLA agreements and reducing energy consumption. One of the most innovative aspects of +Cloud is the design of agents with advanced reasoning capabilities for the distribution of resources, as will be explained in a future section.

In order to model an architecture such as that proposed in this study, it is necessary to have advanced design methodologies. The GORMAS (Guidelines for Organization-based MutlAgent Systems) [Argente, 2008] methodology is used in the present study. It is based on six meta models (agent, activity, interaction, environment, organization and norms), which make it possible to describe any MAS organization from four points of view: structural, functional, social and dynamic. The following sections describe the proposed architecture.

7.3.1 FORMALIZING THE ARCHITECTURE

The proposed architecture is based on organizational aspects and, as such, it is necessary to identify the organizational structure to be used. To do so, the first step involves identifying the components of the architecture, which

permits establishing the interaction model based on an analysis of the needs of the potential system users. Based on this analysis, it was possible to deduce the roles of the users and components that participate in the system and the way they will exchange information.

The development of a monitoring and management system for a CC environment that follows a MAS-based design model differs from traditional models that control this type of platform, which tends to have a centralized decision-making process [Buyya *et al.*, 2010b]. The scope of this doctoral thesis follows an alternative model based on the theory of agents and MAS in which the responsibilities, primarily monitoring and decision-making, are distributed among the platform components. This model allows the decision-making process to be carried out right where the information is gathered, on the base that provides local knowledge, which has made it possible to design agile control processes based on uncertain information, prior knowledge, and the interaction among similar agents. To a certain extent, this unique feature may lead to a situation in which, while the system adapts to demand by following the principle of elasticity of CC systems, some of the agents enter and exit the system according to the life cycle of the physical components where they are located. Figura 46 shows how each one of the agents/roles that participate in the organization is located throughout the entire computational environment.

In following the indicated distribution model, each physical server in the CC environment contains an agent in charge of monitoring (*Local Monitor*) and another responsible for the local level (*Local Manager*). Between the two they have the authority to completely control the physical server (PR) where they are located, which in turn implies a distribution of resources in the virtual machine. However, when the resources must be distributed, which involves the assignment or removal of nodes for a particular service, another specialized agent (*Global Manager*), which is also located in each one of the physical nodes of the infrastructure, is in charge of making these types of decision, which involves more than one physical node on the CC platform.

Following a similar model, each service offered to the users is associated with two agents, one for monitoring (*Service Monitor*) and the other for control (*Service Supervisor*), both of which are in charge of ensuring compliance with the previously established SLA agreement. They are physically located in the node that balances the work-load among the different worker nodes, which permits them to have precise information available to make the correct decisions at their level. In this sense, the tasks for this level are related to the workload balance among the different nodes, error detection and, most importantly, monitoring the quality parameters for the service.

There are also other agents with very different tasks located in the entry point of the CC system. First, two control agents, the first of which is in charge of controlling the hardware infrastructure (*Hardware Supervisor*), its state, and the starting or stopping of the physical servers according to demand. A supervisor agent is the global controller (Global Supervisor) that ensures that the remaining components and agents function correctly and in accordance with their specification. Finally, there is also an agent in charge of establishing service agreements with the platform users (*SLA Broker*), which can negotiate the QoS level of services according to user needs and the state of the system at any given moment. It should be noted that this aspect of the CC paradigm extends beyond the scope of this research project and is considered part of future work to be carried out. Nevertheless, the state of the art includes a great variety of techniques and algorithms, some of them based on MAS [An *et al.*, 2010] [Alhamad *et al.*, 2010a] [Son *et* Sim, 2012] [Sim, 2010] [Venticinque *et al.*, 2011].

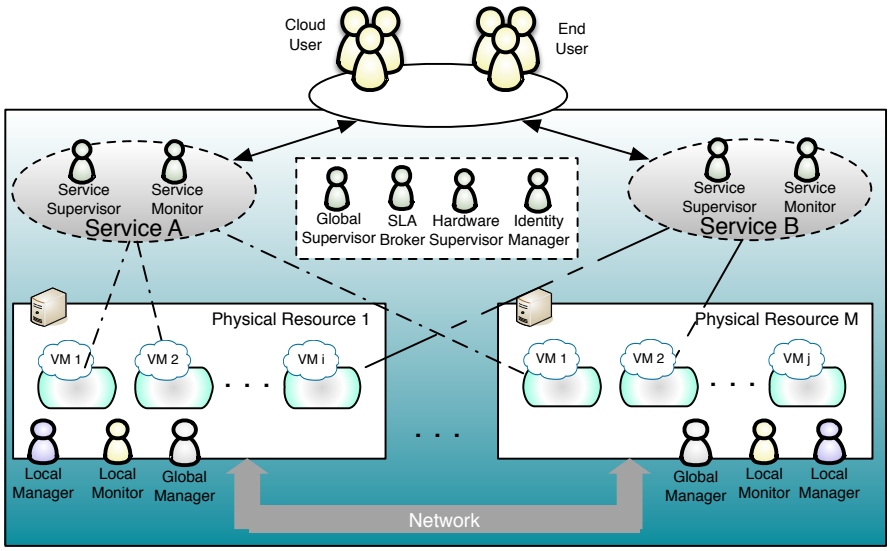

Figura 46.- Agents distributed over the infrastructure

Finally the system also includes an intelligent agent linked to the human users with the aim of simplifying the user's interaction with the system. The agents that are linked to external (human) entities are the *Cloud User* and the *End User*. The *Cloud User* agent is linked to the *Cloud Consumer* role according to the architecture proposed by NIST [Liu *et al.*, 2011]; in other words, it consumer the services and products provided by the CC system, which in this case are persistence and deployment for web applications. The *End User* agent is the end user of the applications deployed by third parties in the CC system. Additionally, we have considered the existence of another agent, called

Identity Manager, which is linked to the entity in charge of managing the entry and exit of users and their affiliation with agents within the system.

Given this identification of agents and the roles that participate in the system, it is possible to design an organization that is unified and intuitive, and contains a high level of abstraction [Agüero *et al.*, 2009]. In line with the guidelines indicated in the GORMAS methodological guide, one of the first tasks is to instantiate the functional view (*mission*) of the organizational model, which is shown in Figura 47. This view presents the products and services offered by the system, the global objectives to pursue (mission and justification) and the affected interest groups.

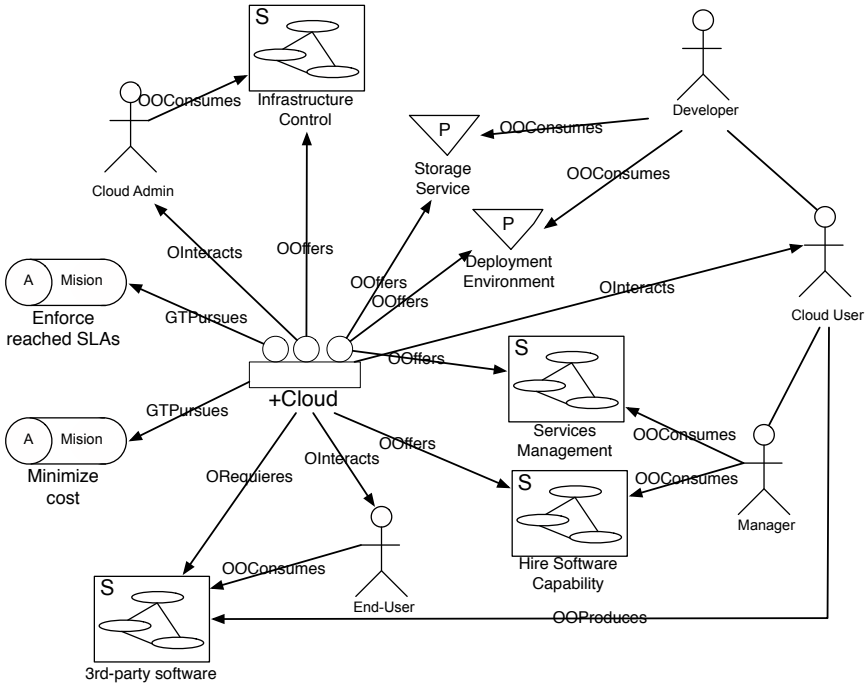

Figura 47.- Functional view (mission) of the +Cloud organizational unit

Thus, the first mission and reason for the existence of an organization will be to comply with the service agreements agreed upon with the *Cloud User* role, while minimizing the costs associated with this mission. The diagram indicates the types of users that use the system (*Cloud Admin*, *Cloud User* and *End User*) and the products that are offered (storage and deployment of software). In order to facilitate the interaction of the platform, the following intrinsic services are also offered: software management, software hiring, and infrastructure control. Among the services offered, it should be noted that the platform also offers as a service those applications that can be deployed in the system by third parties (*Cloud User*); in other words, these types of applications

are required by the platform to justify the need to offer storage and infrastructure products. However, given that a CC platform is a simple means (and not an end), these applications are also services that are offered to the *End User*.

According to the GORMAS methodology, it is necessary to specify four organizational dimensions: departmentalization, specialization, coordination and normalization. Based on this analysis of the organizational dimensions, the departmentalization and coordinating mechanisms within the system are analyzed. The *coalition* organizational pattern is selected as the most appropriate to model the system, since the various entities that compose a structure of this type will form groups according to functional similarities, and will coordinate among themselves to offer the most complex and elaborate functionality; this description is perfectly suited to the coordination model pursued in +Cloud. Using this pattern, the main organizational unit (+Cloud) is departmentalized in the following specialized organizational (sub) units.

- *SLA Negotiation.* In charge of grouping tasks to establish agreements and negotiations with the external *Cloud User* role.
- *Service Management*. Is assigned with the monitoring and supervision tasks for any products offered, and for overseeing the agreed-upon quality agreements for those products.
- *Infrastructure Management*. Controls the underlying hardware infrastructure, i.e., the system's computational resources (real or virtual) by monitoring the resources.

Finally, the last noteworthy product from the architecture's design will be described within the environment in which the proposed organizational MAS is located. The environment is dynamic, complex, uncertain and hostile. These characteristics define the type of environment in which MAS, especially those based on VO, are the most effective. Figura 48 provides a complete view of the model of the environment; the different system roles interact with the environment to achieve their individual objectives through the provided ports, whether to simply read or modify.

- Two applications that use interfaces to operate with roles external to the system. In other words, the application that can monitor the underlying infrastructure of the CC environment, especially designed for the *Cloud Admin* role and the web desktop, which has specific applications for the *Cloud User* role. The end users access applications directly from third parties that are deployed in the system.
- Three repositories associated with the information that should be persistent, the user´s storage, the agreements established with the users, the real hardware available, and the user's history.

- The figure also shows the different ports to access each of the resources for the environment. We can see the access ports to the resources that have been previously identified (repositories and interfaces), as well as the three ports that can reconfigure the underlying infrastructure (control of the hardware environment, virtual environment, and balancing systems).

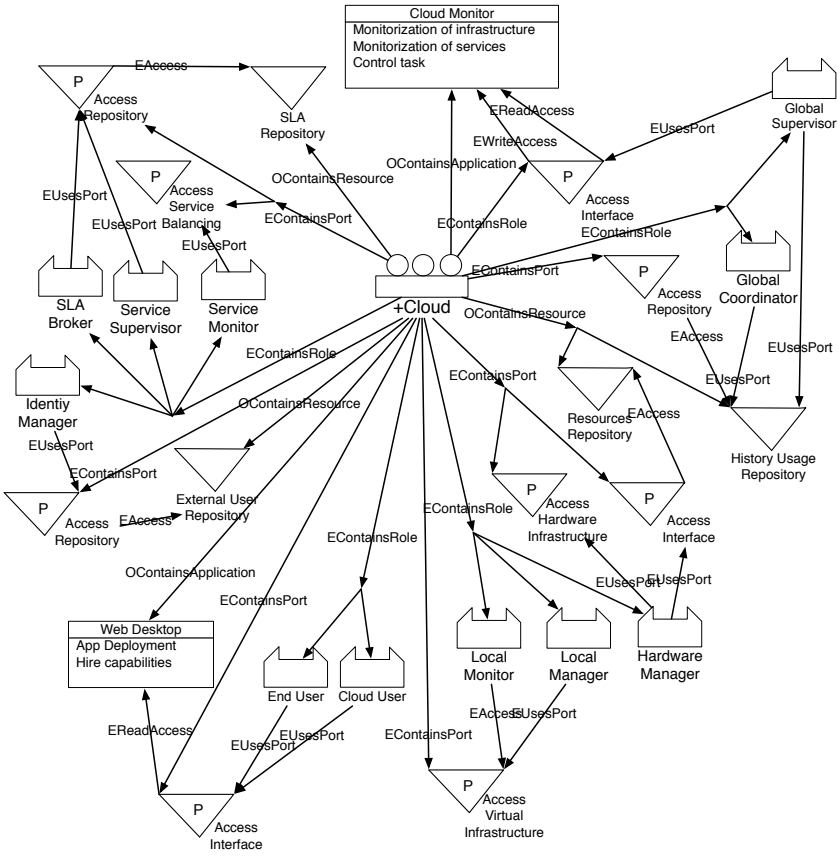

Figura 48.- Model of the environment. Access to the the ports in the environment

In conclusion, the design of the architecture based on the GORMAS methodology can provide a set of artefacts that can specify the system's design. This process has made it possible to identify, from an external level, the mission, services and products provided by the system. Additionally, from an internal level, it has also been possible to identify three organizational units, their corresponding agents, means of interacting with the environment (communication interfaces with the users and access ports to the environment), as well as the interaction and negotiation model used. On the basis of this architecture, the following chapter presents the reasoning model

used by the individual agents that participate in the society to carry out the distribution of computational resources. These algorithms fall in line with the philosophy of MAS and are based on the autonomy of the components, such as distributed decision making. With this model it is possible not only to optimize the resource distribution process, but also to improve system availability. This resource distribution model is innovative in and of itself; since current models are centralized and do not, therefore, consider the system's dynamic self-adaptation in response to changes produced in the environment.

7.3.2 RESOURCE DISTRIBUTION MODEL

The goal of this section is to describe the types of agents specialized in the distribution of computational resources. Since their lifecycle is tied to the machines in which they are located, some of these agents can also describe a self-adaptive model of the +Cloud architecture. This model can handle the demand for services offered on the CC platform without breaching the established SLA agreements.

The distribution of resources within these computational systems is a complex task, as the underlying infrastructure offers many possibilities. Thus, it is possible to distribute resources at a local level, among each individual service or physical node, or at a global level, among different physical nodes. Consequently, the correct distribution of computational resources in a CC environment is a key decision given that the distribution can maximize the efficiency of the environment and reduce associated costs. However, the task is complex since it requires an analysis of current and future demand for services as well as an analysis of resource availability at any given time in the CC environment. Therefore, the distribution of resources can be presented from different points of view according to the capabilities permitted by the underlying technology:

- **Distribution at the service level**. The distribution of resources at this level requires a simple balance of the workload among the worker nodes that are offering a specific service. This balancing has been highly extended from the birth of cluster computational systems in HPC (*High performance Computing*) environments [Aikins *et al.*, 2007].
- **Distribution at the infrastructure level.** The distribution at the infrastructure level is a more complex and novel method which was made possible after the birth of virtualization technology [Oguchi *et* Yamamoto, 2008]. This technology permits much stronger, dynamic and automated tasks by incorporating nodes that are virtual and can be placed in any physical server.

The new reasoning models integrated in the different specialized agents that constitute the multiagent architecture proposed in this doctoral these can

be described according to the two possible methods for redistributing resources offered in current technology (services and infrastructure). The new distribution model follows a hierarchical process starting with a distribution of resources performed at the service level which balances the different requests among the nodes affiliated with the service. When the balance of requests is insufficient, meaning that the nodes that offer the service cannot handle the high demand for requests, the system performs a distribution of the infrastructure from (i) a micro perspective. From the micro perspective the system asks the different physical machines containing the nodes for a specific service to provide more computational resources to each of the nodes; which increases its performance. Finally, in a third phase, when the second distribution process has not been satisfactory, a new distribution of the infrastructure is performed at a (ii) macro level, which allows instantiating a new node that can be hosted by an active physical machine in the CC environment, or by one of the machines in sleep mode.

7.3.2.1 REDISTRIBUTION AT THE SERVICE LEVEL

This is the most simple of the proposed types of balancing, although it has a key role in the new distribution model since it is in charge of ensuring that the SLA agreements are in compliance with regard to the software services agreed upon with the users of the CC platform. There are two agents in this redistribution specialized in monitoring and supervision tasks (see Figura 49): the *Service Monitor* and the *Service Supervisor*.

- The *Service Monitor* agent is in charge of controlling the state of the balancing system. To do so, it implements algorithms that calculate the Quality of Service (QoS) on a continuous basis for each one of the nodes that is being balanced at any moment. To do so, the agent periodically recalculates the weight of each node within the service, applying the following algorithm:
  1. Maintain a set of routing groups, each of which is linked with different worker nodes with similar QoS levels
  2. The nearest neighbor grouping algorithms is used to determine the QoS groups, based on the calculation of the distance between nodes using the Euclidean distance between quality of service vectors in each node $QoS_{n_i}^k = \{\overline{r_1^k} \cdots \overline{r_i^k} \cdots \overline{r_n^k}\}$.
  3. In order to assign a weight to each group, which is an inexact process, an iterative process is used. Initially, all groups have the same weight; however it is adjusted progressively according to the demand that each group can accept. Thus, the groups with the best QoS have a higher weight, while those with a lower QoS have a lower weight. After a period of stabilization, the process is repeated starting with the previous point.

- The *Service Supervisor* agent has a key role given that it must determine, according to the data generated by the *Service Monitor*, whether it is necessary to assign more or fewer resources to the service:
  - It must first decide when to remove the subset of resources associated with the service. During this process one of the nodes affiliated with the service will be eliminated. After removing an execution node, the remaining nodes associated with the service will have to respond to the demands associated with the eliminated node. This increase in the workload of each of the nodes has a corresponding period of stabilization in which the *Service Monitor* recalculates the new QoS parameters.
  - More resources are associated with a service when the existing nodes, and the resources that have been assigned to them, are unable to respond to the demand for requests according to the SLA agreements established with the users. In order to do so, a distribution of resources is first initiated at a micro level. If the problem is not resolved at this step, the distribution of resources can be initiated at the macro level.

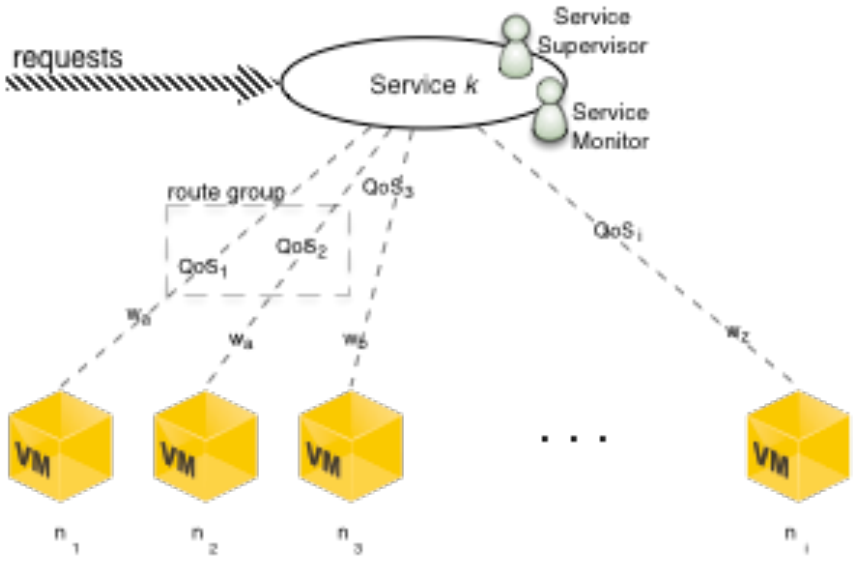

Figura 49.- Redistribution of resources at the service level

7.3.2.2 REDISTRIBUTION AT THE INFRASTRUCTURE (MICRO) LEVEL

Redistribution from the point of view of the infrastructure at a micro level is associated with the redistribution that occurs in a physical machine where different resources are assigned among the different virtual machines it houses, ensuring enough resources are set aside for the principle server. This

process is managed innovatively in the VO-based multiagent architecture proposed in this study and is managed by two specialized agents: the *Local Monitor* agent and the *Local Manager* agent. The new process works as follows:

The *Local Monitor* agent is in charge of retrieving information about the state of each virtual machine and the server where they are located. However, under no circumstances is it possible to modify the characteristics of the resource assignment.
The monitor agent will update a matrix with the following information about the physical server and each of the virtual machines it hosts

$$exec^{e_1} = \begin{Bmatrix} VM_1 \\ \vdots \\ VM_m \\ PR^{e_1} \end{Bmatrix} where \begin{cases} PR^j = \{hostname, IP, mac, state, M_{max}, vcpu_{max}, M_{min}, vcpu_{min}\} \\ VM_i = \{IP, state, M, vcpu, M_i\, \overline{p_{cpu}}\} \end{cases}$$

- The *Local Manager* agent can oversee the assignment of resources to manage the resources within the machine, including the following functionalities: (i) update the resources assigned to each of the virtual machines, (ii) instantiate a new virtual machine according to a template, and finally (iii) detain the execution of a virtual machine. With these three functionalities it is possible to redistribute resources within a physical server. The resources that can be redistributed are the number of virtual processors (*vcpu*) and the physical memory (*M*).

. When the process of distributing resources is initiated, the *Local Manager* agent in the machine configures an instantiation of the state of the machine according to the information provided by the *Local Monitor* agent. The distribution of resources is different for assigning virtual *vcpus* and memory. To begin, when assigning *vcpus*, the goal is not to have underused execution resources; if there were any, they could only be assigned to a physical machine. The process of assigning *vcpus* is initiated by verifying whether the physical machine has *vcpus* that have not yet been assigned. If there are no unassigned *vcpus*, the agent Local Manager iterates over every virtual machine to determine whether the amount of processing that has been assigned is being underused. If the value $vcpu_{used}$ is less than the constant supervised by the administration, one of the cores from that machine is reassigned to another machine that needs more resources.

The process of reassigning memory is more complex and, as such, requires a linear programming model to determine the most optimal assignment of resources.

$$\begin{cases} VM_i(M_{asigned}) = VM_i(M'_{asigned}) + VM_i(M_{max}) * priority_i \;\; if \;\; (VM_i(M_{max}) * priority_i) < \; PR^{e_i}(M_i) \\ VM_i(M_{asigned}) = VM_i(M'_{asigned}) + PR^{e_i}(M_i) \;\; if \;\; (VM_i(M_{max}) * priority_i) > \; PR^{e_i}(M_i) \end{cases}$$

If there is no free memory, it must be reassigned; in other words memory from one set of instances of execution must be taken and reassigned to the

requesting machine. Thus, at a technical level, it is possible to redistribute the memory that is being underused while maintaining a margin of security (M_l) by removing memory from an active node. The problem lies in how to redistribute the underused memory among the different virtual machines and the physical server, so that the amount of underused memory is reduced. The solution is obtained through optimization and is solved by using linear programming so that there are *m* independent variables that correspond to the new memory assignment. The function that must be minimized is composed of the sum of the underused memory that will be assigned to each virtual machine according to the actual percentage of underutilization; this makes it possible to determine what can be gained by assigning memory in each case.

$$MIN\ FR^{e_1}(M_i) = \sum_1^m x_i \left({VM_i(M_i)} \Big/ {FP^{e_1}(M_i)} \right)$$

This minimization model has the following two restrictions:

- The sum of the objective variables will be equal to the sum of the initial underused memory.

$$\sum_1^m x_i \leq PR'^{e_1}(M_i)$$

- It will also be necessary for each variable x_i to be greater than the margin of security.

$$x_i, \ldots, x_m > M_l$$

Once the new distribution of memory has been determined, if the results obtained during the process of calculating the new assignment are greater than the resources of the service machine that initiated the change, then the new assignment will be applied by the *Local Manager* agent, and the *Service Supervisor* agent associated to the service that initiated the process will be informed that the node has received a new resource assignment.

7.3.2.3 REDISTRIBUTION AT THE INFRASTRUCTURE (MACRO) LEVEL

The new model for the redistribution of resources at the macro level will be initiated by a *Service Supervisor* agent when it detects that the resources assigned to the service with which it is associated are insufficient. The redistribution of resources at a macro level is performed by the *Global Manager* agents, which have greater authority than the *Local Manager* agent and can inform that agent of the need to start up a new machine with a specific service and specific characteristics (see Figura 50).

The *Global Manager* is a highly specialized agent that can implement a CBR-BDI [Corchado *et* Laza, 2003] [Corchado *et al.*, 2008] deliberative architecture. As a result, the reasoning process in each physical node is based on past

experience gained from storing similar cases. The case memory is central to the entire CC system; the system's global knowledge can be shared by each of its members, in this case the *Global Manager* agent. Given that this memory can grow exponentially as a maintenance strategy, a high-speed schema-less database is used to provide fast access to the stored data.

As previously indicated, the infrastructure distribution process at a macro level is initiated by the Service Supervisor agent associated to the service that is experiencing difficulties in responding to demand. This agent alerts all the physical machines that host service nodes that a global level redistribution of resources is required to handle the demand for requests. The message is received by the *Global Manager* agent from each physical server. This agent, in turn, alerts the remaining *Global Manager* agents in the CC environment. These agents ask the *Local Monitor* agent from the machine to instantiate the state of the user of resources (I_{ei}^{t}), in order to evaluate the amount of resources available. If the amount of resources is greater than the minimum indispensable resources required to instantiate a node associated with the service (information which is specified in the service level), then the reasoning process, explained below, will be initiated to determine the amount of resources than can be reserved for each execution node as requested by the Service Supervisor for that service. If there are no available resources, or if they are less than those demanded by the service, the *Global Manager* agent will determine that the physical node is not part of the global assignment process. If there is no machine that can respond to the increased service needs, the Hardware Manager agent will be asked to start up a new physical server to instantiate a virtual machine according to the minimum characteristics defined in the service level.

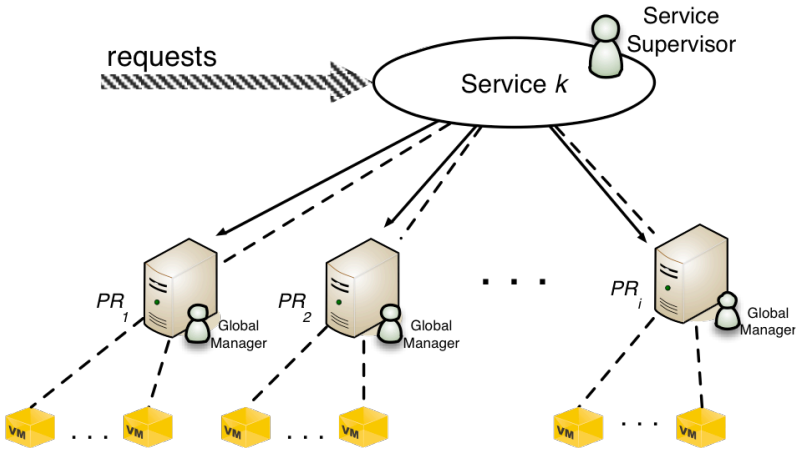

Figura 50.- Initial distribution of the infrastructure at the macro level

To begin, the concept of case must be defined within the reasoning model $C = \{P, S(P), E\}$ where:

- P corresponds to the problema description, which has a matrix-matched representation associated to the instantiation of the use of resources, I_{ei}^{t}, along with the description of the service level that will be instantiated and the temporary indicator that identifies the instant that the problem has been detected.

$$P = \{I_{ei}^{t} \quad VM_{t}^{k} \quad timestamp\}$$

- $S(P)$ is associated with the solution to the problem: $S(P) = \{M, vcpu\}$ in terms of memory and *vcpu*.
- Finally, the efficiency (E) is measured measured from both perspectives (micro and macro):
  - First, the efficiency at the micro level (E_m) is associated with the degree of efficiency of the proposed solution within the physical server where the virtual machine has been instantiated. This degree of efficiency is proposed by the *Local Monitor* agent according to the usage rates of the processor and the allocated memory.

$$E_m = \left\{\overline{p_{cpu}}, M_{used} \Big/ M_{assigned}\right\}$$

  - The efficiency at the macro level (E_M) is associated with the degree of efficiency from the point of view of the service, and is calculated if the application of the proposed solution requires the distribution process of the infrastructure resources to be initiated at a macro level. In this sense, the degree of efficiency measures the number of additional nodes required by the service.

$$E_M = \{\mathrm{n}\}$$

  Therefore, the efficiency is given by the following expression $E = \{E_m \quad E_M\}$

The CBR (Case-Based Reasoning) process is initiated and retrieves similar cases from the case memory. The most similar cases are selected according to the following parameters:

4. Select the cases from the physical machines with similar characteristics, with a degree of efficiency greater than 90%. The similar physical machines are evaluated according to the *benchmark* parameter, which characterizes each physical machine.
5. Based on this subset of retrieved cases, a vector is configured for each case that contains the same number of virtual machines that are in the case, and the free resources available $C_i = \{n, M, vcpu\}$.

6. The cases selected from this subset are those that previously used the same service that is now requesting resources, and during a similar period of time as the current case. This is determined by analyzing a network usage pattern over a period of one week [Abraham, 2003].

A solution to the problem, which is based on the retrieved cases, will be prepared during the reuse phase:

- If the case base does not contain a previous similar case, the solution to the problema will be associated to the minimum resources determined at the level where the service is instantiated:

$$S(P) = \{M_{min}, vcpu_{min}\}$$

- If, on the other hand, similar cases are retrieved, the solution to the problem will be the closest case multiplied by the case efficiency:

$$S(P) = \{\mathrm{M}' * (E_m(1) + E_M(1)), vcpu'(E_m(2))\}$$

- If the values assigned to the previous solution are greater than the values assumed by the machine, due to the fact that there are not many resources available, the result of the case will be the maximum amount of resources available in the machine.

$$S(P) = \{PR(M_{\max}), PR(vcpu_{min})\}$$

Once the solution to the case has been calculated, it is sent to the *Service Supervisor*. This agent reactively selects the node that offers the most resources at the virtual machine level. During the subsequent revise stage, the new node will be instantiated and its use evaluated from a micro and macro perspective, thus providing the value of efficiency for the solution. Finally, during the final state of the proposed CBR cycle, the case and its corresponding efficiency will be stored for future use.

The proposed adaptation model is distributed, which makes it possible to improve the high availability of the system, since the decision making process is made throughout the entire CC system. Furthermore, this model can distribute the strength of the calculation, which requires obtaining the solution; as a result, the impact that the search for a solution has on the CC environment is reduced.

7.3.2.4 COMPACTATION

The proposed distribution of resources model has not yet made mention of the use of key characteristics of virtualization technology, which involves the migration of machines between physical servers. In fact, this characteristic is very effective when compacting virtual machines into the smallest number of servers possible, allowing the set of physical servers that have not yet been

assigned a machine to be turned off or in sleep mode, which significantly increases energy efficiency.

The problem starts when the machines are widely dispersed, which can occur when a *Service Supervisor* agent associated with a specific service detects various nodes that are at the highest priority level (they have a high quality of results) and requests the random elimination of one of these nodes.

The compaction process is simple and is applied by the specialized *Global Manager* agent as follows:

4. A virtual server with a very low number of virtual machines (the number is determined by the system administrator) will ask other servers to appropriate (or host?) its virtual machines so that it can go into hibernate mode and not use any resources.
5. L The nodes with available resources at the time of the request will evaluate the snapshop(I_{ei}^{t}) provided by the *Local Monitor* agent to determine whether it can host another machine with the given characteristics.
6. If there are resources available, the configuration is sent to the *Global Manager* that has made the request and the migration process is initiated. If this agent receives various confirmations simultaneously, it will randomly initiate the migration process with one of the machines. The process is random since the agent does not know all the internal details of the machine that will host the new virtual node.

This simple process makes it possible to compact the set of virtual machines without affecting the quality of service, since the individual resources of each machine are not modified. The system then goes into a *compact* state.

7.4 CASE STUDY

The evaluation of the proposed dynamic distribution model is a thorough task and requires a hardware and software environment specially adapted to its needs. The evaluation and validation of the model for this study will be done through a CC platform developed within the scope of the research done by the BISITE research group, and will include different computational services at the hardware and software level. From the beginning, this platform was conceived to integrate the proposed MAS +Cloud. As a result, the ability to reorganize and to adapt agent behavior are necessary functionalities for the platform to operate correctly.

In order to evaluate the proposed multiagent architecture, a series of experiments were conducted with the aim of simulating the behavior of an organization and its members in a real adaptation case. The results obtained from these experiments have made it possible to empirically evaluate whether the dynamic system responds according to its specification, dynamically adapting according to the state of the environment and the demand for services. Upon verifying the proper functioning of the organization in the simulation, the next step was to evaluate the behavior of the reasoning models that enable the dynamic adaptation of the organizational MAS. This was done through the distribution of the infrastructure resources in the CC platform among the different services offered in response to user demand.

This case study used the +Cloud platform, which was deployed in the HPC environment of the BISITE research group and composed of 15 latest generation machines that support virtualization in the hardware with the use of Intel-VT technology and the KVM virtualization system.

The case study is based on a simulated *Denial of Service* (DoS) attack [Needham, 1993] using methods that expose the platform for persistence of files. The *GetSize* method is a complex function that uses recursion to calculate the sum of the size of the files contained in a directory. In contrast, the *GetFolderContent* method is a much more simple function that only returns the identifiers for the files or directories contained in the path provided as a parameter.

All of the experiments presented below use the same initial state as represented in Figura 51, where the file storage service is deployed in different nodes (VM1 and VM2), each one hosted by a different physical machine (VM1 and VM2 respectively). This starting point, in addition to being didactic and easy to understand, reflects the typical deployment of any service in a CC environment. Likewise, Figura 51 also shows the main agents that intervene in the readaptation process in the case study.

With regard to the distribution of the initial resources, the file storage service is deployed in different nodes (VM1 and VM2), each one hosted by a different physical machine (VM1 and VM2 respectively). As a result of this deployment, the service has a high availability (it is deployed in two servers), but it is at the same time located in physical machines with different computational loads, which is what occurs in a real environment, since both physical machines host other virtual machines that correspond to other services from the CC platform. In other words, the physical server PR1 has many available and unassigned resources, while PR2 has no available resources and the machines it hosts have a high computational load.

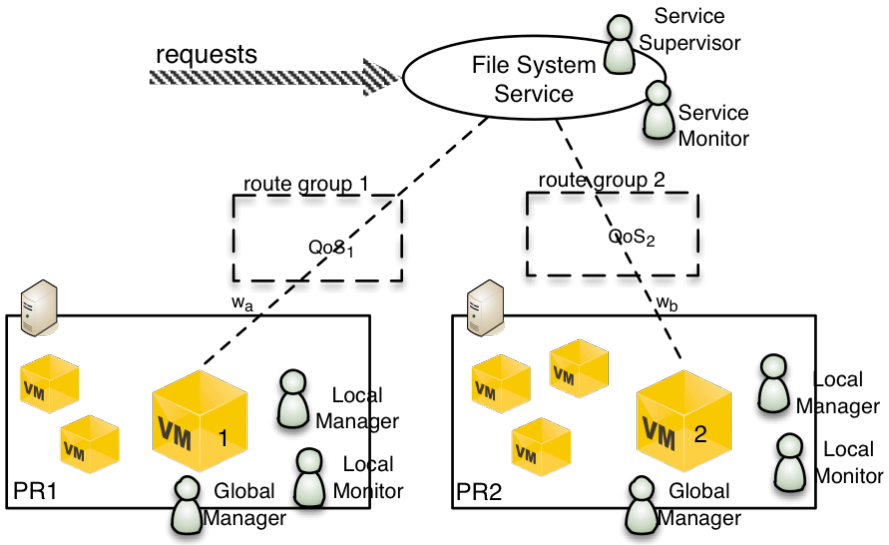

Figura 51.- Initial state of the evaluation case study

7.4.1 EVALUATION OF THE DISTRIBUTION OF THE INFRASTRUCTURE AT MICRO LEVEL

In order to evaluate the readaptation of the infrastructure at a micro level, the first experiments are conducted during which an execution thread is progressively launched every second, up to a maximum of 10 seconds (10 threads). In this case, each thread continuously queries the *GetSize* web service from the file storage service. As previously indicated, this method is complex, since it performs recursive internal queries to find out the size of the object (file or directory). The QoS for the service is considered to be adequate when the response time to a query does not exceed 2.5 seconds.

The result of the experiment is presented in Figura 52, which shows a graph with the response time for the *GetSize* method for the duration of the test.

Once the system detects that the QoS level in the service has decreased, that is, when the average response time is greater than 2.5 seconds, it automatically initiates an adaptation process for the infrastructure at a micro level. As indicated in Figura 52, once the auto-adapt process is complete and the value of the weights have been adjusted, we can see that the response time for the service returns to a value less than the acceptable QoS levels (less than 2.5 seconds).

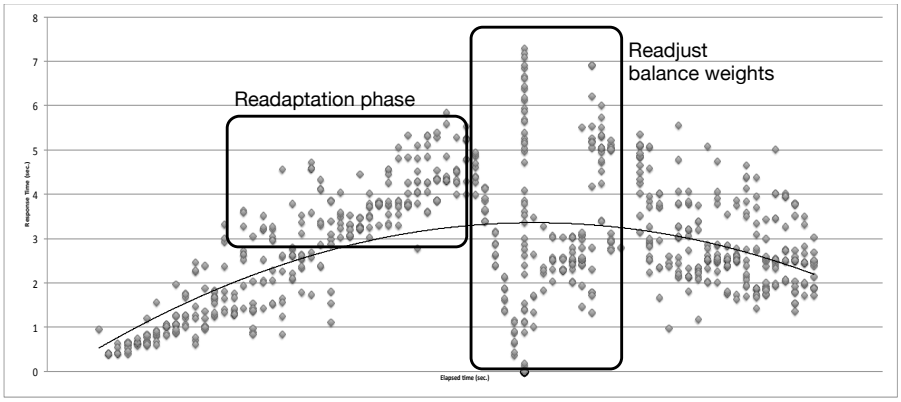

Figura 52.- Experiment 1: Readjust the infrastructure resources at the micro level (*GetSize* method)

We will now provide a detailed description of the process that was carried out during the distribution of infrastructure resources at a micro level. The algorithm begins when the *Service Monitor* role detects that the QoS level associated with the service has reduced progressively, exceeding the acceptable minimum response time (2.5 seconds). The *Service Supervisor* then uses the QoS data for the service and decides to initiate the process for the redistribution of infrastructure resources at the micro level. The complete process is shown in Figura 53.

The *Service Supervisor* agent first sends a message (Step 1, Figura 53) to each of the *Local Manager* agents for each physical machine (PR1 and PR2) that hosts the worker nodes for the service in order to inform them that the persistence of files service (FSS) needs more resources. This message will also indicate the weight for each node in the request balancing process. Each *Local Manager* agent uses this information independently to determine the amount of additional resources that it can allocate to the service. To do so, the *Local Manager* agent for each machine asks its *Local Monitor* counterpart agent for information about the state of each machine; in other words, the instantiation I_{PR}^{t} that characterizes the physical equipment. The *Local Manager* agent retrieves all the local information that it uses to carry out the decision-making process, then the next step is to determine how many resources (*vcpu* and memory) it can provide to the host node, which needs a greater amount of

resources to handle existing demand. Once the reasoning process has finished, the two *Local Manager* agents located in PR1 and PR2 individually notify the *Service Supervisor* agent of the results of the provision of services process (PR1 provides more resources while PR2 does not).

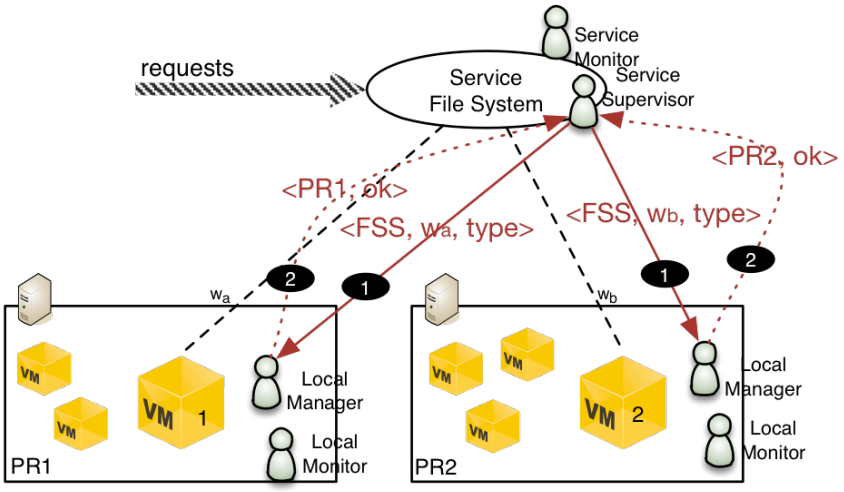

Figura 53.- Experiment 1: Exchange of messages in the distribution of infrastructure at the micro level

Once the individual reasoning process for each physical server has finished, the two *Local Manager* agents located in PR1 and PR2 individually notify (Step 2, Figura 53) the *Service Supervisor* of the results for the provision of services process. Once this information has been received, the *Service Supervisor* will not perform any additional action, since at this point at least one of the physical nodes (in this case both) has provided more resources for the FSS service. In the meantime, the Service Monitor agent balances the weights to adjust demand to the ability of the nodes to provide the new resources. According to this new adjustment in the weights of the nodes, the Service Supervisor will not take any additional action if the average response time of the service decreases (less than the threshold of 2.5 seconds, as was the case). However, if the problem persists, a new adaptation process at the macro level will be carried out.

7.4.2 EVALUATION OF THE DISTRIBUTION OF INFRASTRUCTURE AT THE MACRO LEVEL

For reasons of clarity, only the adaptation of the infrastructure at the macro level is evaluated during the second experiments; no previous consideration of the micro level, as explained in the previous section, is taken into account. Once the *Service Supervisor* detects a decrease in performance, it

directly executes the adaptation process at the macro level. This type of adaptation occurs when the demand for the service is much greater; an increase in the computational load results in a more aggressive increase of the load. In this case, 10 threads are launched every three seconds, to a maximum of 40 threads. The acceptable QoS level for the *GetSize* function in this experiment remains 1.5 seconds, while the threshold QoS level for *GetFolderContent* is set at 0.5 seconds.

The process for exchanging messages among agents during the adaptation of the infrastructure at the macro level is shown in Figura 54. The adaptation process is initiated in the same way as the distribution of infrastructure resources at the micro level; however, in this case, the specialized *Service Supervisor* agent that initiates the service also sends an alert (step 1, Figura 54) to the *Global Manager* agent for each of the physical machines that host the service nodes. We should recall that the *Global Manager* agent is a specialized agent that uses a CBR-BDI reasoning process [Corchado *et* Laza, 2003] [Corchado *et al.*, 2008] in charge of the distribution of resources at a macro level. Once they receive the initial alert, these agents resend the alert message to the remaining *Global Manager* agents in the CC system (step 2, Figura 54).

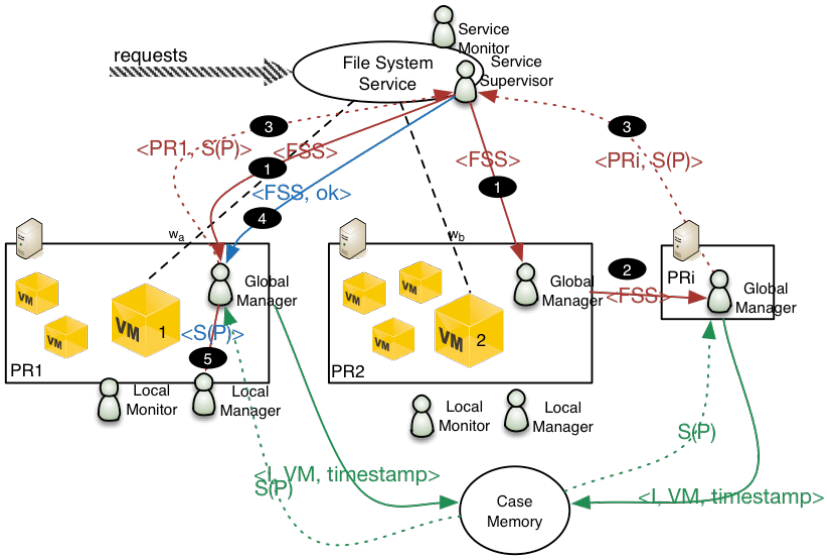

Figura 54.- Experiment 2: Exchange of messages in the infrastructure distribution at the macro level

The next process is carried out in parallel in each of the physical nodes in the CC system. Each of the *Global Manager* agents that has received an alert message asks the *Local Monitor* agent of the machine in which they are located to instantiate the state of the system (I_{FR}^{t}). The agent uses this information to determine if there are available resources to instantiate a new node associated

with the service requesting resources. If the machine, in response to the instantiation, detects that the physical server does not have any available resources, it does not perform any action. This process is conducted in parallel in each of the machines that has available resources.

This problem description is used to retrieve similar cases from the case memory. Each *Global Manager* independently determines which resources it can relinquish to the new node that has just been instantiated. This new provision of resources is itself the solution to the problem for the current case, $S(P)$. The solution provided by each of the CBR-BDI agents from each physical server with available resources is then sent to the *Service Supervisor* agent that initiated the process because of its own performance difficulties (step 3, Figura 54). Once it has received the set of proposed solutions from the different CBR-BDI agents, the agent overseeing the service sends an acceptance message to the *Global Manager* agent that offered the greatest amount of resources for the new node that must be instantiated (step 4, Figura 54). Finally, the *Global Manager* agent that receives the request asks the *Local Manager* (step 5, Figura 54) from its machine to instantiate a new virtual node on the level of the virtual machine associated with the service, according to the proposed problem solution. Once the new execution node has been launched, it is then necessary to evaluate the proposed solution. The *Local Manager* evaluates the solution according to the degree of underused resources in the node that has just been instantiated. If a new resource distribution process must be conducted at the macro level, the *Service Supervisor* agent will complete the evaluation performed by the *Local Manager* agent in order to penalize the solution. In both cases, the efficiency of the proposed solution will be evaluated according to the amount of underused resourced of the new instantiated node. The results in terms of QoS can be seen in Figura 55 and Figura 56, which also show an increase in the quality level after the adaptation has been completed.

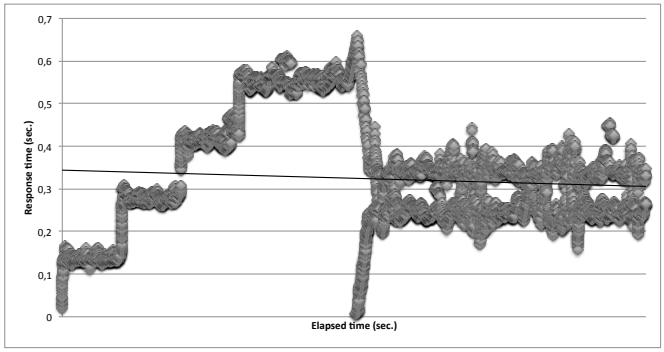

Figura 55.- Experiment 2: Readjustment of the infrastructure resources at the macro level, adaptation 1 (Method: *GetFolderContent)*

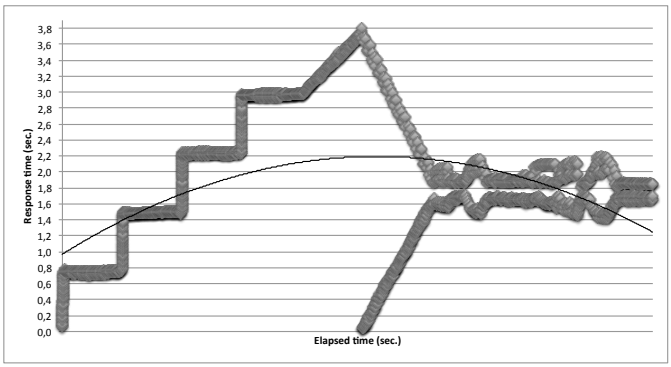

Figura 56.- Experiment 2: Readjustment of the infrastructure resources at the macro level, adaptation 1 (Method: *GetSize*)

The case study was repeated numerous times, which made it possible to store a good number of past experiences in the case memory. This process determined that the reasoning process is strongly influenced by the content of the case memory with regard to quantity and quality of existing cases, those corresponding to past experiences, and produces better results in terms of adapting the infrastructure at a macro level.

Although in the two previous cases the problema gets resolved, the QoS level gets very close to the established margin indicating low quality (0.5 in the *GetFolderContent* case and 2.5 in the *GetSize* case). However, as presented inFigura 57 and Figura 58, when there are many cases in the memory and a number of past experiences similar to the current problem, the adaptation results are actually better because the QoS level is lower. In this case, there is also the disadvantage that the reasoning time is greater because there are greater computational costs associated with retrieving the cases and searching for the best solution among them.

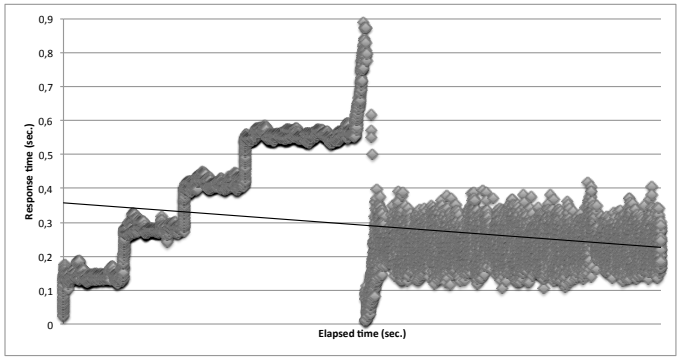

Figura 57.- Experiment 2: Readjustment of infrastructure resources at the macro level, adaptation 2 (Method: *GetFolderContent)*

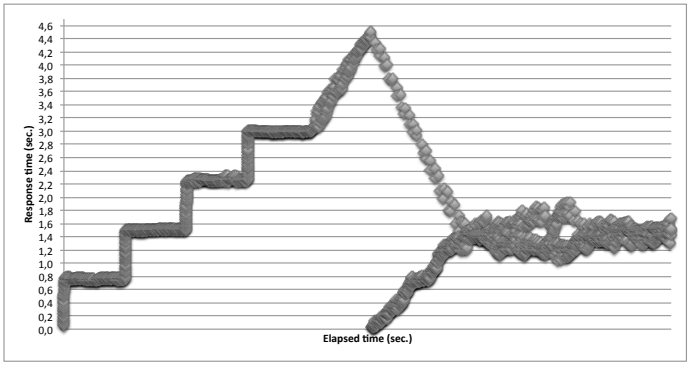

Figura 58.- Experiment 2: Readjustment of infrastructure resources at the macro level, adaptation 2 (Method: *GetSize*).

Subsequent to these experiments another set of experiments were also conducted for the distribution of infrastructure resources at the macro level. These experiments were looking for various consecutive executions in the adaptation process so that a single execution of the adaptation algorithm could not satisfy the demand in the services. These tests, from the point of view of the adaptation model, were positive, functioning perfectly within the limits of the case study. However, as presented in the graph in Figura 59, after 2 or 3 readaptation processes, depending on the method evaluated, the QoS did not lower as with the first readaptations, and the response time was greatly dispersed. This dispersion was not considered to be due to the proposed resource distribution algorithms, but to the overload of the demand for resources in the lower levels that contain persistence data (data base and distributed storage system).

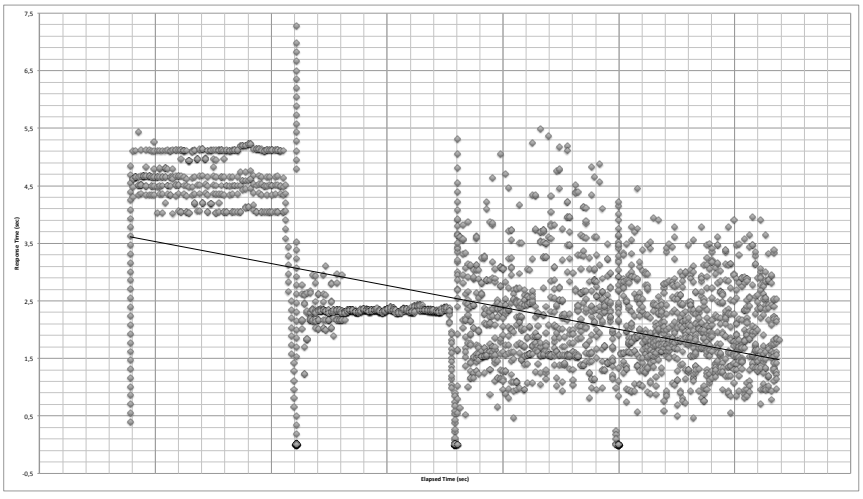

Figura 59.- Experiment 3: Readjustment of infrastructures at a macro level, consecutive adaptations (Method: *GetSize*)

7.5 DISCUSSION

The previous section provided an empirical evaluation in a case study of a multiagent architecture and the adaptation models proposed within the framework of this study. The experiments conducted have made it possible to validate various aspects as explained in detail below.

The proposed dynamic adaptation model is framed within those proposals that seek to increase energy efficiency. These models are still in a state of development and immaturity given that larger platforms use another approach for the distribution of computational resources. The predominant approach is guided by economic interests [Buyya *et al.*, 2008] [You *et al.*, 2009], to the detriment of the interests in energy and efficiency. However, in the framework of this research, the studies analyzed indicate that efforts are focusing on the search for efficient solutions to the distribution of computational resources in CC environments; that is, in line with the model proposed in this research project.

An empirical comparison of the proposed model with other existing approaches in the state of the art is not possible, since it is difficult to recreate the computational environments and/or simulation in which they were evaluated. However, it is possible to perform a theoretical comparison of the approach proposed with respect to other studies in the current state of the art. First, the proposed model follows a distributed approach to solve the problem, which is completely different from other studies in the state of the art [Raghavendra *et al.*, 2008] [Beloglazov *et al.*, 2012]. This approach, which has been demonstrated to be valid for the distribution of computational resources in this type of CC environment, presents certain advantages with respect to availability since there is not just one component in charge of the distribution of resources; instead, it is the system (society) itself that reorganizes through the individual adaptation of its components (agents).

Furthermore, in the approaches in the current state of the art, the execution of the assignment algorithms is a complex task that requires a great deal of computational time and power [Goudarzi *et* Pedram, 2011]. In contrast, the proposed model simplifies the search for an appropriate solution to the problem because (i) it distributes the computational needs among different nodes; (ii) there are fewer values to consider since each node need only consider the data for its own resources; and finally (iii) each node can autonomously apply a partial solution to the problem, thus eliminating the need to coordinate at he global level of the platform. In terms of the specific adaptation algorithms, the proposed solution uses optimization techniques, which have been previously used [Kusic *et al.*, 2009], but never following a distributed approach.

The other big difference in this study with regard to other approaches in the current state of the art is the minimum unit of distribution. While the minimum unit in the state of the art is usually the virtual machine [Buyya *et al.*, 2010b] [Beloglazov *et al.*, 2012], this study considers the minimum units to be the number of *vcpus* and the virtual memory assigned to each virtual machine in the infrastructure. With this approach it is possible to distinguish the micro and the macro level in the distribution of infrastructure resources, which makes it possible to solve the problem of demand without needing to instantiate virtual machines, which is in itself an energy efficient solution and maximizes the use of computational resources.

Finally, the clearest advantage of the proposed model with respect to other studies is its learning capability. The infrastructure distribution model at the macro level used an adaptation algorithm based on a case-based reasoning system that integrates a specialized agent from the +Cloud organization. With this focus, it is possible, as we have demonstrated, for the system to learn from past experiences, thus achieving greater efficiency in the adaptations as it learns and memorizes both positive and negative experiences. In the current state of the art there is no similar approach in which the distribution of computational resources is performed according to the results obtained from prior adaptation processes.

7.6 CONCLUSIONS AND FUTURE WORK

This study initially set forth to be one of the first MAS approaches, or more specifically a VO-based MAS approach, to fall within the framework of control and monitoring systems in a CC environment. The study proposed a new architectural model based on a MAS VO with a clearly integrative character. A series of algorithms for the distribution of computational resources in a CC environment were developed, evaluated and validated within the architecture. Its biggest innovation centers on the system's dynamic ability to automatically adapt according to demand.

The proposed architecture model is appropriate for the problem we need to solve. This new model has demonstrated that a control and monitoring system in a CC environment can be designed with artificial societies. This approach ensures independence of the decision-making process in software layers where the various actions are executed. This characteristics is particularly important in a CC environment because, as shown in the first phases of this research, current platforms exhibit a high dependency on the technological environment (virtualization tools, load balancers, distributed file systems, etc.). In the case of the +Cloud architecture, which uses ports to communicate with the environment, this dependency is limited to implementing the port itself; in other words, the communication interface with the environment. There is no doubt that a change in the capabilities offered by the underlying technology will also require changes to be made in the proposed reasoning models, as with any approach with a traditional design. However, in these cases the organizational models also offer an appropriate response to this challenge. Given the definitions of roles at a high level, if the technology proposes new capabilities, the adaptation in the proposed architecture will consist of modifying the individual or individuals that perform specific tasks or have a role in the organization.

This model to construct software systems allows external agents to access the model to provide certain functionality in Exchange for taking on and complying with a set of norms that ensure the stability and survival of the society. The framework of the proposed study includes a system designed so that the agents are or can be external. Within this study, human agents that participate in the organization (*Cloud User, End user and Cloud Admin*) are, obviously, external; however, the *SLA broker* agent was also designed to be external. The marketing context of a CC system can change in the future, while the +Cloud system can be adapted to the new models from their inception. The design of the SLA broker role as an external agent allows agents to access the organization to provide negotiation and procurement of service functions, regardless of the marketing model on which the agents are based. The design model contributes to the interoperability among platforms due to the fact that

the system is constructed to allow for an open connection among platforms (Intercloud [Sim, 2013]). Additionally, the MAS that follow the design process according to organizational models and systems, such as the case with the proposed architectural model, have distinct advantages with regards to application of the CC systems, since they allow the design to be modified by introducing new roles within the society, which can be done by simply defining their responsibilities, objectives and norms.

In conclusion, in light of the results (empirical and theoretical) presented, the CC platform that integrates the proposed architecture, referred to as +Cloud, validates the hypothesis of this research project: *"it is possible to model the control and monitoring system of a CC platform by using a VO of intelligent agents that self-adapt and reorganize according to the needs of the surrounding environment. Individual agent will autonomously manage the resources and services available and coordinate with the other agents to provide a dynamic and distributed adaptation model according to the needs of the end user."*

The research study that was developed, and which included a detailed memory, has made it possible to contribute to the state of the art in the field of agent theory, virtual organizations and distributed intelligent systems. The main contributions of this research are presented below:

- We have studied the need for a CC environment.
- We have analyzed the relationship between the theory of agents and the CC paradigm.
- We have analyzed the organizational models of intelligent agents to design complex software systems.
- We have performed a broad study of complementary technologies.
- We have characterized the computing environment.
- We have designed a self-organizing multiagent architecture well suited for the problem to solve.
- We have designed a dynamic adaptation model using a case-based reasoning system and optimization algorithms.
- We have validated the proposed architectural model in a real application environment.

In addition to the above mentioned contributions to the state of the art, this doctoral thesis has provided the foundation for the development of an important fundamental non-oriented research project at the national level known as **iHAS: Intelligent Social Computing for Human-Agent Societies** (TIN2012-36586-C03-0), supported by the Ministry of Science and Innovation and FEDER funds. Within this project, the proposed architectural model for this research project is in line with the first demonstration of the project, which aims to develop a social computational model within the framework of

CC environments. Thus, the architectural model developed in this study satisfies the objectives indicated in the development of the previously mentioned demonstration for the following reasons: (i) we have studied the needs of a CC environment; (ii) we have designed a CC system foundation that provides a VO of intelligent agents; with this design, (iii) we have implemented a functional prototype that was iv) evaluated in greater detail within a real exploitation environment. Based on the premise that we have, as previously mentioned, already reached our proposed objectives, not just within the framework of this doctoral thesis, but also within the project associated with this research project, we can propose the following lines of work that will be undertaken over a short and long term basis as a complement to the initially established objectives:

- Define and build an SLA negotiation model that integrates different advanced reasoning models derived from AI to guide the negotiation process, which will make it possible to integrate the automated procurement service for the products offered into the developed platform.
- Define a service interoperability model according to their different types and environment (software, platform and infrastructure), which will make it possible to define a protocole to facilitate its creation regardless of the service provider.
- Define a pricing model that can, in relation to demand, the production model and energy costs, determine a pricing model that can establish variable prices according to the specific state of the CC system.
- The combination of the three previously mentioned lines of work will make it possible to support the interoperability among platforms, at both a horizontal and vertical level, of the products offered on a CC platform.
- Extend the distribution algorithms for computational resources using two main objectives. First, we aim to adapt the proposed dynamic self-adapting model to then include all of the software layers of a CC system, including the persistence layer. Additionally, we also aim to extend the proposed adaptation model to include other infrastructure products, especially those that allow high-performance computing centered on the massive analysis of data.

Finally, the last line of work aims to investigate the development of new user interaction models with the CC system, using the latest available devices and models. This will make it possible to transform the platforms from a CC paradigm into social machines that adapt to the needs of users not just in terms of performance of services, but also in terms of interactions with users.

BIBLIOGRAFÍA Y REFERENCIAS

[Aamodt *et* Plaza, 1994] Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications, 7(1), 39-59.

[Abraham, 2003] Abraham, A. (2003). Business intelligence from web usage mining. Journal of Information & Knowledge Management, 2(04), 375-390.

[Agrawal *et al.*, 2001] Agrawal, D., Das, S., & El Abbadi, A. (2011, March). Big data and cloud computing: current state and future opportunities. In Proceedings of the 14th International Conference on Extending Database Technology (pp. 530-533). ACM.

[Agüero *et al.*, 2009] Agüero, J., Rebollo, M., Carrascosa, C., & Julián, V. (2009, January). Agent design using model driven development. In 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009) (pp. 60-69). Springer Berlin Heidelberg.

[Agüero *et al.*, 2010] Agüero, J., Rebollo, M., Carrascosa, C., & Julián, V. (2010). MDD for Virtual Organization design. In Trends in Practical Applications of Agents and Multiagent Systems (pp. 9-17). Springer Berlin Heidelberg.

[Aikins *et al.*, 2007] Aikins, H. A., Darling, C. L., Gernaey, M. E., & Kaldestad, H. S. (2007). U.S. Patent No. 7,296,268. Washington, DC: U.S. Patent and Trademark Office

[Al Feel *et* Khafagy, 2011] al Feel, H. T., & Khafagy, M. H. (2011, November). OCSS: Ontology Cloud Storage System. In Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on (pp. 9-13). IEEE.

[Alberola *et al.*, 2011] Alberola, J. M., Julian, V., & Garcia-Fornes, A. (2011). Open issues in multiagent system reorganization. In Highlights in Practical Applications of Agents and Multiagent Systems (pp. 151-158). Springer Berlin Heidelberg.

[Alhamad *et al.*, 2010a] Alhamad, M., Dillon, T., & Chang, E. (2010, April). Conceptual SLA framework for cloud computing. In Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on (pp. 606-610). IEEE.

[Alhamad *et al.*, 2010b] Alhamad, M., Dillon, T., & Chang, E. (2010, September). Sla-based trust model for cloud computing. In Network-Based Information Systems (NBiS), 2010 13th International Conference on (pp. 321-324). IEEE.

[Amato *et al.*, 2013] Amato, A., Di Martino, B., & Venticinque, S. (2013). Agent-Based Decision Support for Smart Market Using Big Data. In Algorithms and Architectures for Parallel Processing (pp. 251-258). Springer International Publishing.

[An *et al.*, 2010] An, B., Lesser, V., Irwin, D., & Zink, M. (2010, May). Automated negotiation with decommitment for dynamic resource allocation in cloud computing. InProceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1 (pp. 981-988). International Foundation for Autonomous Agents and Multiagent Systems.

[Anderson *et al.*, 2005] Anderson, T., Peterson, L., Shenker, S., & Turner, J. (2005). Overcoming the Internet impasse through virtualization. Computer, 38(4), 34-41.

[Andrzejak *et al.*, 2010] Andrzejak, A., Kondo, D., & Yi, S. (2010, August). Decision model for cloud computing under sla constraints. In Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on (pp. 257-266). IEEE.

[Annunziato *et* Pierucci, 2003] Annunziato, M., & Pierucci, P. (2003). The emergence of social learning in artificial societies. In Applications of evolutionary computing (pp. 467-478). Springer Berlin Heidelberg.

[Arcos *et al.*, 2005] Arcos, J. L., Esteva, M., Noriega, P., Rodríguez-Aguilar, J. A., & Sierra, C. (2005). Engineering open environments with electronic institutions. Engineering applications of artificial intelligence, 18(2), 191-204.

[Argente *et al.*, 2011] Argente, E., Botti, V., & Julian, V. (2011). GORMAS: An organizational-oriented methodological guideline for open MAS. In Agent-Oriented Software Engineering X (pp. 32-47). Springer Berlin Heidelberg.

[Argente, 2008] Argente Villaplana, E. (2008). Gormas: Guías para el desarrollo de sistemas multiagente abiertos basados en organizaciones.

[Armbrust *et al.*, 2010] Armbrust, Michael, Fox, Armando, Griffith, Rean, Joseph, Anthony D., Katz, Randy, Konwinski, Andy, Lee, Gunho, Patterson, David, Ariel, Rabkin, Stoica, Ion, Matei Zaharia. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

[Artikis *et al.*, 2001] Artikis, A., Kamara, L., Pitt, J.: Towards an Open Agent Society Model and Animation, Proc. of the 2nd. Agent-Based Simulation Workshop, Passau, 2001, pp. 48 – 55.

[Artikis *et* Pitt, 2001] Artikis, A., Kamara, L., & Pitt, J. (2001). Towards an open agent society model and animation. In Proceedings of the Agent-Based Simulation II workshop, Passau (pp. 48-55).

[Aversa *et al.*, 2010] Aversa, R., Di Martino, B., Rak, M., & Venticinque, S. (2010, February). Cloud agency: A mobile agent based cloud system. In Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on (pp. 132-137). IEEE.

[Azeez *et al.*, 2010] Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., Weerawarana, S., Fremantle, P. (2010, July). Multi-tenant SOA middleware for cloud computing. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 458-465). IEEE.

[Azodolmolky *et al.*, 2013] Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). Cloud computing networking: challenges and opportunities for innovations. Communications Magazine, IEEE, 51(7).

[Babu *et al.*, 2014] Babu, S. R., Kulkarni, K. G., & Sekaran, K. C. (2014, January). A Generic Agent Based Cloud Computing Architecture for E-Learning. In ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I (pp. 523-533). Springer International Publishing.

[Badget *et al.*, 2012] Badger, L., Grance, T., Patt-Corner, R., & Voas, J. (2012). Cloud computing synopsis and recommendations. NIST special publication, 800, 146.

[Bajo *et al.*, 2008] Bajo, J., Julián, V., Corchado, J. M., Carrascosa, C., de Paz, Y., Botti, V., & de Paz, J. F. (2008). An execution time planner for the ARTIS agent architecture. Engineering Applications of Artificial Intelligence, 21(5), 769-784.

[Bajo *et al.*, 2010] Bajo, J., Zato, C., de la Prieta, F., de Luis, A., & Tapia, D. (2010). Cloud Computing in Bioinformatics. In Distributed Computing and Artificial Intelligence(pp. 147-155). Springer Berlin Heidelberg.

[Baker *et al.*, 2011] Baker, W., Hutton, A., Hylender, C. D., Pamula, J., Porter, C., & Spitler, M. (2011). 2011 data breach investigations report. Verizon RISK Team, Available: www. verizonbusiness. com/resources/reports/rp_databreach-investigations-report-2011_en_xg. pdf, 1-72.

[Baktashmotlagh *et al.*, 2011] Baktashmotlagh, M., Bigdeli, A., & Lovell, B. C. (2011, August). Dynamic resource aware sensor networks: Integration of sensor cloud and ERPs. InAdvanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on (pp. 455-460). IEEE.

[Barham *et al.*, 2003] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer R., Pratt i. & Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.

[Barroso *et al.*, 2003] Barroso, L. A., Dean, J., & Holzle, U. (2003). Web search for a planet: The Google cluster architecture. Micro, Ieee, 23(2), 22-28.

[Barroso *et* Hözle, 2009] Barroso, L. A., & Hölzle, U. (2009). The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis lectures on computer architecture, 4(1), 1-108.

[Barroso, 2005] Barroso, L. A. (2005). The price of performance. Queue, 3(7), 48-53.

[Bauer *et al.*, 2001] Bauer, B., Müller, J. P., & Odell, J. (2001). Agent UML: A formalism for specifying multiagent software systems. International journal of software engineering and knowledge engineering, 11(03), 207-230.

[Bedia, 2004] Bedia, M. A. G. (2004). Fundamentos cognitivos para el diseño de arquitecturas de agentes planificadores en contextos dinámicos de acción(Doctoral dissertation, Universidad de Salamanca).

[Bellard, 2005] Bellard, F. (2005, April). QEMU, a Fast and Portable Dynamic Translator. InUSENIX Annual Technical Conference, FREENIX Track (pp. 41-46).

[Beloglazov *et al.*, 2012] Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 28(5), 755-768.

[Beloglazov *et* Buyya, 2010] Beloglazov, A., & Buyya, R. (2010, November). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science (p. 4). ACM.

[Beloussov *et al.*, 2008] Beloussov, S. M., Protassov, S. S., & Tormasov, A. G. (2008). U.S. Patent No. 7,461,148. Washington, DC: U.S. Patent and Trademark Office.

[Bendoukha *et* Wagner, 2012] Bendoukha, S., & Wagner, T. Cloud Transition: Integrating Cloud Calls into Workflow Petri Nets.

[Benmerzoug, 2013] Benmerzoug, D. (2013). An Agent-Based Approach For Hybrid Multi-Cloud Applications. Scalable Computing: Practice and Experience, 14(2).

[Berners-Lee, 1996] Berners-Lee, T. (1996). WWW: Past, present, and future. Computer, 29(10), 69-77.

[Bernon *et al.*, 2002] Bernon, C., Gleizes, M. P., Picard, G., & Glize, P. (2002). The Adelfe Methodology For an Intranet System Design. AOIS@ CAiSE, 57.

[Bernon *et al.*, 2003] Bernon, C., Gleizes, M. P., Peyruqueou, S., & Picard, G. (2003). Adelfe: A methodology for adaptive multi-agent systems engineering. In Engineering Societies in the Agents World III (pp. 156-169). Springer Berlin Heidelberg.

[Bezemer, 2010] Bezemer, C. P., Zaidman, A., Platzbeecker, B., Hurkmans, T., & t Hart, A. (2010, September). Enabling multi-tenancy: An industrial experience report. InSoftware Maintenance (ICSM), 2010 IEEE International Conference on (pp. 1-8). IEEE.

[Bhadauria *et* Sanyal, 2012] Bhadauria, R., & Sanyal, S. (2012). Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques. arXiv preprint arXiv:1204.0764.

[Bianchini *et* Rajamony, 2004] Bianchini, R., & Rajamony, R. (2004). Power and energy management for server systems. IEEE Computer, 37(11), 68-74.

[Bing *et al.*, 2009] Bing, G., Yan, S., & Zi-Li, S. (2009). The redefinition and some discussion of green computing. Chinese Journal of Computers, 32(12), 2311-2319.

[Blasch *et al.*, 2014] Blasch, E., Chen, Y., Chen, G., Shen, D., & Kohler, R. (2014). Information Fusion in a Cloud-Enabled Environment. In High Performance Cloud Auditing and Applications (pp. 91-115). Springer New York.

[Boehm, 1984] Boehm, B. W. (1981). Software engineering economics, IEEE Transactions on, 1984, no 1, p. 4-21.

[Bond *et* Gasser, 1988] Alan H. Bond, Leslie George Gasser. (1988) Readings in Distributed Artificial Intelligence. Morgan Kaufmann.

[Bonjean *et al.*, 2012] Bonjean, N., Gleizes, M. P., Chella, A., Migeon, F., Cossentino, M., & Seidita, V. (2012, June). Metamodel-based metrics for agent-oriented methodologies. InProceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2 (pp. 1065-1072). International Foundation for Autonomous Agents and Multiagent Systems.

[Booch *et al.*, 1998] Booch, G., Rumbaugh, J., & Jacobson, I. (1998). The Unified Modeling Language (UML). World Wide Web: http://www. rational. com/uml/(UML Resource Center), 94.

[Bote-Lorenzo *et al.*, 2004] M.L. Bote-Lorenzo, Y.A. Dimitriadis, E. Gómez-Sánchez. Grid Characteristics and Uses: A Grid Definition. Postproceedings of the First European Across Grids Conference (ACG'03), Springer Verlang. LNCS 2970, Spain, Feb. 2004. Pages. 291-298.

[Botti *et* Giret, 2008] Botti, V., & Giret, A. (2008). ANEMONA: A multi-agent methodology for Holonic Manufacturing Systems. Springer.

[Bou *et al.*, 2007] Bou, E., López-Sánchez, M., & Rodríguez-Aguilar, J. A. (2007). Towards self-configuration in autonomic electronic institutions. In Coordination, Organizations, Institutions, and Norms in Agent Systems II (pp. 229-244). Springer Berlin Heidelberg.

[Bousquet *et al.*, 2002] Bousquet, F., Barreteau, O., d'Aquino, P., Etienne, M., Boissau, S., Aubert, S., Le Page C, Babin D. & Castella, J. C. (2002). Multi-agent systems and role games: collective learning processes for ecosystem management. Complexity and Ecosystem Management. The Theory and Practice of Multi-Agent Systems, Edward Elgar, Londres, 248-286.

[Bragg, 2008] Bragg, R. (2008). Cloud computing: When computers really rule. Tech News World, 12(12), 2009.

[Bratman *et al.*, 1988] Bratman M.E., Israel D. and Pollack M.E. (1988). Plans and resource-bounded practical reasoning. Computational Intelligence, 4. pages 349-355.

[Bratman, 1987] Bratman M.E. (1987). Intentions, Plans and Practical Reason. Harvard University Press, Cambridge, M.A.

[Brenner *et al.*, 2012] Brenner, W., Zarnekow, R., & Wittig, H. (2012). Intelligent software agents: foundations and applications. Springer Publishing Company, Incorporated..

[Bresciani *et al.*, 2004] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 8(3), 203-236.

[Brian *et al.*, 2008] Brian, H. A. Y. E. S., Brunschwiler, T., Dill, H., Christ, H., Falsafi, B., Fischer, M., & Zollinger, M. (2008). Cloud computing. Communications of the ACM, 51(7), 9-11.

[Broberg *et al.*, 2008] Broberg, J., Venugopal, S., & Buyya, R. (2008). Market-oriented grids and utility computing: The state-of-the-art and future directions. Journal of Grid Computing, 6(3), 255-276.

[Brunet *et al.*, 20112] Brunet, P. M., Montmorry, A., & Frezouls, B. (2012). Big data challenges, an insight into the Gaia Hadoop solution.

[Bunch *et al.*, 2010] Bunch, C., Chohan, N., Krintz, C., Chohan, J., Kupferman, J., Lakhina, P., ... & Nomura, Y. (2010, July). An evaluation of distributed datastores using the AppScale cloud platform. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 305-312). IEEE.

[Buyya *et al.*, 2008] Buyya, R., Yeo, C. S., & Venugopal, S. (2008, September). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on (pp. 5-13). Ieee.

[Buyya *et al.*, 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

[Buyya *et al.*, 2010a] Buyya, R., Ranjan, R., & Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In Algorithms and architectures for parallel processing (pp. 13-31). Springer Berlin Heidelberg.

[Buyya *et al.*, 2010b] Buyya, R., Beloglazov, A., & Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: A 251 isión, architectural elements, and open challenges. arXiv preprint arXiv:1006.0308.

[Buyya *et al.*, 2011] Buyya, R., Garg, S. K., & Calheiros, R. N. (2011, December). SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In Cloud and Service Computing (CSC), 2011 International Conference on (pp. 1-10). IEEE.

[Caire *et al.*, 2002] Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavón, J., Leal, F., Chainho P., Kearney P., Stark J., Evans R. & Massonet, P. (2002). Agent oriented analysis using MESSAGE/UML. In Agent-oriented software engineering II (pp. 119-135). Springer Berlin Heidelberg.

[Calheiros *et al.*, 2011] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23-50.

[Calheiros *et al.*, 2012] Calheiros, R. N., Toosi, A. N., Vecchiola, C., & Buyya, R. (2012). A coordinator for scaling elastic applications across multiple clouds. Future Generation Computer Systems, 28(8), 1350-1362.

[Camargo *et al.*, 2008] Camargos, F., Girard, G., & Ligneris, B. (2008). Virtualization of Linux servers.Proc. 2008 Linux Syposium, Ottawa, Ontario, Canada, 63-76.

[Capera *et al.*, 2003] Capera, D., Georgé, J.-P., Gleizes, M.-P., & Glize, P. (2003). Emergence of organisations, emergence of functions. Symposium on Adaptive Agents and Multi-Agent Systems, (págs. 103-108).

[Cardellini *et al.*, 1999] Cardellini, V., Colajanni, M., & Philip, S. Y. (1999). Dynamic load balancing on web-server systems. IEEE Internet computing, 3(3), 28-39.

[Caron *et al.*, 2009] Caron, E., Desprez, F., Loureiro, D., & Muresan, A. (2009, September). Cloud computing resource management through a grid middleware: A case study with DIET and eucalyptus. In Cloud Computing, 2009. CLOUD'09. IEEE International Conference on (pp. 151-154). IEEE.

[Carr, 2005] Carr, N. G. (2005). The end of corporate computing. MIT Sloan Management Review, 46(3).http://sloanreview.mit.edu/article/the-end-of-corporate-computing/

[Carrascosa *et al.*, 2008] Carrascosa, C., Bajo, J., Julián, V., Corchado, J. M., & Botti, V. (2008). Hybrid multi-agent architecture as a real-time problem-solving model. Expert Systems with Applications, 34(1), 2-17.

[Carrascosa *et al.*, 2009] Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., & Botti, V. (2009, May). Service oriented MAS: an open architecture. In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2 (pp. 1291-1292). International Foundation for Autonomous Agents and Multiagent Systems.

[Carvalho *et al.*, 2006] Carvalho, G., Almeida, H., Gatti, M., Vinicius, G., Paes, R., Perkusich, A., & Lucena, C. (2006, October). Dynamic law evolution in governance mechanisms for open multi-agent systems. In Second workshop on software engineering for agent-oriented systems.

[Castelfranchi, 2000] Castelfranchi, C. (2000, January). Engineering social order. In Engineering societies in the agents world (pp. 1-18). Springer Berlin Heidelberg.

[Celesti *et al.*, 2010a] Celesti, A., Tusa, F., Villari, M., & Puliafito, A. (2010, July). How to enhance cloud architectures to enable cross-federation. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 337-345). IEEE.

[Celesti *et al.*, 2010b] Celesti, A., Tusa, F., Villari, M., & Puliafito, A. (2010, July). Three-phase cross-cloud federation model: The cloud sso

authentication. In Advances in Future Internet (AFIN), 2010 Second International Conference on (pp. 94-101). IEEE.

[Cernuzzi *et al.*, 2004] Cernuzzi, Luca, Juan, Thomas, Sterling, Leon, Zambonelli, Franco (2004). The Gaia methodology: basic concepts and extensions, Kluwer Academic Publishers

[Cernuzzi *et* Zambonelli, 2006] Cernuzzi, L., & Zambonelli, F. (2006). Dealing with adaptive multi-agent organizations in the gaia methodology. In Agent-Oriented Software Engineering VI (pp. 109-123). Springer Berlin Heidelberg.

[Cetkovic *et* Parmee , 2002] Cvetkovic, D., & Parmee, I. (2002). Agent-based support within an interactive evolutionary design system. In Adaptive Computing in Design and Manufacture V (pp. 355-367). Springer London.

[Chahal *et al.*, 2010] Chahal, S., Hahn-Steichen, J., Kamhout, D., Kraemer, R., Li, H., and Peters, C. (2010, June). And Enterprise Private Cloud Arquitecture and Implemetation Roadmap. IT@Intel Whipe Paper. Business Solutions – Interl Information Technology.

[Chamorro *et* De Solis Montes, 2013] Chamorro, J. M. C., & de Solís Montes, D. R. (2013). An Agent-Based Approach for Data Fusion in Homeland Security.

[Chang *et al.*, 2008] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., ... & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2), 4.

[Chao *et* Sun, 2013] Chao, P., & Sun, H. M. (2013). Multi-agent-Based Cloud Utilization for the IT Office-Aid Asset Distribution Chain: An Empirical Case Study. Information Sciences.

[Che *et al.*, 2008] Che, J., He, Q., Gao, Q., & Huang, D. (2008, December). Performance measuring and comparing of virtual machine monitors. In Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on (Vol. 2, pp. 381-386). IEEE.

[Che *et al.*, 2010] Che, J., Yu, Y., Shi, C., & Lin, W. (2010, December). A synthetical performance evaluation of OpenVZ, Xen and KVM. In Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific (pp. 587-594). IEEE..

[Chella *et al.*, 2006] Chella, A., Cossentino, M., Sabatucci, L., & Seidita, V. (2006). Agile PASSI: An agile process for designing agents. International Journal of Computer Systems Science & Engineering, 21(2), 133-144.

[Chellappa, 1997] Chellappa, R. (1997, October). Intermediaries in cloud-computing: A new computing paradigm. In INFORMS Annual Meeting, Dallas.

[Chen *et al.*, 2008] Chen, W., Lu, H., Shen, L., Wang, Z., Xiao, N., & Chen, D. (2008, November). A novel hardware assisted full virtualization technique. In Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for (pp. 1292-1297). IEEE.

[Chen *et al.*, 2010] Chen, G., Lu, J., Huang, J., & Wu, Z. (2010, August). Saaas-the mobile agent based service for cloud computing in internet environment. In Natural Computation (ICNC), 2010 Sixth International Conference on (Vol. 6, pp. 2935-2939). IEEE.

[Chen *et al.*, 2013] Chen, D., Wang, L., Wu, X., Chen, J., Khan, S. U., Kołodziej, J., Tiana M., Huang F. & Liu, W. (2013). Hybrid modelling and simulation of huge crowd over a hierarchical Grid architecture. Future Generation Computer Systems, 29(5), 1309-1317.

[Chiu, 2010] Chiu, D. (2010). Elasticity in the cloud. ACM Crossroads, 16(3), 3-4.

[Chowdhury *et* Boutaba, 2009] Chowdhury, N. M. K., & Boutaba, R. (2009). Network virtualization: state of the art and research challenges. Communications Magazine, IEEE, 47(7), 20-26.

[Ciancarini *et al.*, 2000] Ciancarini, P., Omicini, A., & Zambonelli, F. (2000). Multiagent system engineering: the coordination viewpoint. In Intelligent Agents VI. Agent Theories, Architectures, and Languages (pp. 250-259). Springer Berlin Heidelberg.

[Cimatti *et* Roveri, 2000] Cimatti, A., & Roveri, M. (2000). Conformant planning via symbolic model checking. J. Artif. Intell. Res.(JAIR), 13, 305-338.

[Cisco, 2009] Cisco (2009), Cisco Cloud Computing - Data Center Strategy, Architecture and Solutions, White Paper for U.S Public Sector, 1st edition

[Claus *et* Boutilier, 1998] Claus, C., & Boutilier, C. (1998, July). The dynamics of reinforcement learning in cooperative multiagent systems. In AAAI/IAAI (pp. 746-752).

[Clifford *et al.*, 2007] Clifford, M., Miles, N., & Rabe, B. R. (2007). U.S. Patent No. 7,194,538. Washington, DC: U.S. Patent and Trademark Office.

[Cohen, 2013] Cohen, B. (2013). PaaS: New Opportunities for Cloud Application Development.Computer, 46(9), 0097-100.

[Collier *et* North, 2011] Collier, N., & North, M. (2011). Repast HPC: A platform for large-scale agentbased modeling (pp. 81-110). Wiley.

[Corchado *et al.*, 2008] Corchado, J. M., Glez-Bedia, M., De Paz, Y., Bajo, J., & De Paz, J. F. (2008). Replanning mechanism for deliberative agents in dynamic changing environments. Computational Intelligence, 24(2), 77-107.

[Corchado *et* Laza, 2003] Corchado, J. M., & Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. International Journal of Intelligent Systems,18(12), 1227-1241.

[Cossentino, 2005] Cossentino, M. (2005). From requirements to code with the PASSI methodology. Agent-oriented methodologies, 4, 79-106.

[Cristelli *et al.*, 2014] Cristelli, M., Tacchella, A., & Pietronero, L. (2014). An Overview of the New Frontiers of Economic Complexity. In Econophysics of Agent-Based Models(pp. 147-159). Springer International Publishing.

[Crosby *et* Brown, 2006] Crosby, S., & Brown, D. (2006). The virtualization reality. Queue, 4(10), 34-41.

[CSA, 2011] CSA (2011). Security guidance for critical areas of focus in cloud computing v3. 0. Cloud Security Alliance.

[Cusumano, 2010] Cusumano, M. (2010). Cloud computing and SaaS as new computing platforms. Communications of the ACM, 53(4), 27-29.

[Cuzzocrea *et al.*, 2013] Cuzzocrea, A., Fortino, G., & Rana, O. (2013, May). Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-Of-The-Art and Future Research Directions. In Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on (pp. 583-588). IEEE.

[D'Inverno *et* Luck, 2004] D'Inverno M. y Luck M. (2004) Understanding Agent Systems. Springer Verlag. isbn 3540407006.

[Daft, 2008] Daft Richard, L. (1998). Organization theory and design. South Western College Publishing (Cincinnati, Ohio), ISBN, 538879025.

[Damiani *et* Pagano, 2010] Damiani, E., & Pagano, F. (2010). Handling confidential data on the untrusted cloud: an agent-based approach. arXiv preprint arXiv:1012.0759.

[Dasgupta *et* Serageldin, 2000] Dasgupta, P., & Serageldin, I. (Eds.). (2000). Social capital: a multifaceted perspective. World Bank Publications.

[Dautenhahn, 2000] Dautenhahn, K. (2000). Reverse engineering of societies-a biological perspective. In Proceedings of the AISB Symposium, Starting from Society-the application of social analogies to computational systems.

[Davidsson *et* Johansson, 2006] Davidsson, P., & Johansson, S. (2006). On the potential of norm-governed behavior in different categories of artificial

societies. Computational & Mathematical Organization Theory, 12(2-3), 169-180.

[Davidsson, 2001] Davidsson, P. (2001). Categories of artificial societies. In Engineering Societies in the Agents World II (pp. 1-9). Springer Berlin Heidelberg.

[De Clercq, 2002] De Clercq, J. (2002). Single sign-on architectures. In Infrastructure Security(pp. 40-58). Springer Berlin Heidelberg.

[De la Prieta *et al.*, 2009] de la Prieta, F., Pérez-Lancho, B., De Paz, J. F., Bajo, J., & Corchado, J. M. (2009, July). Ovamah: Multiagent-based adaptive virtual organizations. In Information Fusion, 2009. FUSION'09. 12th International Conference on (pp. 990-997). IEEE.

[De la Prieta *et al.*, 2014] De la Prieta, F., Gil, A. B., Rodríguez, S., Pérez, J. B., Coria, J. A. G., & Corchado, J. M. (2014, January). An Enhanced Approach to Retrieve Learning Resources Over the Cloud. In The 2nd International Workshop on Learning Technology for Education in Cloud (pp. 193-203). Springer Netherlands.

[De Mántaras *et al.*, 2005] De Mántaras R., McSherry D., Bridge D., Leake D., Smyth B., Craw S., Faltings B., Maher M.L., Cox M.T., Forbus K., Keane M., Aamodt A. y Watson I. (2005). Retrieval, reuse, revision and retention in case-based reasoning. The Knowledge Engineering Review, 20(03), 215-240.

[De Mántaras *et* Plaza, 1997] De Mántaras, R. L., & Plaza, E. (1997). Case-based reasoning: an overview. AI communications, 10(1), 21-29.

[De Paz *et al.*, 2009] De Paz, J. F., Rodríguez, S., Bajo, J., & Corchado, J. M. (2009). Mathematical model for dynamic case-based planning. International Journal of Computer Mathematics, 86(10-11), 1719-1730.

[Dean *et* Ghemawat, 2008] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[Decraene *et al.*, 2010] Decraene, J., Cheng, Y. Y., Low, M. Y. H., Zhou, S., Cai, W., & Choo, C. S. (2010, August). Evolving agent-based simulations in the clouds. In Advanced Computational Intelligence (IWACI), 2010 Third International Workshop on (pp. 244-249). IEEE

[Decraene *et al.*, 2011] Decraene, J., Lee, Y. T., Zeng, F., Chandramohan, M., Cheng, Y. Y., & Low, M. Y. H. (2011, May). Evolutionary design of agent-based simulation experiments. In The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3 (pp. 1321-1322). International Foundation for Autonomous Agents and Multiagent Systems.

[DeLoach *et al.*, 2001] DeLoach, S. A., Wood, M. F., & Sparkman, C. H. (2001). Multiagent systems engineering. International Journal of Software Engineering and Knowledge Engineering, 11(03), 231-258.

[Deloach *et al.*, 2008] Deloach, S. A., Oyenan, W. H., & Matson, E. T. (2008). A capabilities-based model for adaptive organizations. Autonomous Agents and Multi-Agent Systems, 16(1), 13-56.

[DeLoach *et* Garcia-Ojeda, 2010] DeLoach, S. A., & Garcia-Ojeda, J. C. (2010). O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems.International Journal of Agent-Oriented Software Engineering, 4(3), 244-280.

[DeLoach *et* Matson, 2004] DeLoach, S. A., & Matson, E. (2004, July). An organizational model for designing adaptive multiagent systems. In The AAAI-04 Workshop on Agent Organizations: Theory and Practice (AOTP 2004) (pp. 66-73).

[DeLoach, 1999] DeLoach, S. A. (1999). Multiagent systems engineering: A methodology and language for designing agent systems. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.

[DeLoach, 2009] DeLoach, S. (2009). OMACS: A framework for adaptive, complex systems.Handbook of research on multi-agent systems: Semantics and dynamics of organizational models, 76-98.

[Desisto, 2013] Desisto, R.P. (2013, November). The Top Three Impacts of Cloud Computing on Sales and Business Applications. In: https://www.gartner.com/doc/2622822

[Dewan *et* Hansdah, 2011] Dewan, H., & Hansdah, R. C. (2011, July). A survey of cloud storage facilities. In Services (SERVICES), 2011 IEEE World Congress on (pp. 224-231). IEEE.

[Dignum *et al.*, 2002] Dignum, V., Meyer, J. J., Weigand, H., & Dignum, F. (2002, July). An organization-oriented model for agent societies. In Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications.

[Dignum *et al.*, 2004] Dignum, M. V., Sonenberg, E., & Dignum, F. P. M. (2004). Dynamic reorganization of agent societies.

[Dignum *et* Dignum, 2006] Dignum, V., & Dignum, F. (2006). A landscape of agent systems for the real world. Institute of Information and Computing Sciences, Utrecht University, Tech. Rep.

[Dignum, 2004] V. Dignum (2004) A model for organizational interaction: based on agents, founded in logic, PhD. Thesis,

[Dlamini, 2013] Dlamini, C. S., & Lombardi, T. (2013). Cloud Computing-The Vision in Mobile Perspective.

[Dossauge *et* Garrete, 1999] Dussauge, P., & Garrette, B. (1999). Cooperative strategy: Competing successfully through strategic

[Dustdar *et* Schereiner, 2005] Dustdar, S., & Schreiner, W. (2005). A survey on web services composition. International journal of web and grid services, 1(1), 1-30.

[Edmonds, 1999] Edmonds, B. (1999). Capturing Social Embeddedness: a constructivist approach. Adaptive Behavior, 7(3-4), 323-347.

[Eggleston, 2013] Eggleston, M. (2013, November). Using big data for computational modeling of infectious diseases. In 141st APHA Annual Meeting (November 2-November 6, 2013). APHA.

[Ejarque *et al.*, 2010] Ejarque, J., Sirvent, R., & Badia, R. M. (2010, November). A multi-agent approach for semantic resource allocation. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 335-342). IEEE.

[Ekanayake *et* Fox, 2010] Ekanayake, J., & Fox, G. (2010). High performance parallel computing with clouds and cloud technologies. In Cloud Computing (pp. 20-38). Springer Berlin Heidelberg.

[Elkington *et al.*, 2005] Elkington, S., Hess, R., Korgaonkar, A., Leveille, J., Lubbers, C., McCarty, J., & Sicola, S. J. (2005). U.S. Patent No. 6,880,052. Washington, DC: U.S. Patent and Trademark Office.

[Emeakaroha *et al.*, 2010] Emeakaroha, V. C., Brandic, I., Maurer, M., & Dustdar, S. (2010, June). Low level Metrics to High level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments. InHPCS (pp. 48-54).

[Emeakaroha *et al.*, 2012] Emeakaroha, V. C., Netto, M. A., Calheiros, R. N., Brandic, I., Buyya, R., & De Rose, C. A. (2012). Towards autonomic detection of sla violations in cloud infrastructures. Future Generation Computer Systems, 28(7), 1017-1029.

[Erdogmus, 2009] Erdogmus, H. (2009). Cloud computing: does Nirvana hide behind the nebula?. Software, IEEE, 26(2), 4-6.

[Erl, 2008] Erl, T. (2008). Soa: principles of service design (Vol. 1). Upper Saddle River: Prentice Hall.

[Esteva *et al.*, 2001] Esteva, M., Rodriguez-Aguilar, J. A., Sierra, C., Garcia, P., & Arcos, J. L. (2001). On the formal specification of electronic institutions.

In Agent mediated electronic commerce (pp. 126-147). Springer Berlin Heidelberg.

[Fan *et al.*, 2011] Fan, C. T., Wang, W. J., & Chang, Y. S. (2011, September). Agent-Based Service Migration Framework in Hybrid Cloud. In High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on (pp. 887-892). IEEE.

[Ferber *et al.*, 2004] Ferber, J., Gutknecht, O., & Michel, F. (2004). From agents to organizations: an organizational view of multi-agent systems. In Agent-Oriented Software Engineering IV (pp. 214-230). Springer Berlin Heidelberg.

[Ferber *et al.*, 2005] Ferber, J., Michel, F., & Báez, J. (2005). AGRE: Integrating environments with organizations. In Environments for multi-agent systems (pp. 48-56). Springer Berlin Heidelberg.

[Ferber *et* Gutknecht, 1998] Ferber, J., & Gutknecht, O. (1998, July). A meta-model for the analysis and design of organizations in multi-agent systems. In Multi Agent Systems, 1998. Proceedings. International Conference on (pp. 128-135). IEEE.

[Fernández-Caballero *et* Gascueña, 2010] Fernández-Caballero, A., & Gascueña, J. M. (2010). Developing multi-agent systems through integrating Prometheus, INGENIAS and ICARO-T. In Agents and Artificial Intelligence (pp. 219-232). Springer Berlin Heidelberg.

[Fisher *et al.*, 2010] Fisher, P., Pant, R., & Edberg, J. (2010). Cloud Computing: Assessing Azure, Amazon EC2, Google App Engine and Hadoop for IT Decision Making and Developer Career Growth. Apress.

[Fittkau *et al.*, 2012] Fittkau, F., Frey, S., & Hasselbring, W. (2012, September). CDOSim: Simulating cloud deployment options for software migration support. In Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the (pp. 37-46). IEEE.

[Foster *et al.*, 2008] Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008, November). Cloud computing and grid computing 360-degree compared. In Grid Computing Environments Workshop, 2008. GCE'08 (pp. 1-10). Ieee.

[Franklin *et* Graesser, 1997] Franklin, S., & Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Intelligent agents III agent theories, architectures, and languages (pp. 21-35). Springer Berlin Heidelberg.

[Fraternali *et al.*, 2010] Fraternali, P., Rossi, G., & Sánchez-Figueroa, F. (2010). Rich internet applications. Internet Computing, IEEE, 14(3), 9-12.

[Fraternalli *et al.*, 2010] Fraternali, P., Rossi, G., & Sánchez-Figueroa, F. (2010). Rich internet applications. Internet Computing, IEEE, 14(3), 9-12.

[Galbraith, 1977] Galbraith, J (1977). Organization Design . Addison-Wesley.

[Garcia *et al.*, 2012] García, E., Gallego, V., Rodríguez, S., Zato, C., & Bajo, J. (2012). Visualization of Agents and Their Interaction within Dynamic Environments. InManagement Intelligent Systems (pp. 15-24). Springer Berlin Heidelberg.

[Garcia-Camino *et al.*, 2005] García-Camino, A., Noriega, P., & Rodríguez-Aguilar, J. A. (2005, July). Implementing norms in electronic institutions. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems(pp. 667-673). ACM.

[García-Magariño *et al.*, 2011] García-Magariño, I., Gómez-Sanz, J. J., & Fuentes-Fernández, R. (2011). Model transformations for improving multi-agent system development in INGENIAS. In Agent-Oriented Software Engineering X (pp. 51-65). Springer Berlin Heidelberg.

[Garcia-Ojeda *et al.*, 2008] Garcia-Ojeda, J. C., DeLoach, S. A., Oyenan, W. H., & Valenzuela, J. (2008).O-MaSE: a customizable approach to developing multiagent development processes (pp. 1-15). Springer Berlin Heidelberg.

[Gascuena *et al.*, 2009] Gascuena, J. M., & Fernández-Caballero, A. (2009). Prometheus and INGENIAS agent methodologies: A complementary approach. In Agent-Oriented Software Engineering IX (pp. 131-144). Springer Berlin Heidelberg.

[Gasser, 1992] Gasser, L. (1992). An overview of DAI. Distributed Artificial Intelligence: Theory and Praxis, 9(9-29), 28.

[Georgeff *et* Lansky, 1987] Georgeff, M. P., & Lansky, A. L. (1987, July). Reactive reasoning and planning. In AAAI (Vol. 87, pp. 677-682).

[Gilbert, 1999] Gilbert, N. (1999). Simulation A New Way of Doing Social Science. American Behavioral Scientist, 42(10), 1485-1487.

[Giret *et al.*, 2005] Giret, A., Botti, V., & Valero, S. (2005). MAS methodology for HMS. In Holonic and Multi-Agent Systems for Manufacturing (pp. 39-49). Springer Berlin Heidelberg.

[Giret *et al.*, 2010] Giret, A., Julián, V., Rebollo, M., Argente, E., Carrascosa, C., & Botti, V. (2010). An open architecture for service-oriented virtual organizations. InProgramming Multi-Agent Systems (pp. 118-132). Springer Berlin Heidelberg.

[Giunchiglia *et al.*, 2003] Giunchiglia, F., Mylopoulos, J., & Perini, A. (2003). The tropos software development methodology: processes, models and

diagrams. In Agent-Oriented Software Engineering III (pp. 162-173). Springer Berlin Heidelberg.

[Goiri *et al.*, 2010] Goiri, Í., Julià, F., Fitó, J. O., Macías, M., & Guitart, J. (2010). Resource-level QoS metric for CPU-based guarantees in cloud providers. In Economics of Grids, Clouds, Systems, and Services (pp. 34-47). Springer Berlin Heidelberg.

[Golding *et* Rosenbloom, 1989] Golding, A. R., & Rosenbloom, P. S. (1989, May). Combining analytical and similarity-based CBR. In Proc. of the 2 nd Workshop on Case-Based Reasoning (pp. 259-263).

[Gomes-Casseres, 1996] Gomes-Casseres, B. (1996). The alliance revolution: The new shape of business rivalry. Harvard University Press

[Gomez-Sanz *et* Fuentes, 2002] Gomez-Sanz, J. O. R. G. E., & Fuentes, R. (2002). The INGENIAS Methodology. In Fourth Iberoamerican Workshop on Multi-Agent Systems Iberagents.

[Gong *et al.*, 2010] Gong, Z., Gu, X., & Wilkes, J. (2010, October). Press: Predictive elastic resource scaling for cloud systems. In Network and Service Management (CNSM), 2010 International Conference on (pp. 9-16). IEEE.

[Goth, 2007] Goth, G. (2007). Virtualization: Old technology offers huge new potential.Distributed Systems Online, IEEE, 8(2), 3-3.

[Goudarzi *et* Pedram, 2011] Goudarzi, H., & Pedram, M. (2011, July). Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems. In Cloud Computing (CLOUD), 2011 IEEE International Conference on (pp. 324-331). IEEE.

[Govinda *et* Sathiyamoorthy, 2012] Govinda, K., & Sathiyamoorthy, E. (2012). Agent Based Security for Cloud Computing using Obfuscation. Procedia Engineering, 38, 125-129.

[Gray *et* Sieweorek, 1991] Gray, J., & Siewiorek, D. P. (1991). High-availability computer systems. Computer, 24(9), 39-48.

[Grobauer *et al.*, 2011] Grobauer, B., Walloschek, T., & Stocker, E. (2011). Understanding cloud computing vulnerabilities. Security & Privacy, IEEE, 9(2), 50-57.

[Grossman *et al.*, 2009] Grossman, R. L., Gu, Y., Sabala, M., & Zhang, W. (2009). Compute and storage clouds using wide area high performance networks. Future Generation Computer Systems, 25(2), 179-183.

[Gutierrez-Garcia *et* Sim, 2010] Gutierrez-Garcia, J. O., & Sim, K. M. (2010, November). Self-organizing agents for service composition in cloud

computing. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 59-66). IEEE.

[Gutierrez-Garcia *et* Sim, 2012a] Gutierrez-Garcia, J. O., & Sim, K. M. (2012). A family of heuristics for agent-based elastic Cloud bag-of-tasks concurrent scheduling. Future Generation Computer Systems.

[Gutierrez-Garcia *et* Sim, 2012b] Gutierrez-Garcia, J. O., & Sim, K. M. (2012). Agent-based cloud workflow execution. Integrated Computer-Aided Engineering, 19(1), 39-56.

[Guzek *et al.*, 2013a] Guzek, M., Danoy, G., & Bouvry, P. (2013, May). ParaMoise: increasing capabilities of parallel execution and reorganization in an organizational model. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems (pp. 1029-1036). International Foundation for Autonomous Agents and Multiagent Systems.

[Guzek *et al.*, 2013b] Guzek, M., Danoy, G., & Bouvry, P. (2013, September). System design and implementation decisions for ParaMoise organisational model. In Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on(pp. 999-1005). IEEE.

[Habib *et al.*, 2011] Habib, S. M., Ries, S., & Muhlhauser, M. (2011, November). Towards a trust management system for cloud computing. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on (pp. 933-939). IEEE.

[Habib, 2008] Habib, I. (2008). Virtualization with kvm. Linux Journal, 2008(166), 8.

[Habiba *et al.*, 2013] Habiba, M., Islam, M., & Ali, A. B. M. (2013, July). Access Control Management for Cloud. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on (pp. 485-492). IEEE.

[Haghighi, 2013] Haghighi, M. (2013, June). An Agent-based Multi-model Tool for Simulating Multiple Concurrent Applications in WSNs. In Journal of Advances in Computer Networks (JACN), 5th International Conference on Communication Software and Networks, Malaysia (June 2013).

[Hajivali *et al.*, 2013] Hajivali, M., Fatemi Moghaddam, F., Alrashdan, M. T., & Alothmani, A. Z. (2013, October). Applying an agent-based user authentication and access control model for cloud servers. In ICT Convergence (ICTC), 2013 International Conference on (pp. 807-812). IEEE.

[Halinka, 2008] Halinka, T. (2008). The openQRM users guide.

[Hamadi *et* Benatallah, 2003] Hamadi, R., & Benatallah, B. (2003, January). A Petri net-based model for web service composition. In Proceedings of the 14th Australasian database conference-Volume 17 (pp. 191-200). Australian Computer Society, Inc..

[Hamdi, 2012] Hamdi, M. (2012, May). Security of cloud computing, storage, and networking. In Collaboration Technologies and Systems (CTS), 2012 International Conference on (pp. 1-5). IEEE.

[Han *et al.*, 2011] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In Pervasive computing and applications (ICPCA), 2011 6th international conference on (pp. 363-366). IEEE.

[Han *et al.*, 2012] Han, R., Guo, L., Ghanem, M. M., & Guo, Y. (2012, May). Lightweight resource scaling for cloud applications. In Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on (pp. 644-651). IEEE.

[Han *et* Sim, 2010] Han, T., & Sim, K. M. (2010, March). An ontology-enhanced cloud service discovery system. In Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol. 1, pp. 17-19).

[Hannoun *et al.*, 2000] Hannoun, M., Boissier, O., Sichman, J. S., & Sayettat, C. (2000). MOISE: An organizational model for multi-agent systems. In Advances in Artificial Intelligence (pp. 156-165). Springer Berlin Heidelberg.

[Hassan *et al.*, 2009] Hassan, M. M., Song, B., & Huh, E. N. (2009, February). A framework of sensor-cloud integration opportunities and challenges. In Proceedings of the 3rd international conference on Ubiquitous information management and communication (pp. 618-626). ACM.

[Heras *et al.*, 2012] Heras, S., De la Prieta, F., Julian, V., Rodríguez, S., Botti, V., Bajo, J., & Corchado, J. M. (2012). Agreement technologies and their use in cloud computing environments. Progress in Artificial Intelligence, 1(4), 277-290.

[Heydon *et* Najork, 1999] Heydon, A., & Najork, M. (1999). Mercator: A scalable, extensible web crawler.World Wide Web, 2(4), 219-229.

[Holzer *et* Ondrus, 2011] Holzer, A., & Ondrus, J. (2011). Mobile application market: A developer's perspective. Telematics and Informatics, 28(1), 22-31.

[Hoogendoorn *et* Treur, 2009] Hoogendoorn, M., & Treur, J. (2009). An adaptive multi-agent organization model based on dynamic role allocation. International journal of knowledge-based and intelligent engineering systems, 13(3), 119-139.

[Hooper *et al.*, 2008] Hooper, A. N. D. Y. (2008). Green computing. Communication of the ACM,51(10), 11-13.

[Hopf *et* Schaeffer, 1997] Höpf, M., & Schaeffer, C. F. (1997). Holonic manufacturing systems. InInformation Infrastructure Systems for Manufacturing (pp. 431-438). Springer US.

[Horling *et al.*, 1999] Horling, B., Lesser, V., Vincent, R., Bazzan, A., & Xuan, P. (2000). Diagnosis as an integral part of multi-agent adaptability. In DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings (Vol. 2, pp. 211-219). IEEE.

[Horling *et* Lesser, 2004] Horling, B., & Lesser, V. (2004). A survey of multi-agent organizational paradigms. The Knowledge Engineering Review, 19(4), 281-316.

[Horn, 2001] Horn, P. (2001). Autonomic computing: IBM\'s Perspective on the State of Information Technology.

[Hübner *et al.*, 2002] Hübner, J. F., Sichman, J. S., & Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. InAdvances in artificial intelligence (pp. 118-128). Springer Berlin Heidelberg.

[Hübner *et al.*, 2004] Hübner, J. F., Sichman, J. S., & Boissier, O. (2004). Using the\ mathcal {M} oise+ for a Cooperative Framework of MAS Reorganisation. In Advances in artificial intelligence–SBIA 2004 (pp. 506-515). Springer Berlin Heidelberg.

[Hübner *et al.*, 2010] Hübner, J. F., Boissier, O., Kitio, R., & Ricci, A. (2010). Instrumenting multi-agent organisations with organisational artifacts and agents. Autonomous Agents and Multi-Agent Systems, 20(3), 369-400.

[Hursti, 1997] Hursti, J. (1997). Single sign-on. Helsinki University of Technology, 1, l.

[Hutchins, 1999] Hutchins, D. (1999). Just in time. Gower Publishing, Ltd..

[Iglesias *et al.*, 1998] Iglesias, C. A., Garijo, M., González, J. C., & Velasco, J. R. (1998). Analysis and design of multiagent systems using MAS-CommonKADS. In Intelligent Agents IV Agent Theories, Architectures, and Languages (pp. 313-327). Springer Berlin Heidelberg.

[Iqbal *et* Yousaf, 2013] Iqbal, W., & Yousaf, S. (2013). Formal Modeling of Agent Based Cloud Computing Services using Petri nets. VFAST Transactions on Software Engineering, 1(2), 1-6.

[Islam *et* Habiba, 2012a] Islam, R., & Habiba, M. (2012, May). Collaborative swarm intelligence based Trusted Computing. In Informatics, Electronics & Vision (ICIEV), 2012 International Conference on (pp. 1-6). IEEE.

[Islam *et* Habiba, 2012b] Islam, M., & Habiba, M. (2012, December). Agent based framework for providing security to data storage in cloud. In Computer and Information Technology (ICCIT), 2012 15th International Conference on (pp. 446-451). IEEE.

[Jennings *et al.*, 1995] Jennings, N. R., Corera, J. M., Laresgoiti, I. (1995) Developing industrial multi-agent systems. In: Proceedings of the First International Conference on Multi-agent Systems, (ICMAS-95), 423-430.

[Jennings *et* Wooldridge, 1995] Jennings, N. R., & Wooldridge, M. (1998). Applications of intelligent agents. In Agent technology (pp. 3-28). Springer Berlin Heidelberg.

[Jennings, 2001] Jennings, N. R. (2001). An agent-based approach for building complex software systems. Communications of the ACM, 44(4), 35-41.

[Jensen *et* Veloso, 2000] Jensen, R. M., & Veloso, M. M. (2000). OBDD-Based Universal Planning for Multiple Synchronized Agents in Non-Deterministic Domains. In AIPS (pp. 167-176).

[Jones, 2005] Jones, S. (2005). Toward an acceptable definition of service [service-oriented architecture]. Software, IEEE, 22(3), 87-93.

[Juan *et al.*, 2002] Juan, T., Pearce, A., & Sterling, L. (2002, July). ROADMAP: extending the gaia methodology for complex open systems. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1 (pp. 3-10). ACM.

[Juan *et* Sterling, 2004] Juan, T., & Sterling, L. (2004). The ROADMAP meta-model for intelligent adaptive multi-agent systems in open environments. In Agent-Oriented Software Engineering IV (pp. 53-68). Springer Berlin Heidelberg.

[Jung *et al.*, 2011] Jung, G., Sim, K. M., Kwok, P. C., & Zhang, M. (2011, October). A time-driven adaptive mecahnism for cloud resource allocation. In Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on(pp. 441-446). IEEE.

[Kalagiakos *et* Karampelas, 2011] Kalagiakos, P., & Karampelas, P. (2011, October). Cloud computing learning. In Application of Information and Communication Technologies (AICT), 2011 5th International Conference on (pp. 1-4). IEEE.

[Kamara *et al.*, 2010] Kamara, S., & Lauter, K. (2010). Cryptographic cloud storage. In Financial Cryptography and Data Security (pp. 136-149). Springer Berlin Heidelberg.

[Kang *et al.*, 2011] Kang, S., Kang, S., & Hur, S. (2011, May). A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform. In Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on (pp. 462-467). IEEE.

[Kang *et* Sim, 2011a] Kang, J., & Sim, K. M. (2011, January). Cloudle: An ontology-enhanced cloud service search engine. In Web Information Systems Engineering–WISE 2010 Workshops (pp. 416-427). Springer Berlin Heidelberg.

[Kang *et* Sim, 2011b] Kang, J., & Sim, K. M. (2011, June). Ontology and search engine for cloud computing system. In System Science and Engineering (ICSSE), 2011 International Conference on (pp. 276-281). IEEE.

[Kautz *et al.*, 2006] Kautz H., Selman B. y Hoffmann J (2006). SatPlan: Planning as Satisfiability Abstracts of the 5th International Planning Competition

[Keahey, 2009] Keahey, K. (2009, June). Cloud Computing for Science. In SSDBM (p. 478).

[Kepahart *et* Chess, 2003] Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41-50.

[Kibe *et al.*, 2013] Kibe, S., Yamagiwa, M., & Uehara, M. (2013, March). Grid on Cloud. InAdvanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on (pp. 213-218). IEEE.

[Kliazovich *et al.*, 2012] Kliazovich, D., Bouvry, P., & Khan, S. U. (2012). GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing, 62(3), 1263-1283.

[Koestler, 1969] Koestler, A. (1969). Beyond atomism and holism—the concept of the holon.

[Kolp *et al.*, 2006] Kolp, M., Giorgini, P., & Mylopoulos, J. (2006). Multi-agent architectures as organizational structures. Autonomous Agents and Multi-Agent Systems,13(1), 3-25.

[Krautheim *et al.*, 2009] Krautheim, F. J. (2009, June). Private virtual infrastructure for cloud computing. In Proceedings of the 2009 conference on Hot topics in cloud computing (pp. 5-5). USENIX Association.

[Krol *et al.*, 2012] Krol, D., Kryza, B., Wrzeszcz, M., Dutka, L., & Kitowski, J. (2012). Elastic Infrastructure for Interactive Data Farming Experiments. Procedia Computer Science, 9, 206-215.

[Kusic *et al.*, 2009] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., & Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. Cluster computing, 12(1), 1-15.

[Labidi *et* Lejouad, 1993] Labidi, S., Lejouad, W. (1993) Del'intelligence artificielle distribu'eeaux syst'emes multiagents

[Ladan, 2012] Ladan, M. I. (2012). Web services metrics: A survey and a classification.Journal of Communication and Computer, 9(7), 824-829.

[Lai *et al.*, 2012] Lai, Y. L. (2012). Analyzing strategies of mobile agents on malicious cloud platform with Agent-Based Computational Economic Approach. Expert Systems with Applications.

[Leake *et al.*, 2000] Leake, D. B., Bauer, T., Maguitman, A., & Wilson, D. C. (2000). Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In Proceedings of the AAAI-2000 Workshop on Intelligent Lessons Learned Systems (pp. 33-37).

[Leavitt, 2009] Leavitt, N. (2009). Is cloud computing really ready for prime time. Growth,27(5).

[Leitao *et al.*, 2013] Leitao, P., Inden, U., & Rückemann, C. P. (2013, November). Parallelising Multi-agent Systems for High Performance Computing. In INFOCOMP 2013, The Third International Conference on Advanced Communications and Computation (pp. 1-6).

[Lenk *et al.*, 2009] Lenk, A., Klems, M., Nimis, J., Tai, S., & Sandholm, T. (2009, May). What's inside the Cloud? An architectural map of the Cloud landscape. In Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing (pp. 23-31). IEEE Computer Society.

[Li *et al.*, 2010] Li, A., Yang, X., Kandula, S., & Zhang, M. (2010, November). CloudCmp: comparing public cloud providers. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (pp. 1-14). ACM.

[Li *et al.*, 2011a] Li, Z., Chen, C., & Wang, K. (2011). Cloud computing for agent-based urban transportation systems. Intelligent Systems, IEEE, 26(1), 73-79.

[Li *et al.*, 2011b] Li, B. H., Zhang, L., Ren, L., Chai, X. D., Tao, F., Luo, Y. L., Wang Y., Yin C., Huang G. & Zhao, X. (2011). Further discussion on cloud

manufacturing. Computer Integrated Manufacturing Systems, 17(3), 449-457.

[Li *et al.*, 2012] Li, J., Li, B., Wo, T., Hu, C., Huai, J., Liu, L., & Lam, K. P. (2012). CyberGuarder: A virtualization security assurance architecture for green cloud computing. Future Generation Computer Systems, 28(2), 379-390.

[Li *et al.*, 2013] Li, W., Zhong, Y., Wang, X., & Cao, Y. (2013). Resource Virtualization and Service Selection in Cloud Logistics. Journal of Network and Computer Applications.

[Liberman, 2006] Lieberman, P. (2006). White paper: Wake on lan technology.

[Lim *et al.*, 2009] Lim, S. H., Sharma, B., Nam, G., Kim, E. K., & Das, C. R. (2009, August). MDCSim: A multi-tier data center simulation, platform. In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on (pp. 1-9). IEEE.

[Liu *et al.*, 2009] Liu, L., Wang, H., Liu, X., Jin, X., He, W. B., Wang, Q. B., & Chen, Y. (2009, June). GreenCloud: a new architecture for green data center. In Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session (pp. 29-38). ACM.

[Liu *et al.*, 2011] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST cloud computing reference architecture. NIST special publication, 500, 292

[Lohr, 2007] S. Lohr. Google and ibm join in cloud computing research. In: New York Times, October 2007. Rl: http://www.nytimes.com/2007/10/08/technology/08cloud.html?_r=0

[Lopez-Rodriguez *et al.*, 2011] Lopez-Rodriguez, I., & Hernandez-Tejera, M. (2011). Software agents as cloud computing services. In Advances on Practical Applications of Agents and Multiagent Systems (pp. 271-276). Springer Berlin Heidelberg.

[Lou *et al.*, 2011] Luo, J. Z., JIN, J. H., SONG, A. B., & Dong, F. (2011). Cloud computing: architecture and key technologies. Journal of China Institute of Communications, 32(7), 3-21.

[Loutas *et al.*, 2011] Loutas, N., Kamateri, E., Bosi, F., & Tarabanis, K. (2011, November). Cloud computing interoperability: the state of play. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on (pp. 752-757). IEEE.

[Low *et al.*, 2011] Low, C., Chen, Y., & Wu, M. (2011). Understanding the determinants of cloud computing adoption. Industrial management & data systems, 111(7), 1006-1023.

[Machado, 2009] Machado, G. S., Hausheer, D., & Stiller, B. (2009, October). Considerations on the Interoperability of and between Cloud Computing Standards. In 27th Open Grid Forum (OGF27), G2C-Net Workshop: From Grid to Cloud Networks, Banff, Canada.

[Maes, 1990] Maes, P. (Ed.). (1990). Designing autonomous agents: Theory and practice from biology to engineering and back. MIT press.

[Mahjoub *et al.*, 2011] Mahjoub, M., Mdhaffar, A., Halima, R. B., & Jmaiel, M. (2011, November). A comparative study of the current Cloud Computing technologies and offers. InNetwork Cloud Computing and Applications (NCCA), 2011 First International Symposium on (pp. 131-134). IEEE.

[Manyika *et al.*, 2011] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

[Martin *et al.*, 2003] Martin, R. C. (2003). Agile software development: principles, patterns, and practices. Prentice Hall PTR.

[Mas, 2005] Mas, A. (2005). Agentes software y sistemas multiagente: conceptos, arquitecturas y aplicaciones. Prentice Hall.. ISBN 84-205-4367-5 , pp. 29-64.

[McDougall *et* Anderson, 2010] McDougall, R., & Anderson, J. (2010). Virtualization performance: perspectives and challenges ahead. ACM SIGOPS Operating Systems Review, 44(4), 40-56.

[McKendrick, 2012] McKendrick, J. (2012, March). NIST definition of cloud computing doesn't go far enough. In: ZDNET. http://www.zdnet.com/blog/service-oriented/nist-definition-of-cloud-computing-doesnt-go-far-enough/8634

[Meera *et* Swamynathanb, 2013] Meera, A., & Swamynathan, S. (2013). Agent based Resource Monitoring System in IaaS Cloud Environment. Procedia Technology, 10, 200-207.

[Mell *et* Grance, 2011] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (draft). NIST special publication, 800(145), 7.

[Miao *et al.*, 2011] Miao, X., & Han, J. (2011, October). The Design of a Private Cloud Infrastructure Based on XEN. In Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2011 Tenth International Symposium on (pp. 160-164). IEEE.

[Mietzner *et al.*, 2011] Mietzner, R., Leymann, F., & Unger, T. (2011). Horizontal and vertical combination of multi-tenancy patterns in service-oriented applications.Enterprise Information Systems, 5(1), 59-77.

[Miller, 2008] Miller, M. (2008). Cloud computing: Web-based applications that change the way you work and collaborate online. Que publishing.

[Milojičić et al., 2011] Milojičić, D., Llorente, I. M., & Montero, R. S. (2011). OpenNebula: A Cloud Management Tool. IEEE Internet Computing, 15(2).

[Minson *et* Theodoropoulos, 2008] Minson, R., & Theodoropoulos, G. K. (2008). Distributing RePast agent-based simulations with HLA. Concurrency and Computation: Practice and Experience,20(10), 1225-1256.

[Mintzberg, 1992] Mintzberg, H. (1992). Structure in fives : Designing effective organizations. Prentice-Hall.

[Mintzberg, 1993] Mintzberg, H. (1993). La estructuración de las organizaciones. Barcelona: Ariel.

[Mirkin *et al.*, 2008] Mirkin, A., Kuznetsov, A., & Kolyshkin, K. (2008, July). Containers checkpointing and live migration. In Proceedings of the Linux Symposium (pp. 85-92).

[Mitchell *et al.*, 2012] Mitchell Smith, D., Plummer, D.C., Bittman, T.J., Bova, T., Basso, M., Lheureux, B.J., & Prentice, B (2012, December). Predicts 2013: Cloud Computing Becomes an Integral Part of IT. In: https://www.gartner.com/doc/2263916/predicts--cloud-computing-integral

[Molesini *et* Omicini, 2008] Molesini, A., & Omicini, E. N. E. D. A. (2008, November). Advancing object-oriented standards toward agent-oriented methodologies: SPEM 2.0 on SODA. In 9th Workshop" From Objects to Agents"(WOA 2008)–Evolution of Agent Development: Methodologies, Tools, Platforms and Languages (pp. 108-114).

[Morabito *et al.*, 1999] Morabito, J., Sack, I., & Bhate, A. (1999). Organization modeling : Innovative architectures for the 21st

[Moraitis *et* Spanoudakis, 2004] Moraitis, P., & Spanoudakis, N. (2004, April). Combining gaia and jade for multi-agent systems development. In Fourth International Symposium" From Agent Theory to Agent Implementation"(AT2AI'04), Vienna, Austria.

[Moraitis *et* Spanoudakis, 2006] Moraitis, P., & Spanoudakis, N. (2006). The Gaia2Jade process for multi-agent systems development. Applied Artificial Intelligence, 20(2-4), 251-273.

[More *et al.*, 2014] More, A., Vij, S., & Mukhopadhyay, D. (2014, January). Agent Based Negotiation Using Cloud–An Approach in E-Commerce. In ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I (pp. 489-496). Springer International Publishing.

[Moreno-Luzon *et al.*, 2001] Moreno-Luzón, M. D., Bonet, F. J. P., & Cruz, T. F. G. (2001). Gestión de la Calidad y diseño de organizaciones: teoria y estudio de casos. Pearson Educación, SA.

[Moreno-Vozmediano *et al.*, 2013] Moreno-Vozmediano, R., Montero, R. S., & Llorente, I. M. (2013). Key Challenges in Cloud Computing: Enabling the Future Internet of Services.Internet Computing, IEEE, 17(4), 18-25.

[Moscato *et al.*, 2011] Moscato, F., Aversa, R., Di Martino, B., Fortis, T., & Munteanu, V. (2011, September). An analysis of mOSAIC ontology for Cloud resources annotation. In Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on (pp. 973-980). IEEE.

[Mouratidis *et al.*, 2013] Mouratidis, H., Islam, S., Kalloniatis, C., & Gritzalis, S. (2013). A framework to support selection of cloud providers based on security and privacy requirements. Journal of Systems and Software.

[Needham, 1993] Needham, R. M. (1993, December). Denial of service. In Proceedings of the 1st ACM Conference on Computer and Communications Security (pp. 151-153). ACM.

[Newcomer *et* Lomow, 2004] Newcomer, E., & Lomow, G. (2004). Understanding SOA with web services (independent technology guides). Addison-Wesley Professional.

[Niebert *et al.*, 2008] Niebert, N., El Khayat, I., Baucke, S., Keller, R., Rembarz, R., & Sachs, J. (2008). Network virtualization: A viable path towards the future internet.Wireless Personal Communications, 45(4), 511-520.

[Noronha *et al.*, 2008] Noronha, R., & Panda, D. K. (2008, September). IMCa: a high performance caching front-end for GlusterFS on InfiniBand. In Parallel Processing, 2008. ICPP'08. 37th International Conference on (pp. 462-469). IEEE.

[Nouman *et al.*, 2013] Nouman, A., Anagnostou, A., & Taylor, S. J. (2013, October). Developing a Distributed Agent-Based and DES Simulation Using poRTIco and Repast. InDistributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on (pp. 97-104). IEEE.

[Nuñez *et al.*, 2012] Núñez, A., Andrés, C., & Merayo, M. G. (2012). MAScloud: a framework based on multi-agent systems for optimizing cost in cloud computing. In Computational Collective Intelligence. Technologies and Applications (pp. 436-445). Springer Berlin Heidelberg.

[Nwana, 1996] Nwana, H. S. (1996). Software agents: An overview. The knowledge engineering review, 11(03), 205-244.

[O'connor, 2003] O'connor, M. A. (2003). U.S. Patent No. 6,564,228. Washington, DC: U.S. Patent and Trademark Office.

[Oberheide *et al.*, 2008] Oberheide, J., Veeraraghavan, K., Cooke, E., Flinn, J., & Jahanian, F. (2008, June). Virtualized in-cloud security services for mobile devices. In Proceedings of the First Workshop on Virtualization in Mobile Computing (pp. 31-35). ACM.

[Ogrizovic *et al.*, 2010] Ogrizovic, D., Svilicic, B., & Tijan, E. (2010, May). Open source science clouds. In MIPRO, 2010 Proceedings of the 33rd International Convention (pp. 1189-1192). IEEE.

[Oguchi *et* Yamamoto, 2008] Yamamoto, V. Y. O. V. T. (2008). Server virtualization technology and its latest trends. Fujitsu Sci. Tech. J, 44(1), 46-52.

[OMG, 2008] Object Management Group (2008, April). Software & Systems Process Engineering Metamodel Specification (SPEM) version 2.0.

[Omicini *et al.*, 2001] A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors (2001, march). Coordination of Internet Agents: Models, Technologies, and Applications. Springer Verlag, 2001.

[Omicini *et al.*, 2001] Omicini, A. (2001, January). SODA: Societies and infrastructures in the analysis and design of agent-based systems. In Agent-oriented software engineering (pp. 185-193). Springer Berlin Heidelberg.

[Ommicini *et al.*, 2008] Omicini, A., Ricci, A., & Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems, 17(3), 432-456.

[Ortiz, 2011] Ortiz, S. (2011). The problem with cloud-computing standardization. Computer, 44(7), 13-16.

[Othame *et* Hebri, 2012] Othmane, B., & Hebri, R. S. A. (2012, June). Cloud computing & multi-agent systems: A new promising approach for distributed data mining. In Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on (pp. 111-116). IEEE.

[Padgham *et* Winikoff, 2002] Padgham, L., & Winikoff, M. (2002, November). Prometheus: A pragmatic methodology for engineering intelligent agents. In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies (pp. 97-108).

[Padgham *et* Winikoff, 2003] Padgham, L., & Winikoff, M. (2003). Prometheus: A methodology for developing intelligent agents. In Agent-oriented software engineering III (pp. 174-185). Springer Berlin Heidelberg.

[Pal *et al.*, 2011] Pal, S., Khatua, S., Chaki, N., & Sanyal, S. (2011). A new trusted and collaborative agent based approach for ensuring cloud security. arXiv preprint arXiv:1108.4100.

[Palmieri *et al.*, 2013] Palmieri, F., Buonanno, L., Venticinque, S., Aversa, R., & Martino, B. D. (2013). A distributed scheduling framework based on selfish autonomous agents for federated cloud environments. Future Generation Computer Systems.

[Parashar *et* Hariri, 2005] Parashar, M., & Hariri, S. (2005). Autonomic computing: An overview. InUnconventional Programming Paradigms (pp. 257-269). Springer Berlin Heidelberg.

[Parunak *et al.*, 1998] Parunak, H. Van Dyke, Robert Savit and Rick L. Riolo (1998): Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and User's Guide. In Proceedings of Multi-agent systems and Agent-based Simulation (pp. 10-25).

[Parunak, 1997] Parunak, H. V. D. (1997). " Go to the ant": Engineering principles from natural multi-agent systems. Annals of Operations Research, 75, 69-101.

[Patel *et al.*, 2009] Patel, P., Ranabahu, A. H., & Sheth, A. P. (2009). Service level agreement in cloud computing.

[Pavón *et al.*, 2005] Pavón, J., Gómez-Sanz, J. J., & Fuentes, R. (2005). The INGENIAS methodology and tools. Agent-oriented methodologies, 9, 236-276.

[Pavón *et* Gómez-Sanz, 2003] Pavón, J., & Gómez-Sanz, J. (2003). Agent oriented software engineering with INGENIAS. In Multi-Agent Systems and Applications III (pp. 394-403). Springer Berlin Heidelberg.

[Peltzamn, 1976] Peltzman, S. (1976). Toward a more general theory of regulation.

[Pervez *et al.*, 2010] Pervez, Z., Lee, S., & Lee, Y. K. (2010, February). Multi-tenant, secure, load disseminated SaaS architecture. In Advanced Communication Technology (ICACT), 2010 The 12th International Conference on (Vol. 1, pp. 214-219). IEEE.

[Petcu, 2011] Petcu, D. (2011). Portability and interoperability between clouds: challenges and case study. In Towards a Service-Based Internet (pp. 62-74). Springer Berlin Heidelberg. ISBN: 978-3-642-24754-5

[Petrocci, 2010] Petrucci, V., Loques, O., & Mossé, D. (2010, March). Dynamic optimization of power and performance for virtualized server clusters. In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 263-264). ACM.

[Písačka *et al.*, 2012] Písačka, J., Hron, M., Janky, F., & Pánek, R. (2012). Cluster storage for COMPASS tokamak. Fusion Engineering and Design, 87(12), 2238-2241.

[Pressman *et* Jawadekar, 1992] Pressman, R. S., & Jawadekar, W. S. (1987). Software engineering. New York 1992.

[Raghavendra *et al.*, 2008] Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., & Zhu, X. (2008, March). No power struggles: Coordinated multi-level power management for the data center. In ACM SIGARCH Computer Architecture News (Vol. 36, No. 1, pp. 48-59). ACM.

[Rahmani *et al.*, 2009] Rahmani, A., Ji, M., Mesbahi, M., & Egerstedt, M. (2009). Controllability of multi-agent systems from a graph-theoretic perspective. SIAM Journal on Control and Optimization, 48(1), 162-186.

[Ramaswamy *et al.*, 2011] Ramaswamy, A., Balasubramanian, A., Vijaykumar, P., & Varalakshmi, P. (2011, June). A Mobile Agent based Approach of ensuring Trustworthiness in the Cloud. In Recent Trends in Information Technology (ICRTIT), 2011 International Conference on (pp. 678-682). IEEE.

[Ramchurn *et al.*, 2004] Ramchurn, S. D., Huynh, D., & Jennings, N. R. (2004). Trust in multi-agent systems. The Knowledge Engineering Review, 19(01), 1-25.

[Rao *et* Georgeff, 1992] Rao, A. S., & Georgeff, M. P. (1992). An abstract architecture for rational agents. KR'92, 439-449., pp. 439 - 449.

[Rappa, 2004] Rappa, M. A. (2004). The utility business model and the future of computing services. IBM Systems Journal, 43(1), 32-42

[Rapport, 2010] Rapport, M (December, 2010). Some Experts Say Cloud Computing is Nothing New. Credit Union Times.

[Razavi *et al.*, 2005] Razavi, R., Perrot, J., & Guelfi, N. (2005). Adaptive modeling: an approach and a method for implementing adaptive agents. Lecture Notes in Artificial Intelligence , 3446, 136-148.

[Reason *et* Bradbury, 2001] Reason, P., & Bradbury, H. (Eds.). (2001). Handbook of action research: Participative inquiry and practice. Sage.

[Redkar *et* Guidici, 2011] Redkar, T., & Guidici, T. (2011). Windows Azure Platform (Vol. 1). Apress.

[Regola *et* Ducom, 2010] Regola, N., & Ducom, J. C. (2010, November). Recommendations for virtualization technologies in high performance computing. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 409-416). IEEE.

[Reitbauer *et al.*, 2004] 57. Reitbauer, A., Battino, A., Karageorgos, A., Mehandjiev, P., Valckenaers, P., & Saint-Germain, B. (2004). The MaBE middleware: extending multi-agent systems to enable open business collaboration. International Conference on Information Techonology for Balanced Automation Systems in Manufacturing and Services - BASYS04

[Reuben, 2007] Reuben, J. S. (2007). A survey on virtual machine security. Helsinki University of Technology.

[Ricci *et al.*, 2006] Ricci, A., Viroli, M., & Omicini, A. (2006). Programming MAS with artifacts. InProgramming multi-agent systems (pp. 206-221). Springer Berlin Heidelberg.

[Ricci *et al.*, 2007] Ricci, A., Viroli, M., & Omicini, A. (2007, May). Give agents their artifacts: the A&A approach for engineering working environments in MAS. In Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (p. 150). ACM.

[Rimal *et al.*, 2009] Rimal, B. P., Choi, E., & Lumb, I. (2009, August). A taxonomy and survey of cloud computing systems. In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on (pp. 44-51). Ieee.

[Rings *et al.*, 2009] Rings, T., Caryer, G., Gallop, J., Grabowski, J., Kovacikova, T., Schulz, S., & Stokes-Rees, I. (2009). Grid and cloud computing: opportunities for integration with the next generation network. Journal of Grid Computing, 7(3), 375-393.

[Rochwerger *et al.*, 2009] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero R., Wolfthal Y., Elmroth E., Cáceres J., Ben-Yehuda M., Emmerich W. & Galán, F. (2009). The reservoir model and architecture for open federated cloud computing. IBM Journal of Research and Development, 53(4), 4-1

[Rodriguez González, 2010] Rodríguez González, S. (2010). Modelo adaptativo para Organizaciones virtuales de agentes.

[Ross *et* Westerman, 2004] Ross, J. W., & Westerman, G. (2004). Preparing for utility computing: The role of IT architecture and relationship management. IBM systems journal, 43(1), 5-19.

[RUP, 2003] RUP, I. (2003). Rational Unified Process. Engenharia de Software, 52.

[Russel *et* Norvig, 1995] Russell S. and Norvig P. (1995). Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[Russom *et al.*, 2011] Russom, P. (2011). Big data analytics. TDWI Best Practices Report, Fourth Quarter.

[Sabater, 2002] Sabater, J., Sierra, C., Parsons, S., & Jennings, N. R. (2002). Engineering Executable Agents using Multi-context Systems. Journal of Logic and Computation, 12(3), 413-442.

[Sabharwal, 2013] Sabharwal, N., & Shankar, R. (2013). Apache Cloudstack Cloud Computing. Packt Publishing Ltd.

[Sahoo *et al.*, 2010] Sahoo, J., Mohapatra, S., & Lath, R. (2010, April). Virtualization: A survey on concepts, taxonomy and associated security issues. In Computer and Network Technology (ICCNT), 2010 Second International Conference on (pp. 222-226). IEEE.

[Sandberg *et al.*, 1985] Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., & Lyon, B. (1985, June). Design and implementation of the Sun network filesystem. In Proceedings of the Summer USENIX conference (pp. 119-130).

[Sandhu *et* Kaur, 2013] Sandhu, A., & Kaur, M. A Review of Evaluation of Various Cloud Based Simulators.

[Schaller, 1997] Schaller, R. R. (1997). Moore's law: past, present and future. Spectrum, IEEE,34(6), 52-59.

[Schank, 1982] Schank, R. C. (1982). Dynamic memory: A theory of learning in people and computers. New York: Cambridge University Press, 1982

[Schertler, 1998] Schertler, W. (1998). Virtual Enterprises in Tourism: Folklore and Facts—Conceptual Challenges for Academic Research—. In Information and Communication Technologies in Tourism 1998 (pp. 278-288). Springer Vienna.

[Schreiber *et al.*, 2000] Schreiber, G. (Ed.). (2000). Knowledge engineering and management: the CommonKADS methodology. the MIT Press.

[Sefraoui *et al.*, 2012] Sefraoui, O., Aissaoui, M., & Eleuldj, M. (2012). OpenStack: Toward an Open-Source Solution for Cloud Computing. International Journal of Computer Applications, 55.

[Sempolinski *et* Thain, 2010] Sempolinski, P., & Thain, D. (2010, November). A comparison and critique of eucalyptus, opennebula and nimbus. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 417-426). Ieee.

[Sethia *et* Karlapalem, 2011] Sethia, P., & Karlapalem, K. (2011). A multi-agent simulation framework on small Hadoop cluster. Engineering Applications of Artificial Intelligence, 24(7), 1120-1127.

[Shen *et al.*, 2006] Shen, W., Wang, L., & Hao, Q. (2006). Agent-based distributed manufacturing process planning and scheduling: a state-of-the-

art survey. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 36(4), 563-577.

[Shenker, 1995] Shenker, S. (1995). Fundamental design issues for the future Internet.Selected Areas in Communications, IEEE Journal on, 13(7), 1176-1188.

[Sim, 2009] Sim, K. M. (2009, December). Agent-based cloud commerce. In Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on (pp. 717-721). IEEE.

[Sim, 2010] Sim, K. M. (2010). Towards complex negotiation for cloud economy. InAdvances in Grid and Pervasive Computing (pp. 395-406). Springer Berlin Heidelberg.

[Sim, 2012a] Sim, K. M. (2012). Agent-based Cloud Computing. IEEE TRANSACTIONS ON SERVICES COMPUTING,.

[Sim, 2012b] Sim, K. M. (2012). Complex and concurrent negotiations for multiple interrelated e-markets.

[Sim, 2013] Sim, K. M. Cloud intelligence: agents within an InterCloud.

[Sims *et al.*, 2008] Sims, M., Corkill, D., & Lesser, V. (2008). Automated organization design for multi-agent systems. Autonomous Agents and Multi-Agent Systems, 16(2), 151-185.

[Singh *et* Malhotra, 2012] Singh, A., & Malhotra, M. (2012). Agent Based Framework for Scalability in Cloud Computing. International Journal of Computer Science & Engineering Technology (IJCSET), 3(4), 41-45.

[So *et* Durfee, 1993] So, Y. P., & Durfee, E. H. (1993). An organizational self-design model for organizational change. Ann Arbor, 1001, 48109.

[Son *et* Sim, 2012] Son, S., & Sim, K. M. (2012). A Price-and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 42(3), 713-728.

[Song *et al.*, 2009] Song, S., Ge, R., Feng, X., & Cameron, K. W. (2009). Energy profiling and analysis of the HPC challenge benchmarks. International Journal of High Performance Computing Applications, 23(3), 265-276.

[Sosinsky, 2011] Sosinsky, B. (2010). Cloud computing bible (Vol. 762). John Wiley & Sons.

[Sotomayor, 2009] Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. Internet Computing, IEEE, 13(5), 14-22. ISSN : 1089-7801

[Spanoudakis *et al.*, 2009] Spanoudakis, N., & Moraitis, P. (2009). Gaia agents implementation through models transformation. In Principles of Practice in Multi-Agent Systems (pp. 127-142). Springer Berlin Heidelberg.

[Staab *et al.*, 2003] Staab, S., Van der Aalst, W., Benjamins, V. R., Sheth, A., Miller, J. A., Bussler, C., ... & Gannon, D. (2003). Web services: been there, done that?. Intelligent Systems, IEEE, 18(1), 72-85.

[Stone *et* Veloso, 2000] Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. Autonomous Robots, 8(3), 345-383.

[Strauch *et al.*, 2011] Strauch, C., Sites, U. L. S., & Kriha, W. (2011). NoSQL databases. URL: http://www. christof-strauch. de/nosqldbs. pdf (дата обращения 07.11. 2012).

[Subashini *et al.*, 2011] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 34(1), 1-11.

[Sullivan *et* Anderson, 1989] Sullivan, M., & Anderson, D. (1989, June). Marionette: A system for parallel distributed programming using a master/slave model. In Distributed Computing Systems, 1989., 9th International Conference on (pp. 181-188). IEEE.

[Surendran *et al.*, 2012] Surendran, R., & Varthini, B. P. (2012, December). Intelligent based large scale multi agent's resource management on shopping service with security. InEngineering (NUiCONE), 2012 Nirma University International Conference on(pp. 1-6). IEEE.

[Szer *et al.*, 2005] Szer, D., Charpillet, F., & Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In 21st Conference on Uncertainty in Artificial Intelligence-UAI'2005.

[Taboada *et al.*, 2013] Taboada, M., Cabrera, E., Epelde, F., Iglesias, M. L., & Luque, E. (2013). Using an Agent-Based Simulation for Predicting the Effects of Patients Derivation Policies in Emergency Departments. Procedia Computer Science,18, 641-650.

[Tai *et al.*, 2010] Tai, S., Nimis, J., Lenk, A., & Klems, M. (2010, May). Cloud service engineering. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2 (pp. 475-476). ACM.

[Talia, 2011] Talia, D. (2011). Cloud Computing and Software Agents: Towards Cloud Intelligent Services. In WOA (pp. 2-6).

[Talia, 2012] Talia, D. (2012). Clouds meet agents: Toward intelligent cloud services.Internet Computing, IEEE, 16(2), 78-81.

[Talib *et al.*, 2010] Talib, A. M., Atan, R., Abdullah, R., & Murad, M. A. A. (2010). Security framework of cloud data storage based on multi agent system architecture: Semantic literature review. Computer and Information Science, 3(4), p175.

[Talib *et al.*, 2011] Talib, A. M., Atan, R., Abdullah, R., & Azrifah, M. (2011, September). CloudZone: Towards an integrity layer of cloud data storage based on multi agent system architecture. In Open Systems (ICOS), 2011 IEEE Conference on (pp. 127-132). IEEE.

[Talib *et al.*, 2011a] Talib, A. M., Atan, R., Abdullah, R., & Azrifah, M. (2011, September). CloudZone: Towards an integrity layer of cloud data storage based on multi agent system architecture. In Open Systems (ICOS), 2011 IEEE Conference on (pp. 127-132). IEEE.

[Talib *et al.*, 2011b]Talib, A. M., Atan, R., Abdullah, R., & Murad, M. A. A. (2011). Towards new data access control technique based on multi agent system architecture for cloud computing. In Digital Information Processing and Communications (pp. 268-279). Springer Berlin Heidelberg.

[Talib *et al.*, 2011c] Talib, A. M., Atan, R., Abdullah, R., & Murad, M. A. A. (2011). Multi Agent System Architecture Oriented Prometheus Methodology Design to Facilitate Security of Cloud Data Storage. Journal of Software Engineering, 5(3).

[Talib *et al.*, 2012a]Talib, A. M., Atan, R., Abdullah, R., & Murad, A. (2012, March). Security framework of cloud data storage based on Multi Agent system architecture-A pilot study. In Information Retrieval & Knowledge Management (CAMP), 2012 International Conference on (pp. 54-59). IEEE.

[Talib *et al.*, 2012b] Talib, A. M., Atan, R. B., Abdullah, R., & Murad, M. A. A. (2012). Towards a Comprehensive Security Framework of Cloud Data Storage Based on Multi Agent System Architecture. J. Information Security, 3(4), 295-306.

[Talib *et al.*, 2012c] Talib, A. M., Atan, R., Abdullah, R., & Murad, M. A. A. (2012). Security Facilitation in Collaborative Cloud Data Storage Implementation Environment Based on Multi Agent System Architecture. Journal of Software Engineering, 6(3), 49-64.

[Tan *et* Wang, 2010] Tan, L., & Wang, N. (2010, August). Future internet: The internet of things. InAdvanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on (Vol. 5, pp. V5-376). IEEE.

[Tao *et al.*, 2011] Tao, F., Zhang, L., Venkatesh, V. C., Luo, Y., & Cheng, Y. (2011). Cloud manufacturing: a computing and service-oriented

manufacturing model.Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 225(10), 1969-1976.

[Tapia *et al.*, 2013] Tapia, D. I., Alonso, R. S., García, Ó., de la Prieta, F., & Pérez-Lancho, B. (2013, January). Cloud-IO: Cloud Computing Platform for the Fast Deployment of Services over Wireless Sensor Networks. In 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing (pp. 493-504). Springer Berlin Heidelberg.

[Tesauro *et al.*, 2004] Tesauro, G., Chess, D. M., Walsh, W. E., Das, R., Segal, A., Whalley, I., Kepart J.O. & White, S. R. (2004, July). A multi-agent systems approach to autonomic computing. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1 (pp. 464-471). IEEE Computer Society.

[Tianfield, 2011] Tianfield, H. (2011, October). Cloud computing architectures. In Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on (pp. 1394-1399). IEEE.

[Tripathi *et* Jigeesh, 2013] Tripathi, S., & Jigeesh, N. (2013). A Review of Factors That Affect Cloud Computing Adoption. IUP Journal of Computer Sciences, 7(4).

[Turner IV *et al.*, 2006] Turner IV, W. P., PE, J., Seader, P. E., & Brill, K. J. (2006). Tier Classification Define Site Infrastructure Performance. Uptime Institute, 17.

[Ullah *et* Xuefeng, 2013] Ullah, S., & Xuefeng, Z. (2013). Cloud Computing Research Challenges. arXiv preprint arXiv:1304.3203.

[Urgaonkar *et al.*, 2005] Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., & Tantawi, A. (2005, June). An analytical model for multi-tier internet services and its applications. In ACM SIGMETRICS Performance Evaluation Review (Vol. 33, No. 1, pp. 291-302). ACM.

[Van der Aalst, 1998] van der Aalst, W. M. (1998). The application of Petri nets to workflow management. Journal of circuits, systems, and computers, 8(01), 21-66.

[Van *et al.*, 2009a] Van, H. N., Dang Tran, F., & Menaud, J. M. (2009, May). Autonomic virtual resource management for service hosting platforms. In Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing (pp. 1-8). IEEE Computer Society.

[Van *et al.*, 2009b] Van, H. N., Tran, F. D., & Menaud, J. M. (2009, October). SLA-aware virtual resource management for cloud infrastructures. In Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on (Vol. 1, pp. 357-362). IEEE.

[Vaquero *et al.*, 2009] Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50-55.

[Vázquez-Salceda *et al.*, 2005] Vázquez-Salceda, J., Dignum, V., & Dignum, F. (2005). Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems, 11(3), 307-360.

[Vazquez-Salceda, 2003] Vázquez-Salceda, J. (2003). The role of Norms and Electronic Institutions in Multi-Agent Systems applied to complex domains. The HARMONIA framework.AI Communications, 16(3), 209-212.

[Venkataramana *et* Padmavathamma, 2012] Venkataramana, K., & Padmavathamma, M. (2012). Agent Based approach for Authentication in Cloud. IRACST-International Journal of Computer Science and Information Technology & Security, 2(3), 598-603.

[Venticinque *et al.*, 2010] Venticinque, S., Aversa, R., Di Martino, B., Rak, M., & Petcu, D. (2011, January). A cloud agency for SLA negotiation and management. In Euro-Par 2010 Parallel Processing Workshops (pp. 587-594). Springer Berlin Heidelberg.

[Venticinque *et al.*, 2011] Venticinque, S., Aversa, R., Di Martino, B., Rak, M., & Petcu, D. (2011, January). A cloud agency for SLA negotiation and management. In Euro-Par 2010 Parallel Processing Workshops (pp. 587-594). Springer Berlin Heidelberg.

[Venticinque *et al.*, 2012] Venticinque, S., Tasquier, L., & Di Martino, B. (2012, July). Agents based cloud computing interface for resource provisioning and management. InComplex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on (pp. 249-256). IEEE.

[Vinothina *et al.*, 2012] Vinothina, V., Sridaran, D. R., & PadmavathiGanapathi, D. (2012). A survey on resource allocation strategies in cloud computing. International Journal of Advanced Computer Science and Applications, 3(6).

[Von Laszewski *et al.*, 2012] von Laszewski, G., Diaz, J., Wang, F., & Fox, G. C. (2012, June). Comparison of multiple cloud frameworks. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 734-741). IEEE

[Voras *et al.*, 2011] Voras, I., Mihaljevic, B., Orlic, M., Pletikosa, M., Zagar, M., Pavic, T., Mihaljevic, B., Orlic, M., Pletikosa, M., Zagar, M., Pavic, T., Zimmer, K., Cavrak, I., Paunovic, V., Bosnic, I.& Tomic, S. (2011, May). Evaluating open-source cloud computing solutions. InMIPRO, 2011 Proceedings of the 34th International Convention (pp. 209-214). IEEE.

[Vouk, 2008] A Vouk, M. (2008). Cloud computing–issues, research and implementations. CIT. Journal of Computing and Information Technology, 16(4), 235-246.

[Wang *et al.*, 2008] Wang, L., Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., & Fu, C. (2010). Cloud Computing: a Perspective Study. New Generation Computing, 28(2), 137-146.

[Wang *et* Shen, 2011] Wang, K., & Shen, Z. (2011). Artificial societies and GPU-based cloud computing for intelligent transportation management. Intelligent Systems, IEEE, 26(4), 22-28.

[Wei *et al.*, 2010a] Wei, G., Vasilakos, A. V., Zheng, Y., & Xiong, N. (2010). A game-theoretic method of fair resource allocation for cloud computing services. The Journal of Supercomputing, 54(2), 252-269.

[Wei *et al.*, 2010b] Wei, L., Junzhou, L., Bo, L., Xiao, Z., & Jiuxin, C. (2010, October). Multi-agent based QoS-aware service composition. In Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on (pp. 3125-3132). IEEE.

[Wei *et al.*, 2012] Wei, Y., & Blake, M. B. (2012, June). An Agent-Based Services Framework with Adaptive Monitoring in Cloud Environments. In Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on (pp. 4-9). IEEE.

[Wei *et* Blake, 2013] Wei, Y., & Blake, M. B. (2013, March). Adaptive Service Workflow Configuration and Agent-based Virtual Resource Management in the Cloud. In Proceedings of IEEE International Conference on Cloud Engineering, San Francisco, CA.

[Wei, 2010] Wei, Y., & Blake, M. B. (2010). Service-Oriented Computing and Cloud Computing: Challenges and Opportunities. IEEE Internet Computing, 14(6).

[Weissman, 2009] Weissman, C. D., & Bobrowski, S. (2009, June). The design of the force. com multitenant internet application development platform. In SIGMOD Conference(pp. 889-896).

[Wen *et al.*, 2012] Wen, X., Gu, G., Li, Q., Gao, Y., & Zhang, X. (2012, May). Comparison of open-source cloud management platforms: OpenStack and OpenNebula. InFuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on (pp. 2457-2461). IEEE.

[Weyns *et al.*, 2004] Weyns, D., Schelfthout, K., Holvoet, T., & Glorieux, O. (2004). Role based model for adaptive agents. Forth Symposium on Adaptive Agents and Multiagent Systems at the AISB04 Convention

[Wickremasinghe *et al.*, 2010] Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (2010, April). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on (pp. 446-452). IEEE.

[Wittek *et* Rubio-Campillo, 2012] Wittek, P., & Rubio-Campillo, X. (2012, December). Scalable agent-based modelling with cloud HPC resources for social simulations. In Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on (pp. 355-362). IEEE.

[Woods *et* Barbacci, 1999] Woods, S. G., & Barbacci, M. R. (1999). Architectural evaluation of collaborative agent-based systems. Technical Report CMU/SEI-99-TR-025, SEI, Carnegie Mellon University, Pittsburgh, USA, 1999, 1999.

[Wooldridge *et al.*, 1996]Wooldridge, M., Bussmann, S., Klosterberg, M. (1996) Production sequencing as negotiation. In: Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Systems, London.

[Wooldridge *et al.*, 2000] Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems, 3(3), 285-312.

[Wooldridge *et* Jennings, 1995] Wooldridge M. y Jennings N. R. (1995). Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, vol. 10(2) pp. 115-152, 1995.

[Wooldridge, 2002] Wooldridge, M. (2002). An Introduction to MultiAgent Systems. Chichester, England, John Wiley & Sons, ISBN 047149691X.

[Xu *et al.*, 2008] Xu, X., Zhou, F., Wan, J., & Jiang, Y. (2008, December). Quantifying performance properties of virtual machine. In Information Science and Engineering, 2008. ISISE'08. International Symposium on (Vol. 1, pp. 24-28). IEEE.

[Xu *et al.*, 2012] Xu, X. (2012). From cloud computing to cloud manufacturing. Robotics and computer-integrated manufacturing, 28(1), 75-86.

[Yang *et al.*, 2011] Yang, S. Y., Lee, D. L., Chen, K. Y., & Hsu, C. L. (2011). Energy-saving information multi-agent system with web services for cloud computing. In Security-Enriched Urban Computing and Smart Grid (pp. 222-233). Springer

[Yang, 2012] Yang, S. Y. (2012). A novel cloud information agent system with Web service techniques: Example of an energy-saving multi-agent system. Expert Systems with Applications.

[You *et al.*, 2009] You, X., Xu, X., Wan, J., & Yu, D. (2009, August). Ras-m: Resource allocation strategy based on market mechanism in cloud computing. In ChinaGrid Annual Conference, 2009. ChinaGrid'09. Fourth (pp. 256-263). IEEE.

[Youseff *et al.*, 2006] Youseff, L., Wolski, R., Gorda, B., & Krintz, C. (2006, January). Paravirtualization for HPC systems. In Frontiers of High Performance Computing and Networking–ISPA 2006 Workshops (pp. 474-486). Springer Berlin Heidelberg.

[Youseff *et al.*, 2008] Youseff, L., Butrico, M., & Da Silva, D. (2008, November). Toward a unified ontology of cloud computing. In Grid Computing Environments Workshop, 2008. GCE'08 (pp. 1-10). IEEE.

[Yu, 2011] Yu, E. (2011). Modelling strategic relationships for process reengineering.Social Modeling for Requirements Engineering, 11, 2011.

[Yun *et al.*, 2012] Yun, Y., Li, Y., & Han, H. (2012, November). A Mobile Agent-Based Secure and Efficient Task Allocation Algorithm for Cloud& Client Computing. InMultimedia Information Networking and Security (MINES), 2012 Fourth International Conference on (pp. 1-3). IEEE.

[Yuriyama *et* Kushida, 2010] Yuriyama, M., & Kushida, T. (2010, September). Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing. InNetwork-Based Information Systems (NBiS), 2010 13th International Conference on (pp. 1-8). IEEE.

[Zambonelli *et al.*, 2001] Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2001, January). Organisational abstractions for the analysis and design of multi-agent systems. In Agent-Oriented Software Engineering (pp. 235-251). Springer Berlin Heidelberg.

[Zambonelli *et al.*, 2003] Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM), 12(3), 317-370.

[Zambonelli *et al.*, 2004] Zambonelli, F., Gleizes, M.-P., Mamei, M., & Tolksdorf, R. (2004). Spray computers: frontiers of self-organisation for pervasive computing. Workshop on Enabling Techonologies: Infrastructure for Collaborative Enterprises - WETICE04 (págs. 397-402). IEEE Computer Society

[Zeng *et al.*, 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for web services composition. Software Engineering, IEEE Transactions on, 30(5), 311-327.

[Zhang *et al.*, 2010] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, 1(1), 7-18.

[Zhang *et al.*, 2011] Zhang, F., Chen, J., Chen, H., & Zang, B. (2011, October). Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (pp. 203-216). ACM.

[Zhang *et al.*, 2012] Zhang, Y. Q., & Li, W. M. (2012). Research and Design of the High-Performance Cluster Architecture Based on the MySQL and NoSQL. Advanced Materials Research, 460, 313-316.

[Zhang *et* Zhou, 2009] Zhang, L. J., & Zhou, Q. (2009, July). CCOA: Cloud computing open architecture. In Web Services, 2009. ICWS 2009. IEEE International Conference on (pp. 607-616). IEEE.

[Zhu *et* Wang, 2008] Zhu, J., & Wang, W. (2008, December). New-knowledge-view based ontology cloud model. In Computer Science and Software Engineering, 2008 International Conference on (Vol. 5, pp. 1140-1143). IEEE.

ANEXO A
ANÁLISIS Y DISEÑO DEL SISTEMA +CLOUD MEDIANTE LA METODOLOGÍA GORMAS

Las primeras metodologías que surgieron para el desarrollo de SMA se clasifican como *orientadas al agente* [Hannoun *et al.*, 2000], debido a que en ellas no se introduce explícitamente el concepto de *organización* [Rodriguez González, 2010]. En este tipo de metodologías el diseñador se centra en las acciones individuales de los agentes, mientras que las estructuras sociales y los protocolos de interacción entre los individuos no se modelan, sino que forman parte de la descripción de los agentes individuales. Todas estas metodologías ya han sido utilizadas por lo que han demostrado su eficacia y versatilidad. Por contra, estas metodologías de diseño presentan algunas desventajas como la incapacidad para predecir comportamientos, establecer estrategias o aprovechar las características del dominio donde se integran. Los sistemas que se diseñaban eran generalmente cerrados y no permitían que agentes externos entraran en ellos [Zambonelli *et al.*, 2003].

Para superar estos retos, ha surgido una nueva corriente metodológica que permite diseñar los SMA siguiendo principios de *orientación a la organización* [Zambonelli *et al.*, 2003] [Ferber *et al.*, 2004]. Gracias a este nuevo modelo se permite el diseño mediante SMA de entornos heterogéneos y abiertos. Es decir, estas metodologías permiten que agentes externos accedan a la funcionalidad del sistema, pero obligándoles a cumplir un conjunto de normas sociales. En este tipo de metodologías, el diseñador se centra desde un principio en la organización del sistema, analizando el SMA desde una perspectiva global donde los objetivos de la organización representan una descripción a alto nivel de los propósitos de la sociedad.

GORMAS (*Guidelines for Organization-based MultiAgent Systems*) [Argente, 2008] es la metodología utilizada en este trabajo. Es una guía metodológica para el diseño de SMA abiertos desde la perspectiva de las organizaciones humanas, basándose en la teoría de la organización. Sus características, junto con las fases que propone para cubrir el análisis, diseño de la estructura organizativa y diseño de la dinámica de la organización se verán en los siguientes subapartados. Así pues, el apartado A.1 presenta un pequeño resumen de la metodología donde se destacan sus características principales. Los apartados siguientes describen el análisis (A.2), diseño (A.3) y diseño de la dinámica (A.4) del SMA organizativo propuesto en el marco de este trabajo de investigación.

A.1 DESCRIPCIÓN DE LA METODOLOGÍA

Para el análisis y diseño del sistema se ha utilizado la guía metodológica GORMAS (*Guidelines for ORganization-based MultiAgent Systems*) [Argente, 2008] [Argente *et al.*, 2011] que facilita la especificación de SMA abiertos siguiendo la perspectiva de las organizaciones humanas. El modelo descrito por la guía metodológica GORMAS extiende a los modelos propuestos en la metodología ANEMONA [Botti *et* Giret, 2008], que a su vez es una extensión de INGENIAS [Pavón *et* Gómez-Sanz, 2003].

Antes de analizar y diseñar el sistema mediante esta metodología, conviene conocer los seis meta-modelos en los que se basa para describir cualquier sistema. A continuación se presentan brevemente los seis metamodelos que incluye el modelo:

- El **meta-modelo de agente** permite describir a los *agentes* como entidades individuales, pero sin referenciar su comportamiento social.
  En el marco de la metodología, y por lo tanto del meta-modelo asociado, cabe destacar el uso del concepto de *Agente Abstracto* (*A-Agente*) para definir un holón [Koestler, 1969] durante las fases del análisis y diseño. Gracias a este concepto es posible tratar a uno o varios agentes como una entidad única con responsabilidades y objetivos asociados. Cada A-Agente se específica en términos de la funcionalidad asociada (tareas y servicios), los objetivos que persigue y los mecanismos de razonamiento que utiliza.
- El **meta-modelo de actividad** permite refinar las *tareas* y *objetivos* en subtareas y subobjetivos más concretos.
  Para ello, permite describir cuándo se pueden ejecutar las tareas y que cambios sobre el estado del agente produce su ejecución, así como la motivación del agente para realizar estas tareas. El proceso de ejecución de tareas puede consumir y producir recursos, utilizar aplicaciones y modificar estados o entidades mentales.
  Así mismo, también se integra el concepto de *servicio*, que se asocia con la organización y con las tareas que los desempeñan.
- El **meta-modelo de interacción** define las *interacciones* del sistema y los A-Agentes que las llevan a cabo.
  Cada interacción concreta se modela en términos de unidades de interacción, cada una de las cuáles definen los A-Agentes que participan en la interacción (emisor y receptor/es), los objetivos que se persiguen y los protocolos que se utilizan.
- El **meta-modelo de entorno** define los *recursos* y las *aplicaciones* que utilizan los A-Agentes.

Por un lado, los recursos se modelan como objetos indispensables para la ejecución de las tareas. Por su parte, las aplicaciones son modeladas como un objeto del entorno que proporciona una funcionalidad concreta.
El meta-modelo también incluye las actuaciones y percepciones de los A-Agentes y el acceso a los servicios y a los diferentes elementos del entorno a través de puertos.

- El **meta-modelo de organización** describe los *flujos de trabajo* y la *coordinación* entre participantes, los *objetivos globales* y las *restricciones* en el comportamiento de los agentes.
  Para ello, se utiliza el concepto de *unidad organizativa*, que extiende el concepto general de A-Agente. Gracias a este nuevo concepto es posible describir agrupaciones entre los miembros de un mismo sistema, incluyendo su estructura interna, las posiciones o roles que juegan para una funcionalidad concreta y los objetivos asociados. Finalmente, también se describen como parte de la unidad organizativa los recursos y aplicaciones propios de la unidad, así como las normas de comportamiento específicas.
- El **meta-modelo normativo** en el que se detallan las *normas* de la organización, los objetivos normativos a seguir, así como las sanciones y recompensas que se producen como resultado de la ejecución de las acciones.

A partir de estos seis meta-modelos que describen los conceptos básicos que maneja la metodología, es posible modelar cualquier SMA basado en OV. Sin embargo, para facilitar esta tarea, la propia metodología propone una secuencia-guía para definir procedimentalmente el análisis y diseño del SMA. Esta guía metodológica parte de las guías de diseño de organizaciones humanas, las cuáles constan de varias fases en el análisis y diseño de la propia organización. Concretamente, GORMAS parte de la secuencia-guía básica de diseño organizativo propuesta por Moreno-Luzon *et al.* [Moreno-Luzon *et al.*, 2001] que integra los trabajos de Mintzberg [Mintzberg, 1993], en el que se definen 9 párametros para el diseño de organizaciones, y el trabajo de Galbraith [Galbraith, 1977], que propone cinco grupos de variables organizativas. Así, a partir de estos trabajos se divide el análisis, diseño del sistema y diseño de la dinámica de un SMA organizativo en las siguientes ocho fases:

- **Fase A. Misión.** Se realiza un análisis de la motivación que se persigue al definir la organización o sistema, es decir, el porqué de la existencia de dicha organización o para qué se crea. Así como los resultados que, en conjunto, se esperan conseguir. También se determina el entorno en el que existe el SMA, detallando los productos y/o servicios a ofrecer, cuáles son los grupos de interés y su localización.

- **Fase B. Tareas y procesos.** Se analizan con mayor detalle los servicios a ofrecer en el sistema, sus requisitos y los procesos que conllevan. Se detallan también las tareas y objetivos asociados a dichos servicios.
- **Fase C. Dimensiones organizativas.** Donde se analizan las dimensiones de la organización (departamentalización, especialización, sistema decisor, formalización, coordinación), que imponen ciertos requisitos sobre los tipos de trabajo, así como sobre la diversidad e interdependencia de las tareas a realizar.
- **Fase D. Estructura organizativa.** Se determina y selecciona la estructura organizativa más adecuada para la organización, en función de sus dimensiones. Se hace uso de modelos organizativos para especificar los roles, interacciones y normas relacionados con la propia estructura.
- **Fase E. Procesos de información-decisión.** Para cada servicio identificado, se detallan las interacciones (flujos de información y de adopción de decisiones) necesarias para llevar a cabo el servicio. Además, se definen los contratos de calidad de servicio a los que se comprometen los proveedores y consumidores cuando éste sea llevado a cabo.
- **Fase F. Dinamicidad del sistema abierto.** Se establece la funcionalidad ofrecida como sistema abierto, que incluye los servicios que se deben publicitar y las políticas de adquisición y liberación de roles. Además, se diseñan los agentes propios del sistema.
- **Fase G. Sistemas de medición, evaluación y control.** Se cuantifican o evalúan las tareas y actividades y se establecen mecanismos para determinar si los objetivos del sistema se cumplen. Así mismo, se revisan las normas de la organización para especificar quiénes se encargan de ellas y las supervisan.
- **Fase H. Sistemas de recompensas.** Se determina el sistema de incentivos, para recompensar a los miembros que avancen en dirección de los intereses de la organización. También se analizan los sistemas de sanción para aquellos miembros que no cumplan con las normas dadas.

Las fases A y B anteriormente detalladas se enmarcan en la etapa de análisis del sistema, las fases C y D se corresponden con el diseño y, finalmente, las fases E, F, G y H se corresponden con la etapa de diseño de la dinámica del sistema. El desarrollo de estas etapas/fases da como resultado un conjunto de vistas y documentos que describen claramente un SMA abierto basado en OV. El conjunto de vistas que se proponen en la metodología son las siguientes:

- La **vista estructural** que define cuáles son los componentes estáticos de la organización, es decir, todo aquello que es independiente de las entidades ejecutoras finales.
- La **vista funcional** que describe la misión o motivo de la existencia de la organización, así como la funcionalidad de cada unidad organizativa tanto a nivel externo, como interno.
- La **vista social** donde se describen las relaciones existentes entre los roles y los A-Agentes en términos de supervisión, monitorización e información.
- La **vista dinámica** se establecen los patrones de diseño de servicios propios de una unidad organizativa, que a su vez permite gestionar sus componentes estructurales y de ejecución.

A.2 ANÁLISIS DEL SISTEMA

La primera etapa de la metodología GORMAS está desarrollada con la definición de la misión del sistema y el análisis inicial de las tareas y procesos implicados. En esta etapa se llevan a cabo dos fases (A y B) que se presentan a continuación.

A.2.1 FASE A. MISIÓN

Durante esta fase se establece el porqué de la organización y cuál será su relación con el entorno. En este sentido, la misión es la pieza clave que guía el proceso de obtención de la estructura organizativa, que será posteriormente analizada en detalle en la Fase D (apartado A.3.2.). Durante esta primera fase de la metodología se presenta a nivel general qué prestaciones y servicios proporciona el sistema.

Así pues, se identificarán (i) los servicios y productos que proporciona, (ii) los grupos de interés, (iii) las condiciones del entorno, y finalmente (iv) se justificará la necesidad de desarrollar el sistema. Los resultados, documentos y modelos que se van a elaboran en esta fase son los siguientes:

- Documento A.1.- Misión organizativa.
- Documento A.2.- Grupos de interés.
- Documento A.3.- Condiciones de entorno.
- Vista funcional del modelo de organización

En primer lugar, se presenta en la Tabla 12 el *Documento A.1.* que presenta los productos y servicios que ofrece la organización, es decir, sus resultados. También se enumeran los grupos de interés y se describe el entorno en el que está situado el sistema. Finalmente se presenta la motivación y justificación a la hora de definir el sistema.

A.1.- Misión Organizativa
Nombre: +Cloud
Dominio: Servicios computacionales en entornos distribuidos
Resultados: • Producto: *Entorno de despliegue* o *Finalidad*: Entorno de despliegue de productos *software* orientados a Internet (aplicaciones web) desarrollados por terceros y orientados al cliente final (*End user*). o *A quién*: Usuarios *Cloud*, en especial la especialización en el rol *Desarrollador* de *software* o servicios. o *Dónde*: Solicitar el alta en el sistema como desarrollador a través del servicio *Contratación de capacidades computacionales*. • Producto: *Servicio de almacenamiento* o *Finalidad*: Servicios de persistencia en la nube (ficheros e información). o *A quién*: Usuarios *Cloud*, en especial la especialización en el rol *Desarrollador* de *software* o servicios. o *Dónde*: Solicitar el alta en el sistema como desarrollador a través del servicio *Contratación de capacidades computacionales*. • Servicio: *Configuración de servicios de software* o *Finalidad:* Alterar la configuración de los servicios desplegados en el sistema (características, disponibilidad, calidad del mismo, etc.). o *A quién:* Desarrolladores o gestores de *software* orientado a Internet (*Cloud Users*). o *Dónde:* Contactando con el sistema • Servicio: *Software de terceros* o *Finalidad:* Proporcionar a los usuarios finales el *software* desplegado en el sistema por terceros, es decir, el *software* que despliegan en el sistema el rol *Cloud User* utilizando para ello los productos y servicios que ofrece el sistema. o *A quién:* Usuarios finales (*End users*) o *Dónde:* Contactando con el sistema o el *software* desplegado. • Servicio: *Contratación de capacidades computacionales* o *Finalidad:* Establecer un acuerdo de uso acerca de los productos proporcionados por el sistema (*Entorno de despliegue* y *Servicio de almacenamiento*). o *A quién:* Cloud Users, principalmente la especialización en el rol *Manager* de éste. o *Dónde:* Contactando con el sistema. • Servicio: *Control de la infraestructura* o *Finalidad:* Supervisar y configurar la infraestructura *hardware* real y virtual subyacente del sistema o *A quién:* Administradores del sistema o *Dónde:* Contactando con el sistema.

Grupos de interés: • *Cloud User:* Usuario que contrata las capacidades del entorno CC para proporcionar sus servicios al usuario final. • *End User:* Usuario final que contrata los servicios computacionales proporcionados por el entorno CC. • *Cloud Admin:* Usuario que controla el correcto funcionamiento del sistema y configura las características de la capa subyacente.
Tipo de Entorno: • Virtual y distribuido.
Justificación: • *Sistemas similares:* Plataformas CC de las grandes compañías tecnológicas en Internet (Amazon, Google, Microsoft, etc.) y plataformas CC de *software* abierto desarrolladas por la comunidad open source o financiadas por grandes compañías (OpenNebula, Eucalyptus, OpenStack, etc.). • *Ventajas:* Automonitorización dinámica, negociación y adaptación automática. • *Desventajas:* mayor complejidad al ser un sistema adaptivo automático y menor infraestructura subyacente. • *Singularidades:* Diseño mediante un SMA organizativo y la capacidad de reorganización y adaptación automática.

Tabla 12.- Documento A.1.- Misión Organizativa

A partir del Documento A1 presentado en la Tabla 12, el cuál se ha específicado según la metodología GORMAS, se instancia la vista funcional (misión) del modelo de organización. La misión de la organización es maximizar la calidad de los servicios, minimizando el coste asociado en términos de recursos computacionales, pero siempre con el objetivo de no violar los acuerdos SLA que se hayan alcanzado con los usuarios del sistema +Cloud en cuanto al uso y calidad de los mismos.

El sistema ofrece dos productos principales, un entorno de despliegue *software* y una capa de persistencia de ficheros e información, ambos productos servirán de base para el despliegue de aplicaciones desarrolladas por terceros (*Cloud Users*). Para este tipo de usuarios, el sistema ofrece dos servicios específicos, por un lado, un sistema para la gestión y control de las características de configuración del *software* desplegado y por otro lado, un servicio de contratación de los productos de la plataforma, es decir, los productos de despliegue *software* y almacenamiento. Cabe destacar que el sistema ofrece como servicio a Usuarios finales (*End Users*), el *software* que se ha desplegado en el sistema por terceros. Por lo tanto, el sistema requiere de este *software* como razón de su existencia, pero al mismo tiempo, este *software* orientado a Internet es ofertado a un nuevo grupo de interés, que son

los consumidores finales. Finalmente, también existe un tercer grupo de interés, que es el administrador del sistema (*Cloud Admin*), cuyo objetivo consiste en controlar, configurar y supervisar que el sistema funciona correctamente, así como que la plataforma está utilizando eficientemente los recursos computacionales disponibles.

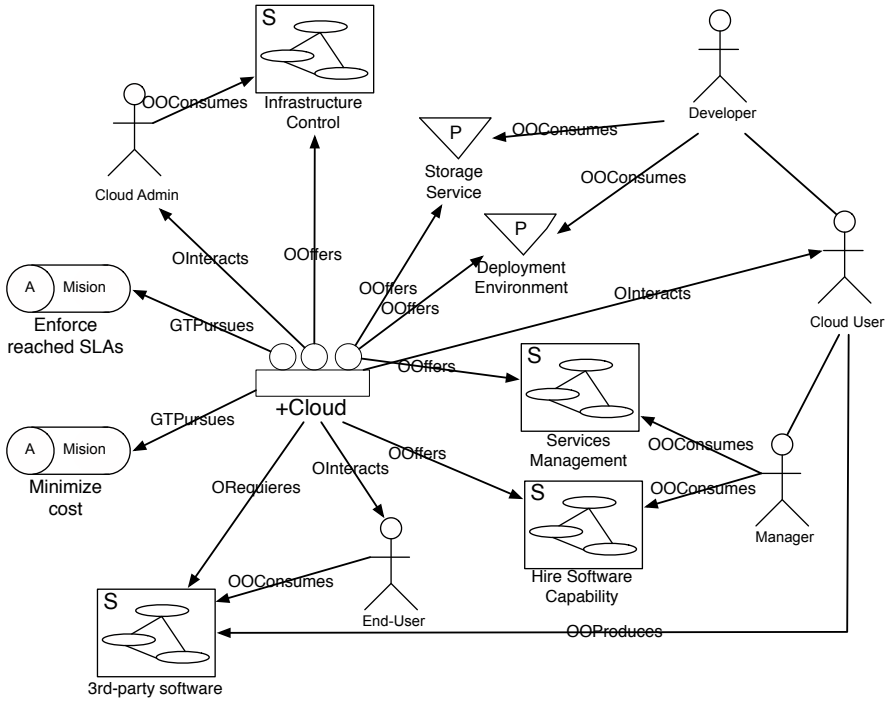

Figura 60.- Vista funcional (misión) del modelo de organización de +Cloud

En la Tabla 13 recoge el *Documento A.2.* que presenta en detalle las características de los grupos de interés del sistema, sus objetivos, requisitos, utilidad, frecuencia de uso del sistema, los beneficios que obtienen a nivel individual y si se puede influenciar sus intereses. Es necesario destacar que un grupo de interés puede ser primario o secundario en función de si resultan imprescidibles para el correcto funcionamiento sistema.

A.2.- Grupos de interés			
Grupo / Caract.	**End User (Cliente final)**	**Cloud User**	**Cloud Admin (Administrador)**
Beneficiario	Secundario	Primario	Secundario
Tipo	Cliente	Cliente/Provedor	Regulador
Objetivos	Satisfacer sus necesidades de *software*	Disponer de servicios computacionales.	Verificar que el sistema funciona correctamente y tareas de control.
Requiere	Servicios *software* de naturaleza variable	Servicio de despliegue y almacenamiento de naturaleza variable, pero predecible	--
Proporciona	--	*Software*	Supervisión y control el sistema
Frecuencia	Frecuente	Ocasional	Ocasional y establecido
Beneficios	Uso del *software* solicitado, intangible	Servicios computacionales, tangibles.	--
Poder de decisión	No	Si (contratación de servicios)	Si (cambios en la configuración)
Influencia sobre sus intereses	No	Si (a través del servicio de contratación)	No
Aportación	--	Servicios *software*	Control y supervisión por expertos.

Tabla 13.- Documento A.2.- Grupos de Interés

La Tabla 14 se corresponde con el *Documento A.3.* que incluye el análisis de las condiciones del entorno, es decir, todos los factores externos que condicionen al sistema. En este sentido, es necesario analizar la tasa de cambio, complejidad, incertidumbre, receptividad y la diversidad.

A.3- Condiciones de entorno		
Condición	**Valor**	**Justificación**
Tasa de cambio	Dinámico	Ya que constantemente pueden cambiar los acuerdos con los usuarios y el *software* desplegado en el sistema +Cloud.
Complejidad	Complejo	Existe un gran número de componentes en el sistema con múltiples relaciones entre ellos.
Incertidumbre	Alto	Es un entorno dinámico y complejo.
Receptividad	Hostil	Ya que el *software* requerido es desarrollado por terceros y podría contener errores.
Diversidad	Uniforme	Ya que los servicios que se ofertan y se proporcionan son homogéneos (servicios computacionales).

Tabla 14.- Documento A.3.- Condiciones de entorno

A.2.2 FASE B. TAREAS Y PROCESOS

La segunda fase de la guía metodológica GORMAS se centra en el análisis de la tecnología que requiere el sistema, atendiendo a las características principales de los productos y servicios, así como a las tareas y procesos implicados.

Gracias a este análisis es posible (i) identificar los productos y servicios que se ofertan, (ii) identificar las tareas asociadas a estos productos y servicios, (iii) asociar objetivos a la producción de esos productos y servicios, (iv) identificar los recursos y aplicaciones a los que se accede para ofrecer esta funcionalidad, y, finalmente, (v) definir los roles que se asocian a los grupos de interés identificados en la Tabla 13. Los resultados, documentos y modelos que se van a elaboran en esta fase son los siguientes:

- Documento B.1.- Tecnología esencial.
- Documento B.2.- Tecnología de la unidad de trabajo, tanto para los servicios, como para los productos.
- Vista estructural del modelo de organización.
- Vista funcional del modelo de organización (funcionalidad interna y externa).
- Modelo de actividad, para cada servicio especificado en la anterior.
- Modelo de entorno.
- Vista funcional actualizada del modelo de organización

La tecnología se encarga de los procesos de transformación de los servicios computacionales básicos del sistema en los productos o servicios finales que se

le ofrecen a los usuarios. En este sentido, la tecnología se observa desde tres puntos de vista a considerar: (i) la tecnología esencial o tecnología a nivel organizativo, (ii) la tecnología departamental o de unidades de trabajo y, finalmente, (iii) la complejidad de las diversas tareas organizativas.

En primer lugar, en la Tabla 15 se presenta el *Documento B.1.* que se corresponde con la tecnología esencial, es decir, aquella que hace referencia al modo en el que se deben manufacturar los productos y servicios que oferta el sistema, en función de la demanda y las necesidades de los clientes. Según el análisis de la tecnología, los productos y servicios que ofrece el sistema están disponibles sin tener en cuanta a los clientes finales. No obstante, a la hora de su comercialización de los productos (despliegue y persistencia), éstos se ofrecen en paquetes acotados a las necesidades de los clientes, en este caso el rol *Cloud User*.

B.1- Tecnología esencial
Tecnología de producto: *producción continua y en pequeños lotes.* • Producción continua: Los productos están disponibles para los grupos interés sin un fin claro. • Producción en pequeños lotes: Los productos se ofrecen de acuerdo a los requisitos acordados de forma individual con cada cliente (*Cloud User*).
Tecnología de servicio: *tecnología intensiva* • Tecnología intensiva: Los servicios dependen de los requisitos del cliente.

Tabla 15.- Documento B.1.- Tecnología esencial

En la Figura 61 se presenta la vista estructural del modelo de organización, en el que aparecen los A-Agentes que representan a las entidades externas que interactúan con el sistema (*Cloud User*, *Cloud Admin* y *End User*). Cabe destacar que el A-Agente *Cloud User* se especializa en desarrollador y *manager*, dónde este último rol a su vez puede ser una entidad humana o automática. Posteriormente en la Figura 62 se presenta la vista de la funcionalidad interna del modelo de organización, donde se muestra como cada rol externo está asociado a un rol en el sistema: *End User*, *Cloud User* y *Global Supervisor*, respectivamente.

Continuando con la vista estructural del modelo de organización, en ella también se puede observar como la unidad organizativa +Cloud dispone de:

- Tres recursos propios, que son los repositorios de recursos de la infraestructura subyacente, de acuerdos SLA y de histórico de uso de las capacidades computacionales ofertadas.
- Dos aplicaciones que actúan con interfaces con los A-Agentes externos del sistema, es decir, la aplicación que permite monitorizar la infraestructura subyacente del entorno CC, específicamente diseñada

para el rol *Cloud Admin* y el escritorio web que dispone de las aplicaciones específicas para el rol *Cloud User*. Los usuarios finales acceden directamente a las aplicaciones de terceros que están desplegadas en el sistema.

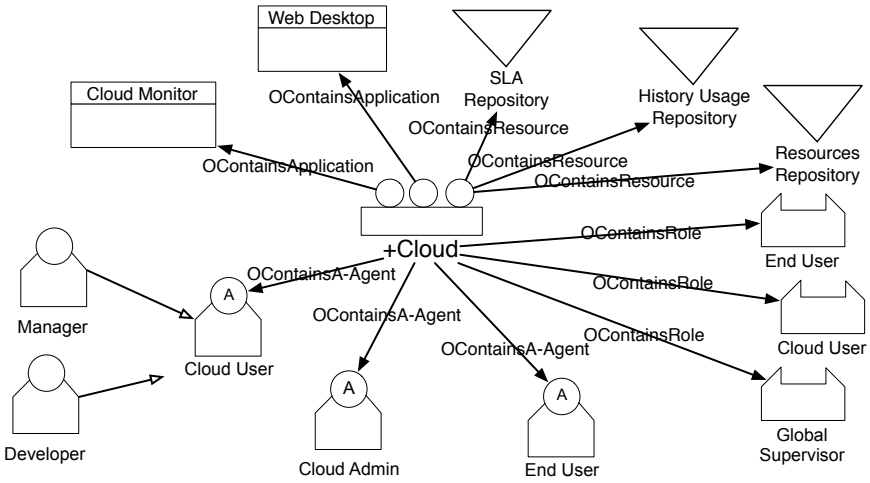

Figura 61.- Vista estructural del modelo de organización de +Cloud

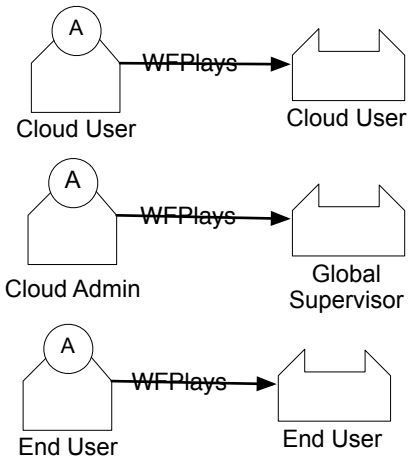

Figura 62.- Vista funcional (funcionalidad interna) del modelo de organización de +Cloud

La vista funcional externa del modelo de organización, Figura 63, presenta los servicios que el sistema ofrece y los agentes externos (en términos de roles) que los proporcionan o consumen. Cabe destacar, como ya se mencionó, que el servicio de *software* de terceros, es un tipo de servicio que se ofrece al consumir final, pero que es requerido por el sistema y ofrecido al propio sistema +Cloud por el rol *Cloud User* que despliega sus aplicaciones orientadas a Internet en la plataforma.

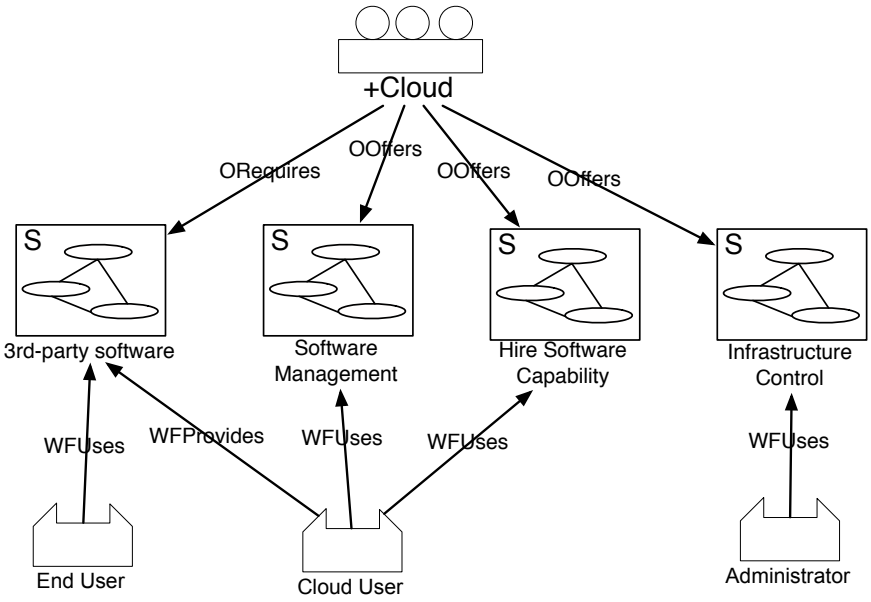

Figura 63.- Vista funcional (funcionalidad externa) del modelo de organización de +Cloud

A nivel interno cada uno de estos servicios, presentados Figura 63, se descomponen en un conjunto de tareas (A-Tareas) que determinan su funcionalidad. Por cada uno de los servicios y tareas es necesario determinar cuáles son los recursos requeridos y los productos a obtener, que se presentan a través del *Documento B.2.*, que recoge la tecnología de la unidad de trabajo y del modelo de actividad, que a su vez incluye el perfil del servicio y relación entre las A-tareas que los forman.

En este sentido, se presentan estos artefactos para los principales servicios que ofrece el sistema, es decir, el servicio de gestión de infraestructura y el servicio de contratación de capacidades (servicios computacionales). Para los otros dos servicios que ofrece la plataforma el modelado de los artefactos que los definen es similar.

Así pues, a continuación se presenta la tecnología de la unidad de trabajo para los servicios de gestión de la infraestructura (Tabla 16) y la contratación de servicios (Tabla 17).

B.2.- Tecnología de la unidad de trabajo	
Servicio	Control de infrastructura (*Infrastructure Control*)
Descripción	Permite gestionar (crear, añadir, modificar) los recursos computacionales del sistema, así como la asignación de estos a cada uno los servicios proporcionandos, ya sean servicios intrínsecos al sistema, o servicios desplegados por terceros en el sistema. Finalmente, también permite supervisar la correcta asignación de recursos y el control del sistema en su conjunto por el rol externo *Cloud admin*.
Condiciones	
Contexto	El servicio es ejecutado de forma continuada para controlar y gestionar la infraestructura subyacente, ya sea de forma supervisa o no supervisada.
Excepciones	El servicio no funciona correctamente cuando el entorno subyacente falla, ésto es el susbsistema de gestión de *hardware* real o virtual, lo que por ende implica que la gestión de recursos no está disponible. No obstante, se tiene que asegurar que el resto de servicios funcionen pese al fallo parcial del sistema. En este sentido, tan sólo fallaría la adaptación automática del sistema.
Consumidor	Administrador *Cloud (Cloud Admin)*
Objetivo	Supervisar el reparto de los recursos manualmente, o modificar el reparto realizado por el sistema automático.
Beneficio	Garantizar el correcto funcionamiento de los sistemas y aseguramiento de las misiones de la organización.
Perfil del servicio	
Entradas	*Propiedades máquinas físicas + propiedades máquinas virtuales + Propiedades entorno persistencia + propiedades de servicio + distribución de recursos* • *Propiedades máquinas físicas*. Características de acceso a las máquinas físicas. • *Propiedades máquinas virtuales.* Características de acceso a las máquinas virtuales. • *Propiedades entorno persistencia.* Características del entorno de persistencia. • *Propiedades de servicio.* Características de los servicios *software* proporcionados. • *Distribución de recursos.* Relación de recursos computacionales (máquinas virtuales o área de persistencia) y su asociación a los recursos ofertados.

Precondiciones	Deben existir recursos físicos en el sistema y un entorno de virtualización que permita la distribución dinámica de recursos. Así como la existencia de un área de almacenamento distribuida en red, que pueda ser modificada dinámicamente.
Salidas	Configuración de recursos en función de la demanda existente, minimizando el coste.
Postcondiciones	La configuración resultante tiene que satisfacer los acuerdos SLA acordados.
Funcionalidad	
Tareas	• *Gestión recursos físicos.* Permite la gestión de la configuración de las máquinas físicas, así como añadir y eliminar máquinas existentes. • *Gestión recursos virtuales.* Permite la gestión de la configuración de las máquinas virtuales, así como añadir y eliminar máquinas existentes. • *Distribución de recursos.* Permite cambiar la distribución de los recursos ofertados por las máquinas físicas entre las diferentes máquinas virtuales existentes. • *Gestión almacenameinto.* Permite el control sobre las diferentes áreas de almacenamiento en red del sistema. • *Monitorización de recursos.* Permite monitorizar los recursos físicos de la infraestructura *hardware* o virtual subyacente.
Recursos	Repositorio de recursos computacionales. Repositorio acuerdos SLA. Repositorio de histórico de uso.
Proveedor	+Cloud
Productos	Servicio de despliegue *software*. Servicio de almacenamiento.

Tabla 16.- Documento B.2.- Tecnología de Unidad de Trabajo (Control de infrastructura)

A partir de la tecnología de la unidad de trabajo que se acaba de presentar, en la Figura 64 se puede observar la descomposición del servicio en A-Tareas. Posteriormente, en la Figura 64 se presenta la interdependencia entre estas A-Tareas, identificando las entradas y salidas de cada una de ellas.

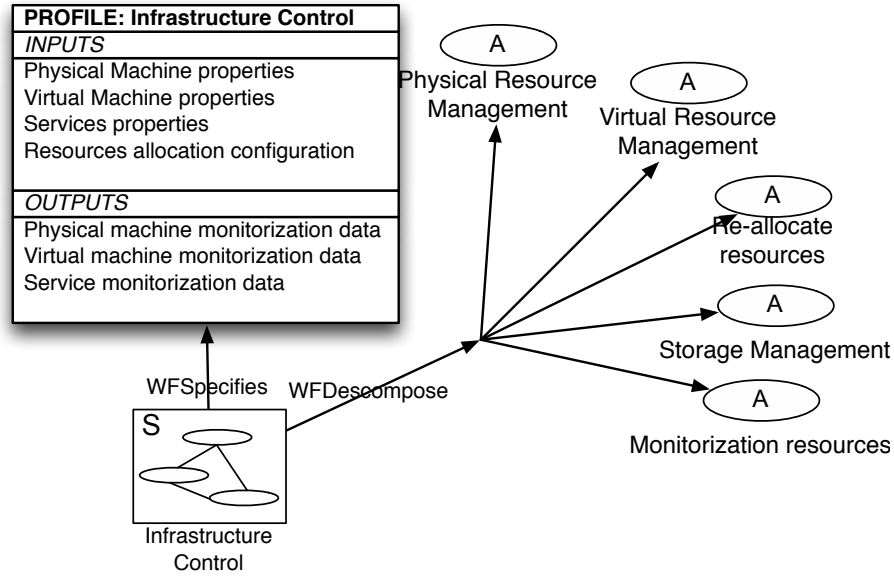

Figura 64.- Modelo de actividad (Perfil) del servicio *Control de infrastructura*

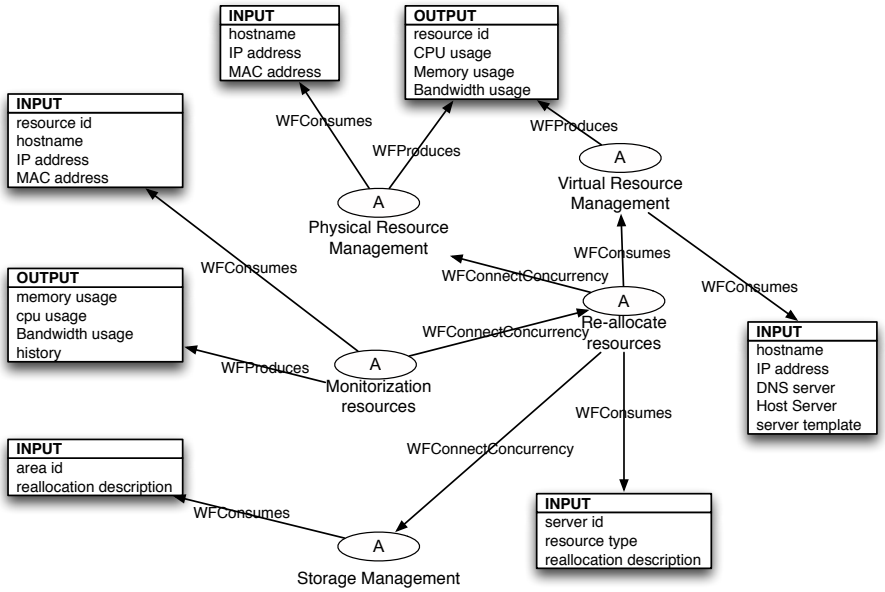

Figura 65.- Modelo de actividad (Relaciones entre las A-Tareas) del servicio *Control de infrastructura*

Del mismo modo a como se ha descrito el servicio anterior, en la Tabla 17 se presenta el Documento B.2. para el servicio de contratación, así como su modelo de actividad en la Figura 66 y en la Figura 67.

B.2- Tecnología de la Unidad de Trabajo	
Servicio:	Contratación de servicios (*Hire software capability*)
Descripción	Permite la contratación de los productos ofertados por el sistema, es decir los servicios computacionales de almacenamiento y despliegue de *software* orientado a Internet. Así, el rol *Cloud User*, puede a su vez ofertar a terceros (*End User*) sus servicios a través de la plataforma
Consumidor	Usuario Cloud *(Cloud User)*
Objetivo	Obtener los recursos computacionales necesarios al menor coste posible.
Beneficio	Satisfacer sus necesidades computacionales para el despliegue de aplicaciones.
Perfil del servicio	
Entradas	*Recursos computacionales requeridos* • *Recursos computacionales*. Recursos requeridos por el usuario cloud
Precondiciones	Deben existir recursos suficientes en el entorno para satisfacer las demandas del consumidor, sin perjudicar el resto de acuerdos previamente alcanzados.
Salidas	Condiciones del acuerdo alcanzado y recursos computacionales requeridos.
Postcondiciones	Los acuerdos alcanzados deben ser satisfechos durante el tiempo que dure el acuerdo.
Funcionalidad	
Tareas	• *Contratación servicio.* Requerimiento de servicio de despliegue. • *Contratación almacenamiento.* Requerimiento de servicio de almacenamiento. • *Negociación.* Proceso de modificación de la oferta y el servicio acordado. • *Establecimiento de acuerdo.* Estabilización del acuerdo según las condiciones acordadas y reserva de recursos computacionales asociados.
Recursos	Repositorio de acuerdos SLA. Repositorio de recursos computacionales.
Proveedor	+Cloud

Productos	Servicio de almacenamiento. Servicio de despliegue *software*.

Tabla 17.- Documento B.2.- Tecnología de Unidad de Trabajo (Configuración de *software*)

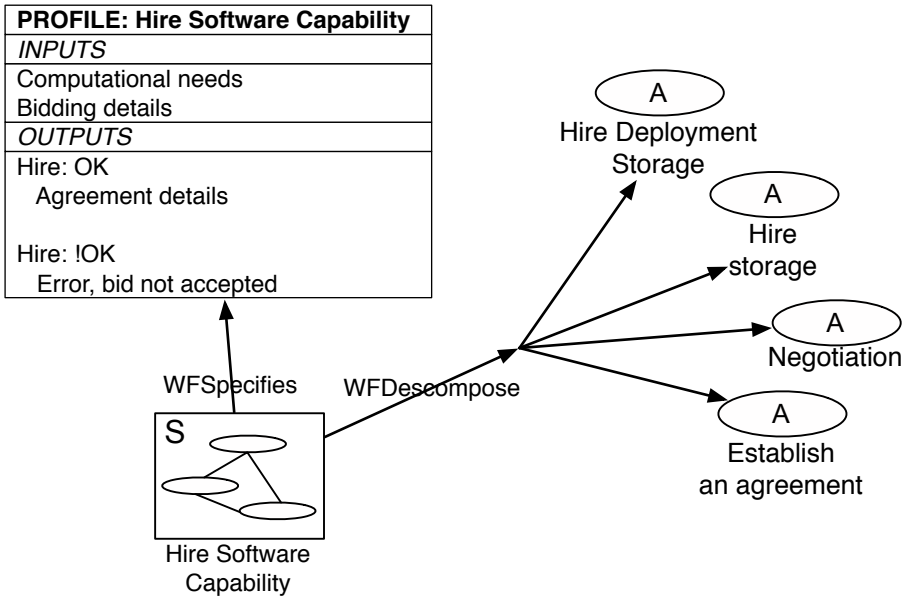

Figura 66.- Modelo de actividad (Perfil) del servicio *Contratación de capacidades software*

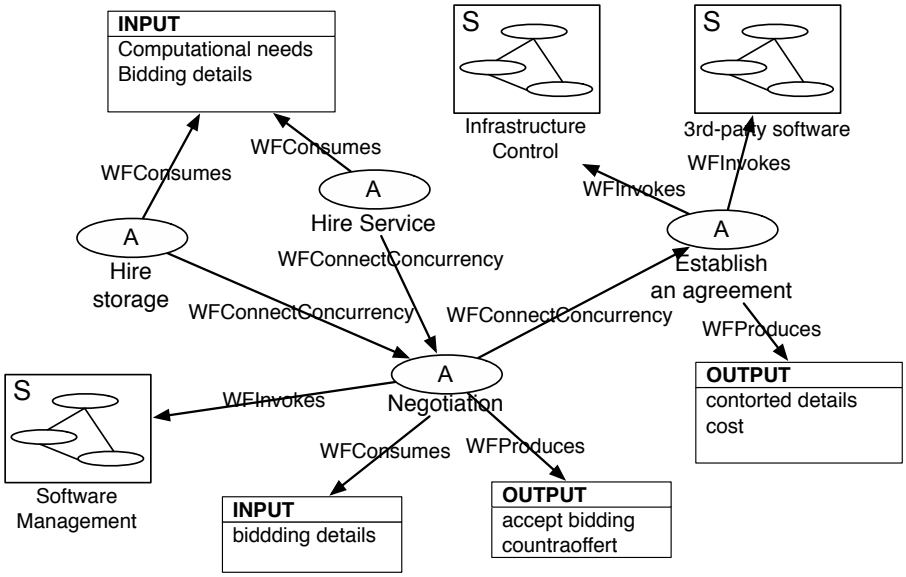

Figura 67.- Modelo de actividad (Relaciones entre las A-Tareas) del servicio *Contratación de capacidades software*

Siguiendo la metodología, a continuación en la Figura 68 se presenta el modelo de entorno donde se recoge la interacción del sistema con el ambiente que lo rodea, así como los roles externos con los que interacciona. Todos estos elementos, en parte ya se presentaron en la vista estructural del modelo de organización (Figura 61). En este sentido, por un lado, se presentan los recursos disponibles que son el almacen de acuerdos de calidad de servicio, el almacen de recursos computacionales (físicos o virtuales) y el histórico de uso de los servicios. Por otro lado, en cuanto a las interfaces, se tiene el escritorio web accesible para el rol *Cloud User*, así como el monitor CC disponible para los *Cloud Admin*.

La principal diferencia con el modelo de organización, en esta fase del diseño del sistema, radica en que el modelo de entorno muestra la relación (*EPerceive*) que existe entre los roles externos y aplicaciones que proporciona el sistema.

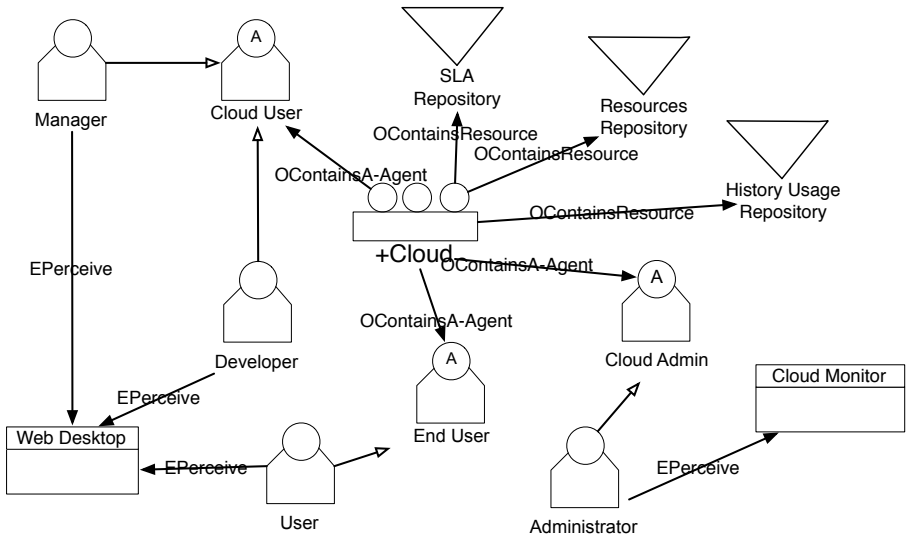

Figura 68.- Modelo de entorno

Para finalizar esta fase y la etapa análisis, siguiendo con el marco metodológico descrito por GORMAS, en la Figura 69 se presenta la relación entre servicios y los objetivos funcionales del modelo de actividad. Los objetivos funcionales representan las acciones específicas de las divisiones, departamentos o unidades de la organización, así como los resultados que se espera que consigan. Así pues en esta vista del modelo de actividad, cada una de las misiones del sistema que han sido especificadas en la vista estructural del modelo de organización (Figura 60) y que son objetivos de alto nivel, se descomponen en objetivos funcionales que son satisfechos por los servicios que ofrece y requiere la plataforma.

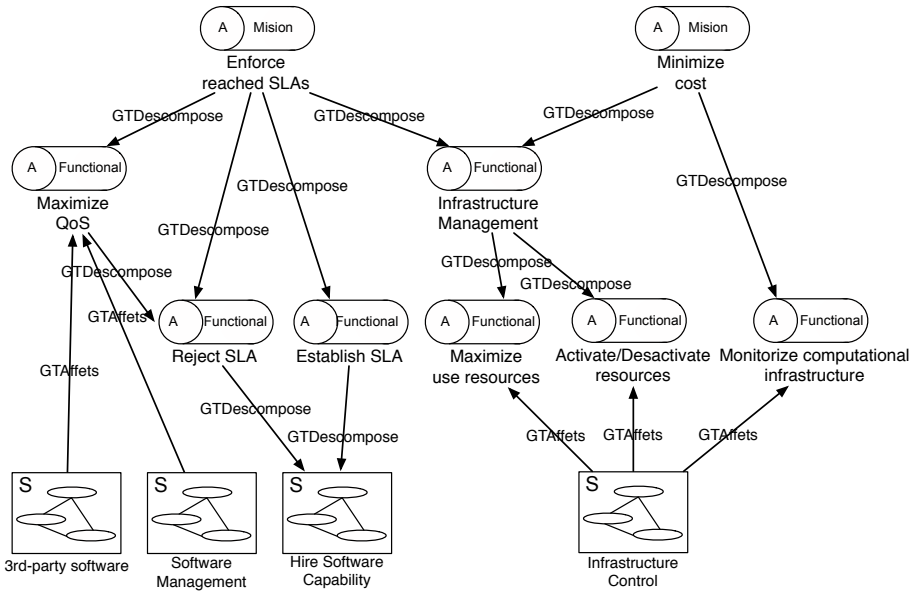

Figura 69.- Modelo de actividad (Relación entre servicios y objetivos funcionales)

A.3 DISEÑO DEL SISTEMA

Una vez que se ha completado la etapa de análisis del sistema, que incluye las Fases A y B de la metodología, en las dos siguientes se analizarán las dimensiones de la organización a través de la definición de las características intrínsecas (Fase C), para posteriormente definir un patrón estructural que adoptará la organización (Fase D).

A.3.1 FASE C. DIMENSIONES ORGANIZATIVAS

En la Fase C se analizan las dimensiones organizativas del sistema, lo que por consiguiente impone ciertos requisitos sobre los tipos de trabajos a realizar. En este sentido, las dimensiones que se proponen en esta fase están definidas en el trabajo de Mintzberg [Mintzberg, 1993] que son la departamentalización, especialización, sistema decisor, normalización y los mecanismo de coordinación. La especificación de estas características permite modelar las características internas de la organización.

Gracias a lo cuál es posible (i) asignar tareas e (ii) identificar restricciones en cuanto a la formalización de la plataforma y los mecanismos de coordinación necesarios. Los resultados, documentos y modelos que se van a elaboran en esta fase son los siguientes:

- Documento C.- Dimensiones organizativas.
- Vista estructural actualizada del modelo de organización.
- Vista funcional actualizada del modelo organización.
- Modelo de actividad actualizado por cada servicio complejo identificado.

Según la metodología las vistas estructural y funcional actualizadas son opcionales en esta fase del desarrollo. No obstante, para una mejor comprensión del diseño y del propio sistema +Cloud en su conjunto se presentarán las vistas estructural y funcional del modelo de organización actualizado. De esta última vista sólo se presentará la funcionalidad externa ya que la vista estructural en esta fase del desarrollo recoge la relación *WFPlays* entre A-Agentes y roles del sistema, que es el principal cometido de la vista de funcionalidad interna.

La fase C, en primer lugar comienza con la especificación del *Documento C*, donde se detallan las dimensiones anteriormente presentadas [Mintzberg, 1993]:

- Se comienza por la asignación de tareas, en la dimensión denominada *departamentalización*, en la que se contempla la departamentalización funcional si la fragmentación se realiza en base a los conocimientos y habilidades de los especialistas que realizan las tareas; o en divisiones si

en cambio se tiene en cuenta las características del mercado y los potenciales clientes. Posteriormente se analizará la *especialización* o división del trabajo, que puede ser horizontal si se agrupan las tareas en función del rol; o vertical si por el contrario depende del ámbito de control sobre la información y la actualización. Finalmente, la asignación de tareas también incluye el análisis de los procesos de toma de decisión, que puede ser centralizado o descentralizado en función del tamaño y complejidad del sistema.

- Posteriormente, a la hora de identificar restricciones se analizan, por un lado, los *mecanismos de coordinación* contemplándose los casos de adaptación mutua (negociación) y la supervisión directa (jerarquía). Y por otro lado, también es necesario analizar la *normalización* que permite determinar el grado de estandarización del sistema, distinguiendo normalización de tareas que implica la especialización de tareas de forma concreta; o normalización de resultados que define las tareas por los objetivos a cumplir, dejando libertad a los roles para alcanzarlos en función de sus conocimientos y habilidades.

La Tabla 18 que se presenta a continuación detalla cada una de las dimensiones organizativas del sistema.

C. Dimensiones organizativas
Departamentalización: *Funcional* • Se agrupan las tareas en función de los conocimientos y habilidades de quién las realiza (control de servicios, gestión de infraestructura, etc.). Esto es así, debido al grado de complejidad y especialización de las tareas a realizar, en este sentido se tiene en cuenta que la producción de los resultados es intensiva.
Especialización + Coordinación: *Especialización horizontal con ampliación vertical. Descentralización* • Los roles que realizan las diferentes tareas están altamente especializados debido a la complejidad del entorno con el que se interacciona para satisfacer la funcionalidad que se busca. Este tipo de organización se asocia a la especialización horizontal. Además tienen el control sobre el mecanismo a utilizar para la consecución de los objetivos perseguidos. Sin embargo, también se desea que los roles tengan flexibilidad y capacidad de decisión sobre las tareas a realizar, lo que se asocia a un modelo de organización con ampliación vertical. • La coordinación se realiza de forma descentralizada por lo que se hace necesaria la negociación entre los roles para llevar a cabo la toma de decisiones.

Coordinación y Formalización. *Adaptación mutua y normalización de resultados* • Existe adaptación mutua ya que el modo de realización de las tareas es flexible y requiere coordinación entre los diferentes roles implicados. • Existe normalización de resultados ya que importa la calidad de los resultados obtenidos, teniendo en cuenta que se está trabajando en un sistema de alta disponibilidad. El sistema es libre de tomar las acciones que considere oportuno para llegar a alcanzar los resultados previstos.

Tabla 18.- Documento C.- Dimensiones organizativas

Posteriormente, una vez que se han presentado las dimensiones organizativas según el modelo organizativo de Mintzberg [Mintzberg, 1993], analizando de este modo la departamentalización y los mecanismos de coordinación existentes dentro del sistema, a continuación la unidad organizativa principal se segmentará en diferentes unidades organizativas especializadas. Así pues, la Figura 70 y, posteriormente, la Figura 71 muestran la actualización del modelo de organización teniendo en cuenta la nueva estructura departamental del sistema +Cloud. La Figura 70 muestra la departamentalización de la unidad organizativa principal tal y como sigue:

- *SLA Negotiation.* Es la unidad organizativa encargada de agrupar las tareas de establecimiento de acuerdos y la negociación de los mismos con el rol externo *Cloud User*. Este proceso tiene que estar totalmente automatizado, así mismo se debe determinar si el sistema tiene capacidad para asumir nuevos acuerdos sin violar los existentes.
- *Service Management.* Es la unidad que tiene asignadas las tareas de monitorización y supervisión de productos que se ofertan y si estos están siendo ofrecidos atendiendo a los acuerdos de calidad estipulados en el acuerdo de uso específico con cada usuario. Las tareas de esta unidad organizativa, al igual que las anteriores también tienen que estar automatizadas.
- *Infraestructure Management.* Es la unidad que controla la infraestructura *hardware* subyacente, es decir, los recursos computacionales (reales o virtuales) del sistema mediante la gestión y monitorización de los mismos. Aunque las tareas también deben estar automatizadas, esta unidad organizativa puede estar supervisada por el A-Agente *Cloud Admin*, que se asocia al rol interno *Global Supervisor*, como se detallaba en la Figura 63 correspondiente a la vista funcional (funcionalidad interna) del modelo de organización

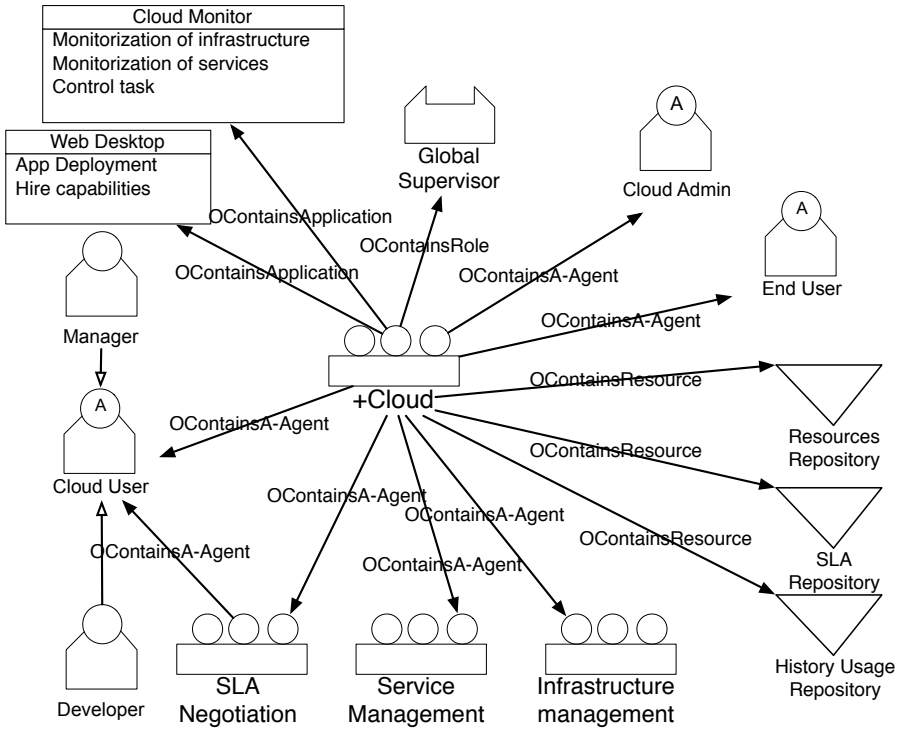

Figura 70.- Vista estructural del modelo de organización actualizado

Posteriormente, a partir de estas tres nuevas unidades organizativas, también se descomponen los servicios identificados durante el análisis del sistema en la etapa anterior. En este sentido, los servicios de gestión de servicios e infraestructura se subdividen cada uno de ellos en dos servicios con funcionalidades similares, uno para la monitorización y otro el control y la toma de decisiones. Así pues, en la Figura 71 se presenta la vista funcional actualizada (funcionalidad externa) del modelo de organización.

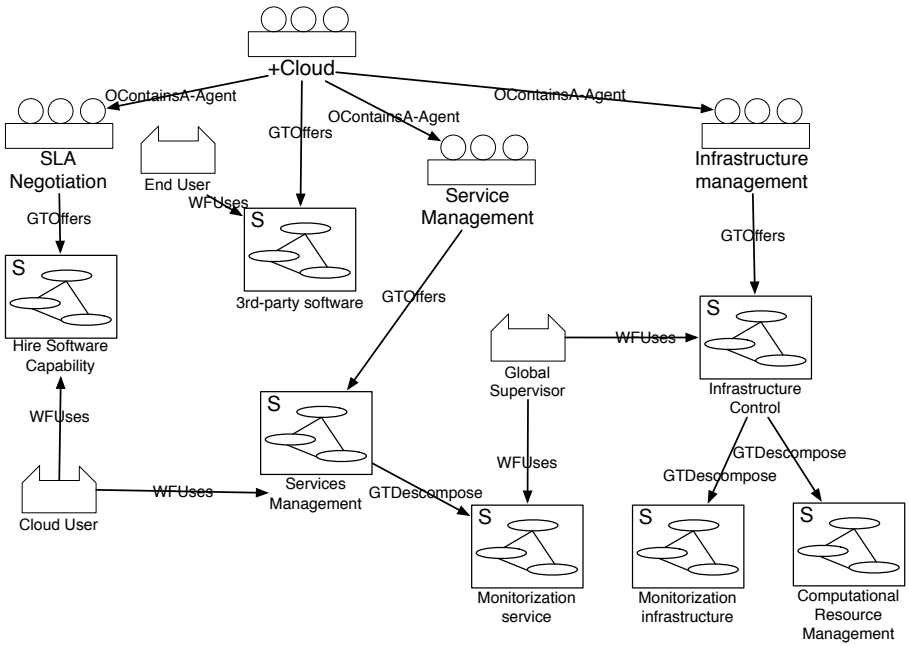

Figura 71.- Vista funcional (funcionalidad externa) del modelo de organización actualizado. Agrupación funcional

A.3.2 FASE D. ESTRUCTURA ORGANIZATIVA

Partiendo de los resultados obtenidos en la fase anterior, es decir, las dimensiones de la organización [Mintzberg, 1993], las restricciones y los mecanismos de coordinación entre cada una de las unidades organizativas. A lo largo de esta fase se procede a identificar la estructura organizativa que mejor conviene aplicar en el sistema.

Por lo tanto, en esta fase se procede a determinar (i) la estructura del sistema y la (ii) adaptación del patrón de diseño a la estructura del problema a tratar. Los resultados, documentos y modelos que se van elaboran en esta fase son los siguientes:

- Vista estructural actualizada del modelo de organización.

Para seleccionar la estructura más adecuada de la organización, en la metodología se propone el uso del árbol de decisión que se presenta en la Figura 72. Este árbol guía al diseñador a través de las dimensiones de departamentalización, especialización y coordinación que se han especificado en la fase anterior.

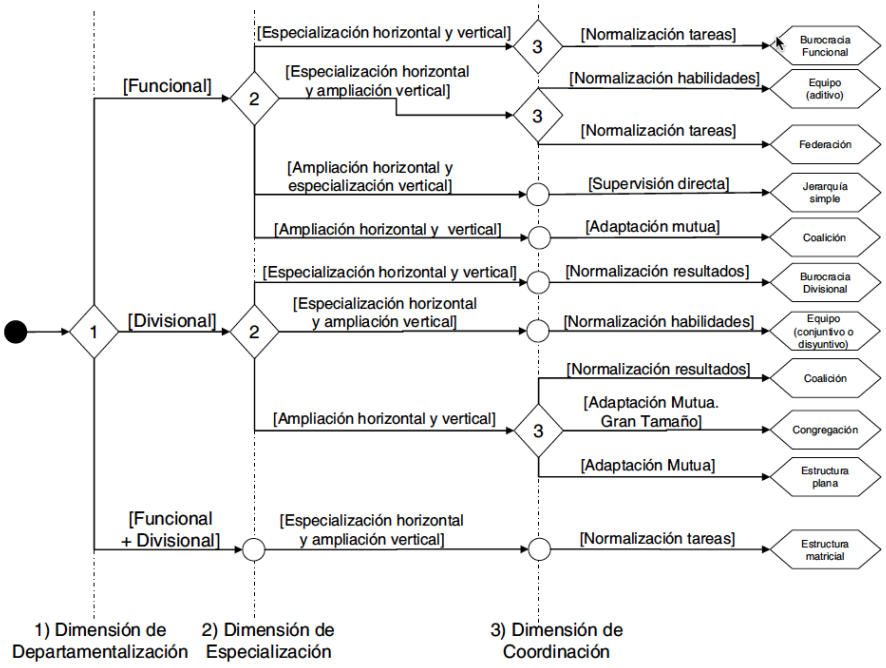

Figura 72.- Árbol de decisión de la estructura organizativa [Argente, 2008]

Teniendo en cuenta los valores asignados a las dimensiones organizativas previamente (funcional, especialización horizontal con amplicación vertical, adaptación mutua y normalización de resultados), si se siguen las diferentes ramas del árbol de decisión no se obtiene una estructura ideal. No obstante, haciendo un análisis detallado, en el que se tiene en cuenta la opción que más se aproxima según el árbol, el tamaño del sistema y la coordiandión existente entre tareas, se selecciona la estructura de *coalización* como la más adecuada para modelar el sistema. Las diferentes entidades que forman parte de una estructura de tipo coalización forman grupos en función de la similitud en cuanto a la funcionalidad, coordinándose entre sí para ofrecer una funcionalidad global más compleja y elaborada. Dentro de esta estructura los agentes individuales son responsables de la gestión de sus propias acciones, por lo que sólo es posible aplicar sanciones o recompensas en base a los resultados de dichas acciones.

A partir de esta estructura organizativa que se acaba de definir, a continuación se presenta la actualización del modelo de organización. En primer lugar, en la Figura 73 se presenta la vista estructural, que con respecto a la versión anterior (Figura 61) los cambios más significativos que se aprecian es la inclusión de cardinalidades de cada uno de los A-Agente que conforman el sistema. Así, en el sistema propuesto cada una de las tres subunidades

organizativas será única, mientras que los diferentes roles y usuarios externos tendrán cardinalidad múltiple.

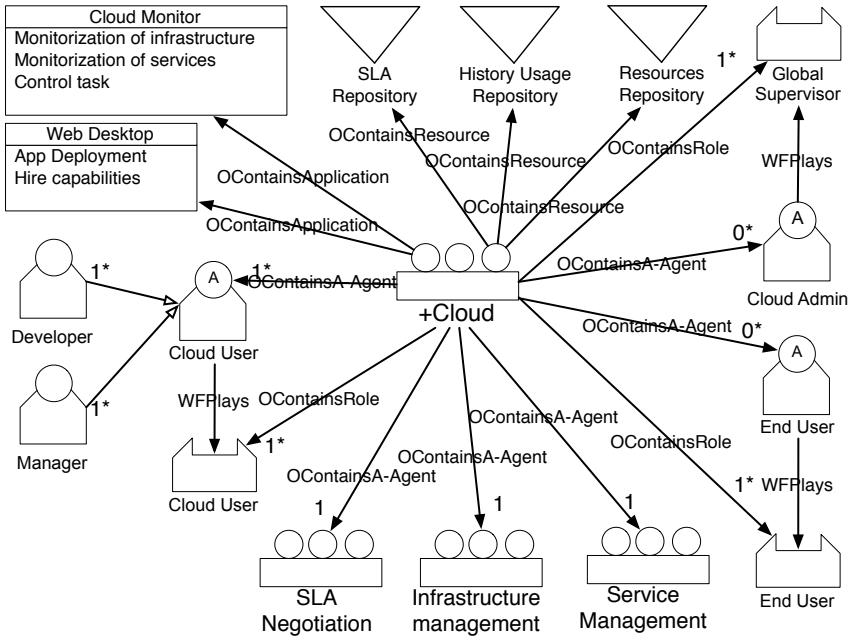

Figura 73.- Vista estructural del modelo de organización actualizado

En la vista funcional (Figura 74) del modelo de organización, una vez que se ha adaptado el patrón coalición que se ha elegido, la unidad organizativa +Cloud tiene tantas unidades como funcionalidades definidas en la dimensión de departamentalización especificada en la fase anterior. Además los roles *Cloud User* y *End User* se refinan dentro de estas nuevas unidades, para especializarse en la utilización de los servicios concretos asociados a estos tipos de productos. A continuación, únicamente, se presenta la estructura interna de la unidad organizativa Negociación SLA para facilitar la claridad y compresión. La estructura interna del resto de unidades se ha modelado de forma similar.

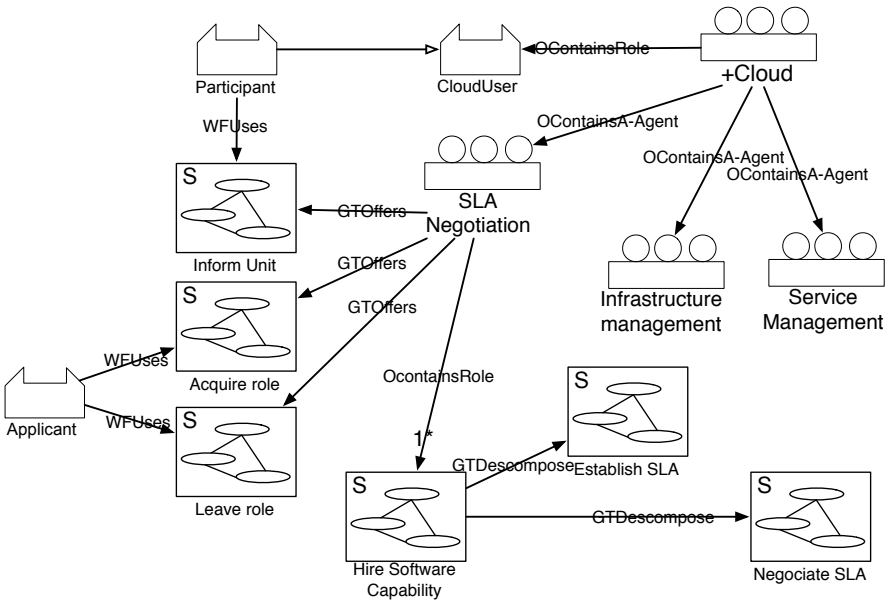

Figura 74.- Vista funcional del modelo de organización actualizado

Todas las unidades organizativas contienen los nuevos servicios de adquirir y dejar el rol e informan a la unidad, además de los servicios propios dependientes del dominio, que ya habían sido identificados anteriormente. Así por ejemplo, un usuario de tipo *Cloud User* deberá utilizar esta funcionalidad concreta de adquisición de rol, de forma previa a la funcionalidad concreta que lleve a cabo en el sistema.

A.4 DISEÑO DE LA DINÁMICA

El diseño de la dinámica de la organización está incluida en la fase de diseño. Esta etapa permite detallar de forma precisa como se comportan los servicios de la organización y cómo se produce la comunicación con el entorno, así como las interacciones que se establecen en el marco de un sistema abierto.

A.4.1 FASE E. PROCESOS DE INFORMACIÓN Y DECISIÓN

El principal objetivo de esta fase es la especificación de los flujos de información y la toma de decisiones. Gracias a lo cuál será posible determinar como se procesa la información para conseguir los resultados esperados. En primer lugar se analizan los procesos de decisión a partir del patrón de organización que se ha definido en la fase anterior. Este proceso de análisis es necesario para determinar quién ofrece los servicios adicionales que se han identificado previamente durante la fase D al aplicar el patrón organizativo. Por otro lado, para algunos de los servicios tan sólo se identificó quién los utilizaba, pero no quién los proporcionaba, por lo tanto, se revisan cada uno de los servicios, identificando los roles de gestión que se encargan de proporcionarlos.

Así pues, esta fase según la guía metodológica GORMAS incluye el (i) análisis de los procesos de decisión y (ii) el análisis de los procesos de información. Los resultados, documentos y modelos que se van a elaborar en esta fase son los siguientes:

- Documento E. Ontología el Dominio.
- Modelo de Interacción actualizado con las entidades de Interacción para cada servicio.
- Diagramas de colaboración para la especificación de las interacciones.
- Modelo de actividad actualizado.
- Modelo de entorno actualizado
- Vista social del modelo de organización.

En primer lugar, con respecto al Documento E que se corresponde con la Ontología de Domino, no se ha desarrollado una ontología específica, ya que en el estado de arte existe suficiente trabajo realizado en este sentido [Han *et* Sim, 2010] [Al Feel *et* Khafagy, 2011] [Kang *et* Sim, 2011] [Zhu *et* Wang, 2008]. Después de analizar las diferentes posibilidades ya existentes se han utilizado las dos siguientes ontologías. Por un lado se hace uso de la ontología propuestas por Yuseff *et al.* [Yuseff *et al.*, 2008], que abarca los principales conceptos generales de este tipo de plataformas tecnológicas. Y, por otro lado, también se ha hecho uso de una ontología específicamente diseñada para

la anotación de recursos computacionales [Moscato *et al.*, 2011], la cuál es de especial interés en el marco de trabajo.

A partir de la definición de la ontología del sistema +Cloud, la primer tarea de esta fase comienza con la identificación de los proveedores de servicios, continúa con la descomposición de los servicios en tareas y finaliza con la identificación de los objetivos operativos y como se relaciona con tareas. Así, en primer lugar se identificarán los proveedores de los servicios a través de la instanciación de la vista funcional (funcionalidad externa) del modelo de organización actualizado. En este sentido, en la Figura 75 se puede apreciar los nuevos roles de gestión y control que se han identificado durante esta fase y cuyo cometido consiste en proporcionar los servicios que son ofertados por la plataforma. En esta vista funcional, por claridad y para facilitar la comprensión, sólo se presentan los servicios relativos a la dinamicidad de un entorno abierto (adquirir rol, y dejar rol e informar unidad) identificados en la fase anterior para la unidad organizativa de negociación de acuerdos SLA. No obstante, este conjunto de servicios y proveedores es similar en el resto de unidades organizativas.

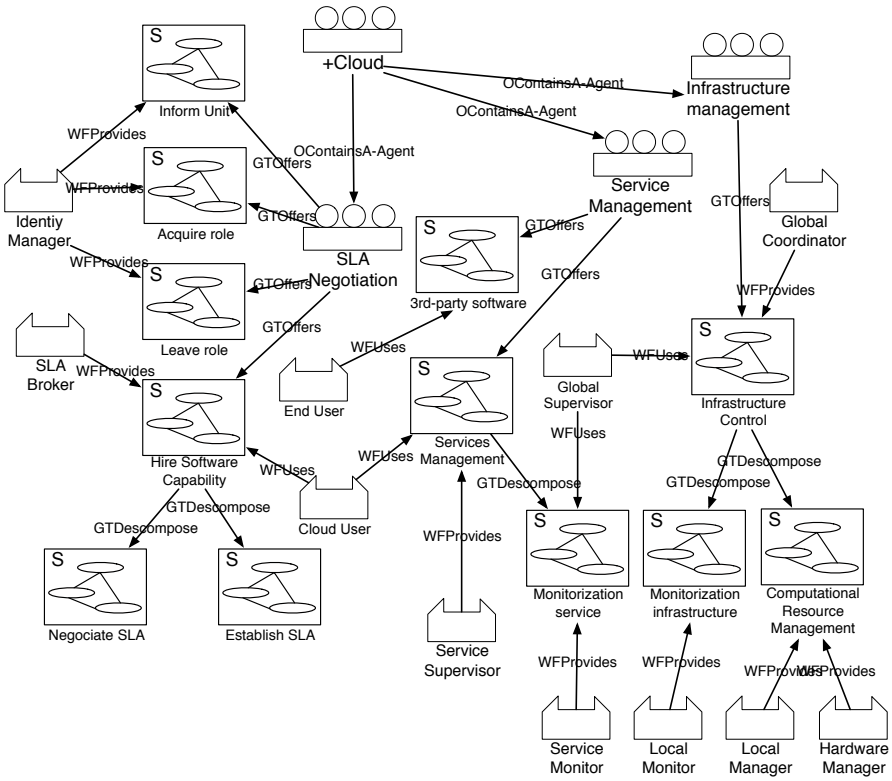

Figura 75.- Vista funcional (funcionalidad externa) del modelo de organización actualizado

Durante esta fase también se identifican las nuevas tareas que se asocian a los nuevos servicios del sistema, los cuáles están relacionados con la dinámica de un entorno abierto. A modo de ejemplo se muestra el perfil del modelo de actividad para el serviciode adquisición de rol en la Figura 76 donde se puede apreciar la descomposición del servicio entre las diferentes A-Tareas que lo forman. Finalmente, la Figura 77 presenta la relación existente entre este conjunto de A-Tareas identificadas y sus flujos de entrada y salida.

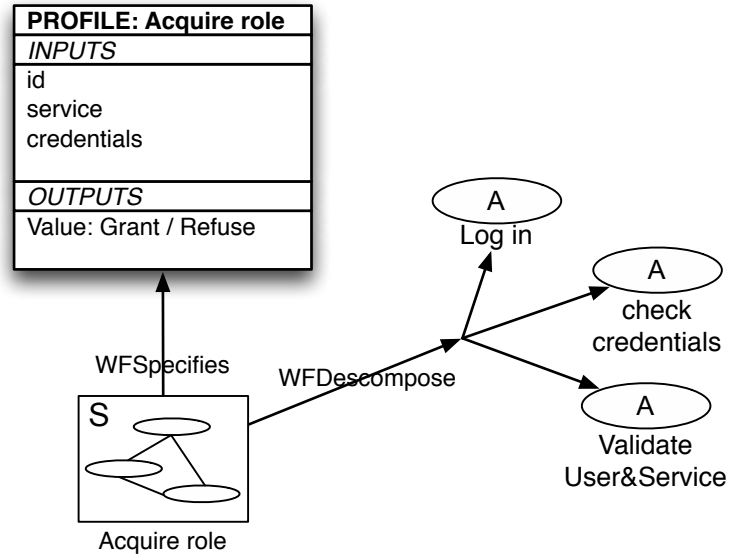

Figura 76.- Diagrama del modelo de actividad actualizado (Perfil) del servicio Adquirir rol

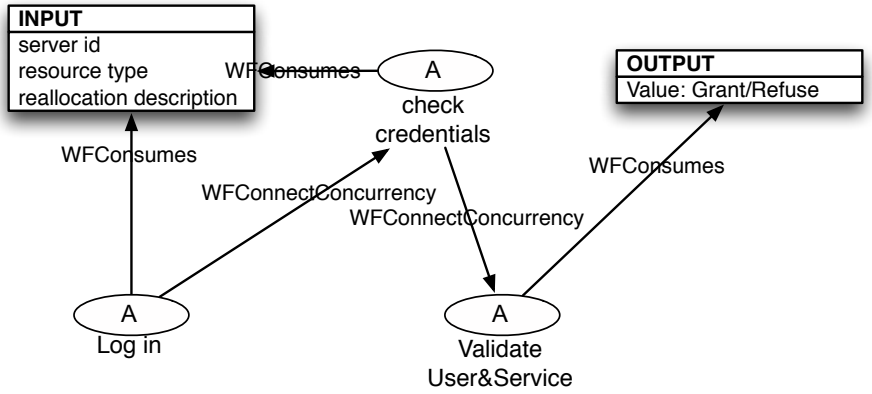

Figura 77.- Diagrama del modelo de actividad actualizado (Relación entre las A-TareaS) del servicio Adquirir rol

Continuando con la metodología, el siguiente paso consiste en la identificación de los objetivos operativos a partir de los objetivos funcionales que ya se presentaron en la Figura 69. Así, a partir de esta definición inicial de los objetivo funcionales que se derivan de las misiones de la unidad organizativa +cloud, en la Figura 78 se presenta el diagrama del modelo de actividad que recoge la descomposición de los objetivos funcionales en objetivos operativos.

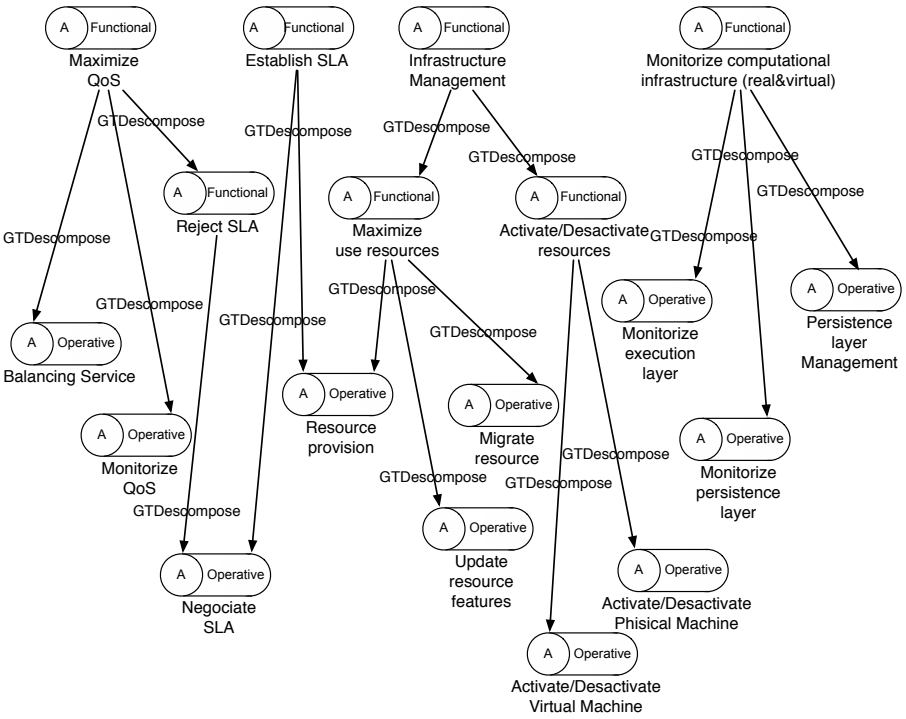

Figura 78.- Modelo de actividad actualizado. Descomposición de objetivos funcionales en objetivos operativos

Finalmente, se concluye la primera parte de esta fase E centrada en el análisis de los procesos de decisión, con la actualización del modelo de actividad en el que se relacionan las A-Tareas identificadas con cada uno de los objetivos operativos. Como muestra, en la Figura 79 se presenta un diagrama que incluye las A-Tareas asociadas al servicio de monitorización de infraestructura y como éstas se asocian a los objetivos operativos de la unidad organizativa de gestión de la infraestructura subyacente.

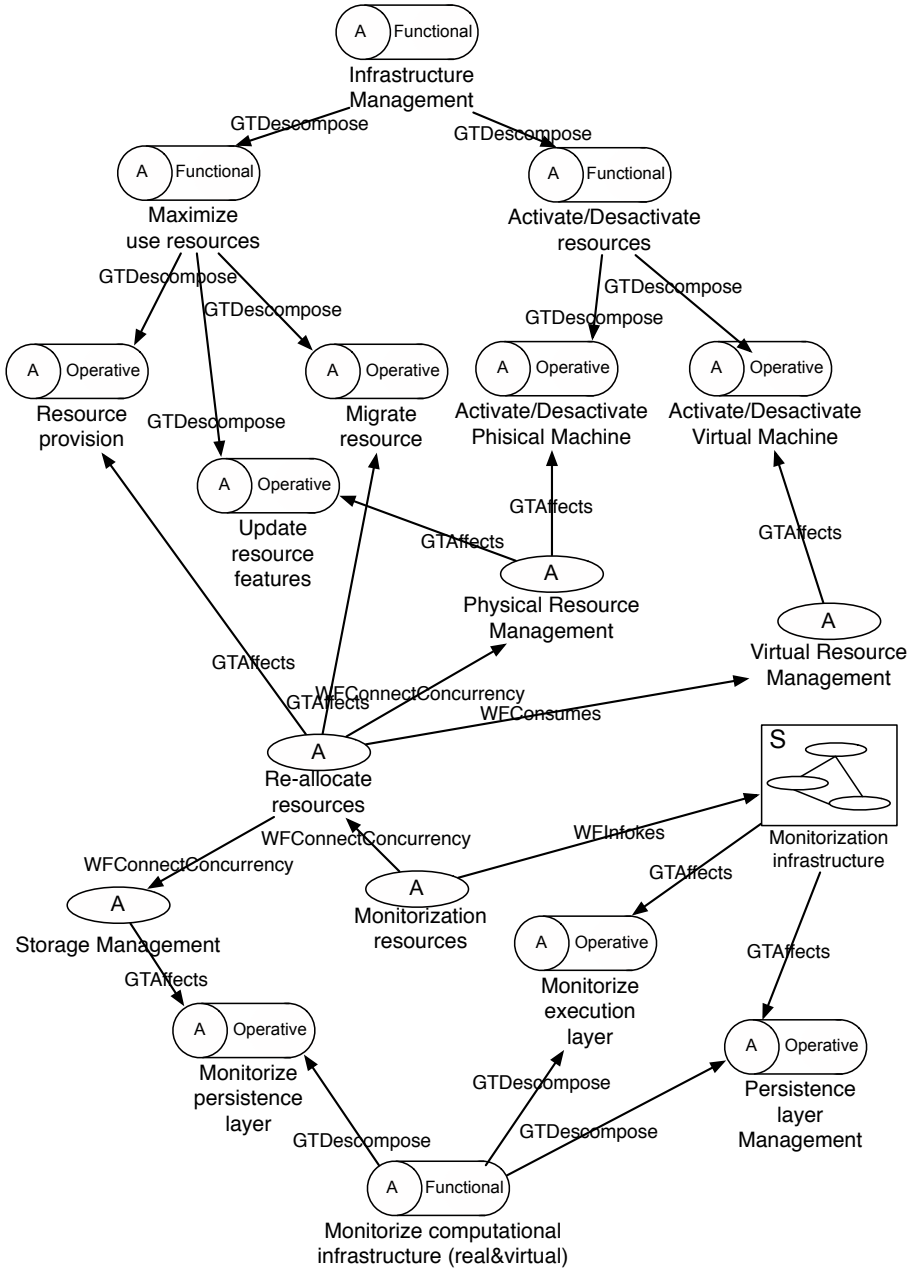

Figura 79.- Modelo de actividad actualizado. Descomposición de los objetivos funcionales en objetivos operativos. Asociación a las tareas del servicio Monitorización de infrastructura.

A continuación se describirán los cambios realizados en el modelo de entorno para la redistribución de recursos. El diagrama de este modelo

actualizado se puede apreciar en la Figura 83. Las principales actualizaciones se describen a continuación:

- El nuevo modelo incluye un nuevo repositorio que se ha identificado durante la fase anterior y cuyo objetivo es almacenar los privilegios de los roles externos.
- También se puede apreciar la existencia de diferentes puertos para el acceso a cada uno de los recursos del entorno. En este sentido, además de los puertos existentes para el acceso a los recursos que ya se han identificado previamente (repositorios e interfaces), cabe destacar la existencia de nuevos puertos que hacen posible el acceso al control de la infrastructura subyacente. Estos puertos son el de control del entorno *hardware* físico, el de la capa de virtualización (*hypervisor*) y el puerto que permite balancear un servicio entre diferentes nodos que colaboran para ofertar el mismo.
- Finalmente, en el diagrama también se aprecia como los diferentes roles del sistema interaccionan con el entorno para conseguir sus objetivos individuales.

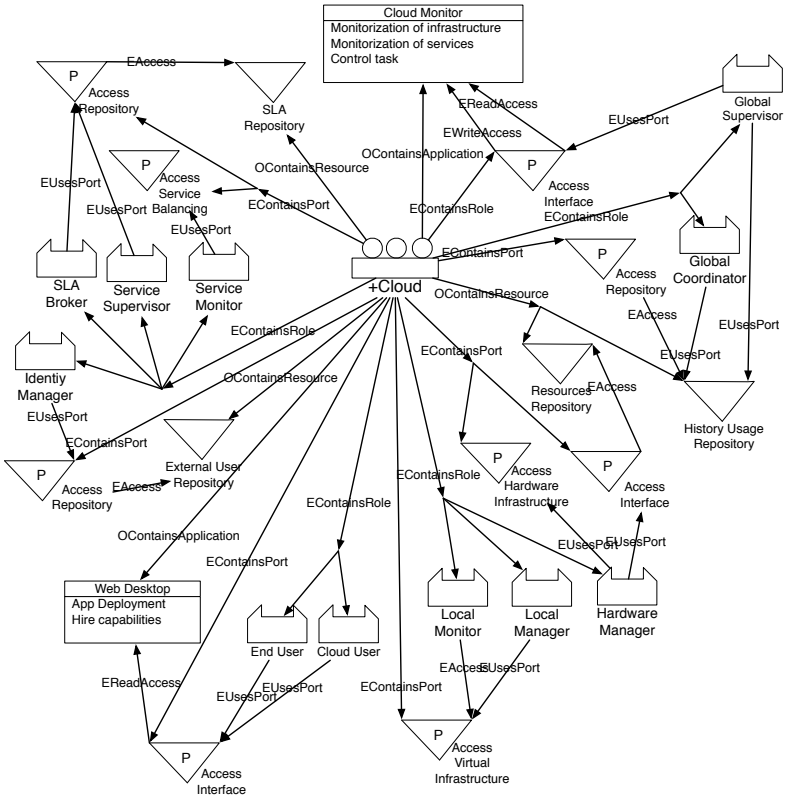

Figura 80.- Modelo de entorno actualizado. Acceso a los puertos del entorno

Finalmente, la segunda parte del diseño en la fase F está centrada en el análisis de los procesos de información. Para ello, en primer lugar es necesario actualizar el modelo de organización incluyendo en la especificación cuáles son los roles que pueden interaccionar entre sí. En este sentido, en la vista estructural que se presenta en la Figura 81 se incluye como novedad la relación (*AGInformation*) que representa los roles que están habilitados para conocerse entre sí, y por lo tanto, comunicarse, interaccionar y negociar con el objetivo de conseguir llevar a cabo sus tareas individuales.

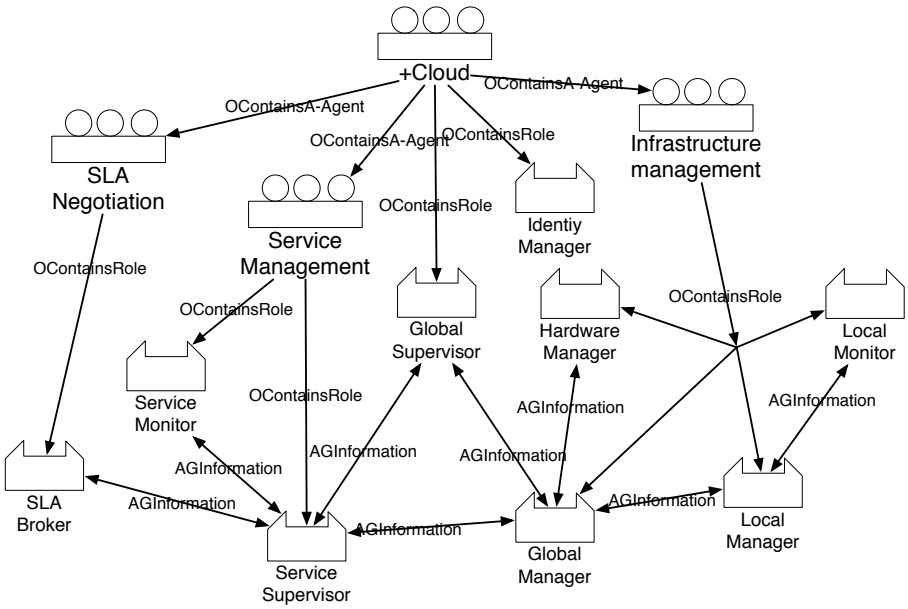

Figura 81.- Vista estructural del modelo de organización actualizado

Siguiendo con el análisis de los procesos de información, la metodología GORMAS utiliza el modelo de interacción para describir las diferentes secuencias de intercambio de datos entre los roles de la plataforma de forma que sea posible proporcionar la funcionalidad requerida. En este sentido, la Figura 82 presenta una muestra del modelo de interacción donde se puede apreciar cómo fluje la información entre los diferentes roles, los servicios utilizados y su relación con los objetivos funcionales del sistema.

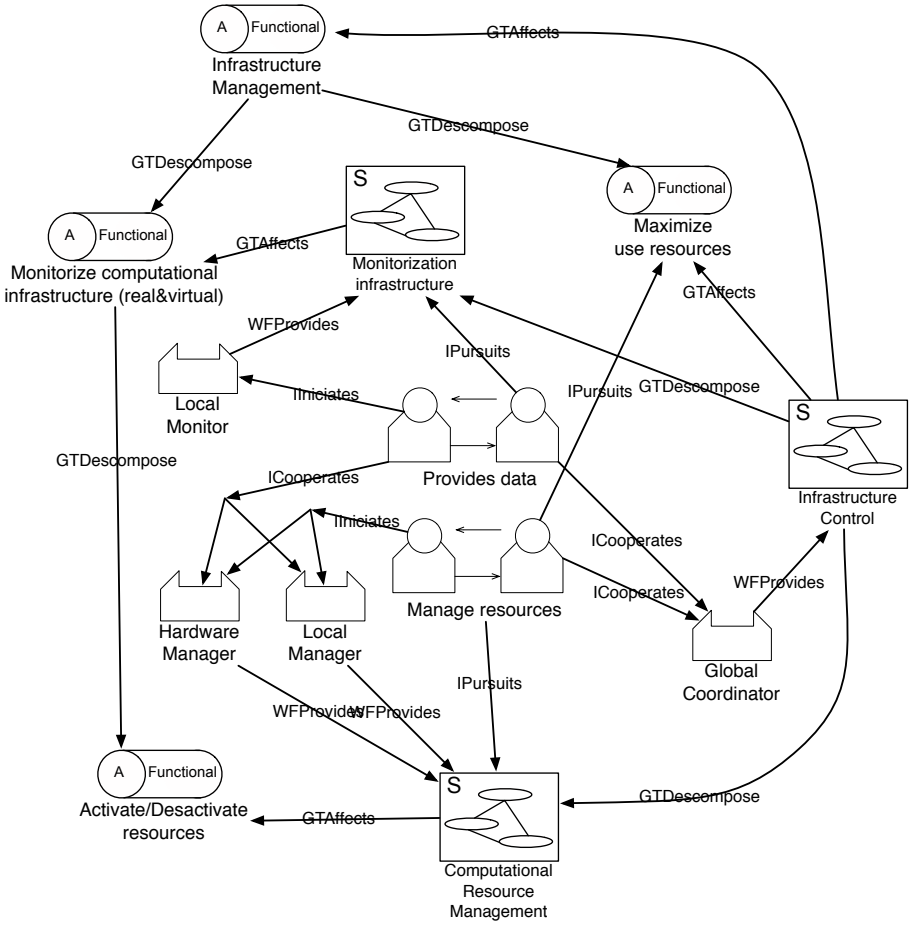

Figura 82.- Modelo de interacción para la unidad organizativa de gestión de la infrastructura

A continuación, la Figura 83 de forma complementaría al modelo de interacción, también se presenta un diagrama de colaboración entre los roles de la plataforma para llevar a cabo una redistribución de recursos a petición de un agente supervisor de servicio. En el diagrama se indican los roles que participan en la interacción y la ejecución del servicio que provoca su activación.

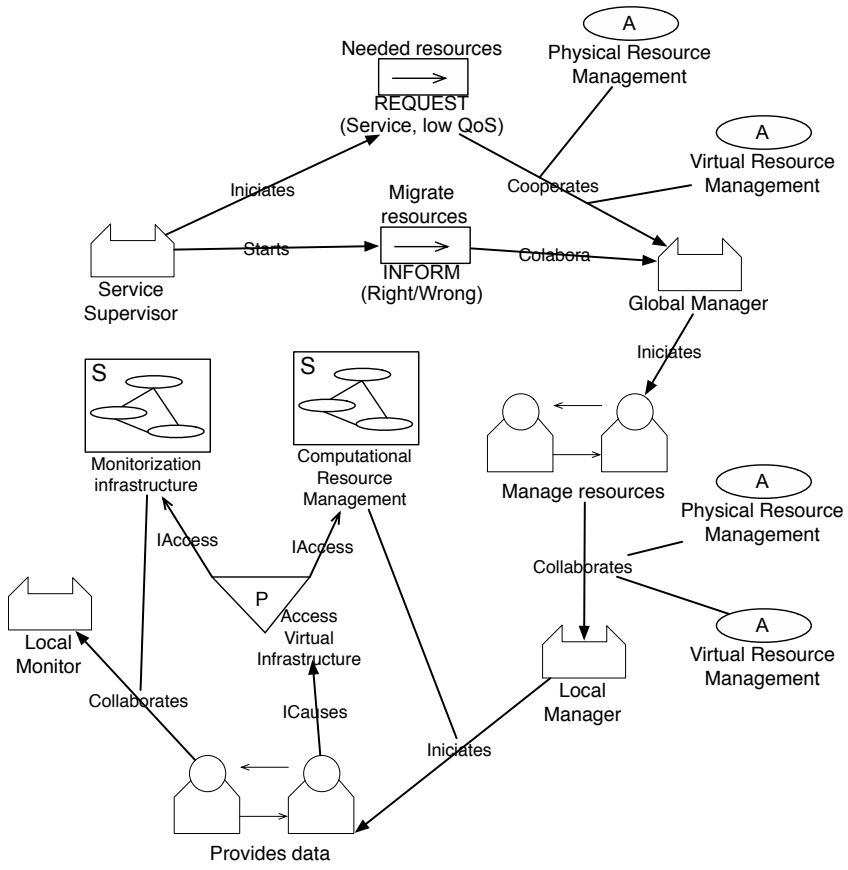

Figura 83.- Diagrama del modelo de colaboración de las actividades

A.4.2 FASE F. DINAMICIDAD DEL SISTEMA ABIERTO

Las organizaciones artificiales en sistemas abiertos, al igual que las organizaciones humanas, deben definir políticas de recursos humanos para la captación, selección, formación y promoción del personal [Moreno-Luzon *et al.*, 2001]. Gracias a estas políticas, la organización puede determinar los roles de cada nuevo miembro y la visibilidad de los objetivos de la organización.

El meta-modelo de agente define dos tipos de agentes. En primer lugar, los agentes internos son desarrollados únicamente por los diseñadores del sistema y llevan asociados roles no accesibles, mientras que en segundo lugar los agentes externos se identifican con todos aquellos roles que participan en la organización, pero su procedencia es heterogénea. Estos agentes externos deben partipaciar en el sistema lo que implica que deben adoptar roles accesibles en las unidades organizativas y se debe controlar su comportamiento a través de normas asociadas al rol.

Así el modelo dinámico del sistema abierto permite (i) determinar la funcionalidad a publicar, (ii) las políticas de adquisición de roles, y (iii) el diseño de los agentes propios del sistema. Los resultados, documentos y modelos que se van a elaboran en esta fase son los siguientes:

- Modelo de entorno actualizado.
- Documento de normas.

El sistema +Cloud se ha diseñado como un sistema abierto, en el que se permite que agentes externos ofrezcan dentro de él servicios y aporten funcionalidad al sistema. Para dar visibilidad a esta condición de sistema abierto, la Figura 84 presenta la última actualización del modelo de organización donde se pueden apreciar los diferentes agente, ya sean internos o externos, que forman parte del sistema. En este sentido cabe destacar, en primer lugar los roles que están directamente relacionados con agentes humanos los cuáles, han sido claramente identificados en fases anteriores. Sin embargo también existen otros dos roles dentro del sistema +Cloud cuyas funcionalidades pueden ser desarrollados por agentes externos. Éstos son el rol Gestor SLA y Supervisor de servicios, que pueden asociarse a los roles *Broker Cloud* y *Auditor Cloud*, respectivamente, siguiendo el modelo de roles propuestos por el NIST en su arquitectura de referencia [Liu *et al.*, 2011].

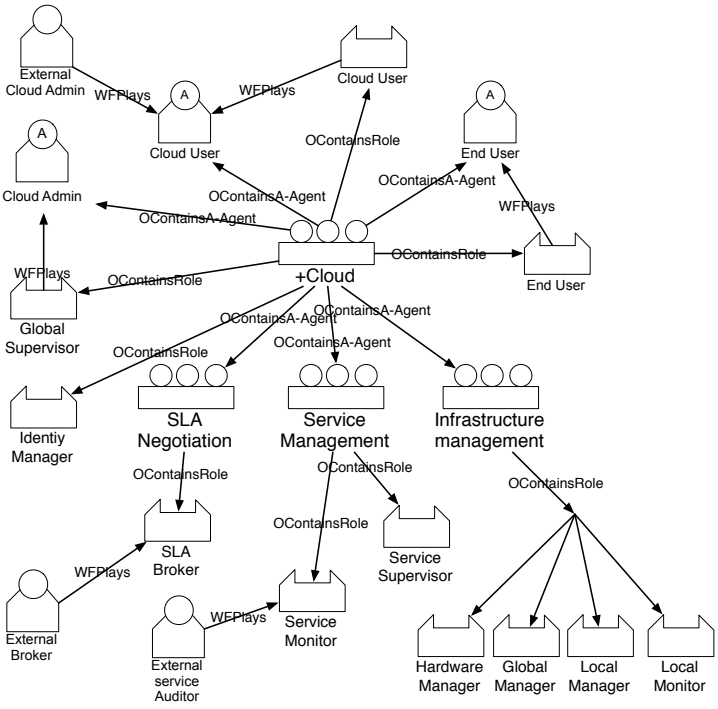

Figura 84.- Vista estructural del modelo de organización actualizado

Así mismo, dentro de esta fase es necesario definir las normas asociadas a la publicación de servicios, el control de cardinalidad de roles, su compatibilidad, así como el orden de ejecución de los servicios. Las normas que se definen están relacionadas con las siguientes tareas:

- Obligar a la unidad organizativa que requiere el servicio a registrar en un directorio de servicios donde se solicitan proveedores del mismo.
- Obligar a esa unidad organizativa a eliminar el registro de dicha demanda de proveedores cuando el número de agentes participantes en la unidad que juegen el rol proveedor sea excesivo y ya no se desee que entren nuevos proveedores.

Como se aprecia, estas normas que se definen de forma genérica en la metodología GORMAS, tienen una aplicación directa en la publicación de las capacidades computacionales (productos) que ofrece el sistema +Cloud. En la Figura 85 se presenta una muestra del documento de normas identificadas durante esta fase perteneciente diseño de la dinamicidad del sistema abierto +Cloud.

```
OBLIGUED OUServiceManagement PUBLISH 3rd-party software
AFTER SLA = accepted
```

Figura 85.- Ejemplo de normas de publicación de un servicio (Software de terceros)

A.4.3 FASE G. SISTEMA DE MEDICIÓN, EVALUACIÓN Y CONTROL

En las organizaciones humanas, los sistemas de control del desempeño se realizan a través de la normalización de tareas (diseño de procesos), la normalización de los resultados (establecimiento de objetivos) y la normalización de habilidades (asunción de creencias, valores y principios comunes) [Moreno-Luzon *et al.*, 2001]. A tenor de ello, durante esta fase se establecen el conjunto de normas y restricciones resultantes de la aplicación de estos tres tipos de normalización.

Por lo tanto, esta fase consta de la aplicación de la (i) normalización de tareas, (ii) la normalización de resultados y, finalmente, (iii) la normalización de habilidades. Los resultados, documentos y modelos que se van a elaboran en esta fase son los siguientes:

- Documento de normas actualizado

En primer lugar, se comienza la fase atendiendo a la normalización de tareas. Esta normalización implica la especificación de reglas que aseguren el correcto orden de ejecución e invocación de los servicios, sus límites de tiempos, el acceso a recursos, etc. Este tipo de normas sólo deben especificarse en el documento B1 que trata sobre la tecnología esencial del

sistema y que ya se ha presentado la Tabla 15 se identificó la tecnología esencial como una tecnología de vinculación prolongada. En el caso del sistema +Cloud, como la tecnología esencial del sistema es producción continua y en pequeños lotes no resulta necesario especificar este tipo de normas, por lo que no resulta necesario actualizar el documento incluyendo un marco normativo para la normalización de tareas.

A continuación con respeto a la normalización de resultados, se deben establecer normas que controlen los productos y servicios ofertados por la plataforma. Para ello, se tiene que revisar la especificación que se ha detallado en los Documentos B.2. sobre la tecnología de la unidad de trabajo (Tabla 16 y Tabla 17) y el Documento A.2. sobre los grupo de interés de la plataforma (Tabla 15). Para definir el marco normativo se puede seguir uno de los siguientes tres enfoques: la (i) calidad como conformidad a unas especificaciones, (ii) calidad como satisfacción de las expectativas del cliente y, finalmente, (iii) calidad como valor con relación al precio. En el marco del sistema +Cloud, dado que para que un servicio sea consumido por un rol externo debe existir un acuerdo de uso asociado al mismo, las normas de normalización de resultados atenderán a la calidad como conformidad a unas especificaciones, en este caso el acuerdo SLA. Así, la Figura 86 incluye una muestra de las normas especificadas en el documento de normas del sistema +Cloud.

```
OBLIGUED CloudUser REGISTER Service
PROFILE SLABroker PROCESS ?p BEFORE hiring
```

Figura 86.- Ejemplo de normas de normalización de resultados

Finalmente, la normalización de habilidades dentro de los SMA se trata a través del concepto de rol, indicando permisos, conocimientos y aptitudes que asume o adquiere un agente al jugar dicho rol. La adopción de roles por parte de agentes se ha tratado ampliamente ya en la fase anterior (Apartado A.4.2), lo que no hace necesaria la inclusión de normas adicionales en esta fase del desarrollo.

A.4.4 FASE H. SISTEMA DE RECOMPENSAS

Una vez que ya se ha definido el sistema, finalmente la última fase de la guía metodológica GORMAS permite especificar el sistema de incentivos para recompensar a los miembros del sistema que avancen en la dirección de los intereses o estrategias de la organización, es decir, contribuyan a satisfacer la misión de la organización. En las organizaciones humanas, los incentivos se plantean como recompensas a las entidades individuales dispuestas a aportar su esfuerzo en un sistema común de cooperación, compartiendo un mismo objetivo o finalidad [Moreno-Luzon *et al.*, 2001].

En ambos sistemas (humanos y artificiales), los miembros presentan pluralidad de intereses y objetivos por lo que la organización debe establecer un sistema de recompensas adecuado a su contribución con la organización. En este sentido se identifican dos tipos diferentes de recompensas en función de las áreas que se quiera mejorar:

- Valores de conservación, que se asocian a la posibilidad de mantenerse en el sistema, progresar en él o adquirir nuevos permisos y reconocimientos.
- Recompensas materiales o inmateriales, que compensan un determinado esfuerzo.

En primer lugar se analiza el tipo de comportamiento que se quiere recompensar, para lo que se hace necesario realizar un análisis de qué tipo de comportamientos generales se desean potenciar, especialmente cuando se trata de un sistema abierto. En este sentido existen cuatro tipos de comportamientos a recompensar [Galbraith, 1977]: (i) la voluntad de unirse a un grupo y permancer, (ii) el rendimiento dependiente del rol que desempeñan, (iii) el esfuerzo sobre unos niveles mínimos (rendimiento, eficiencia y productividad), y, finalmente, (iv) los comportamientos cooperativos. El resultado del análisis sobre los tipos de comportamiento a potenciar se presenta a continuación en el Documento H (Tabla 19).

H- Sistema de recompensas	
Comportamiento a fomentar	**Sistema de recompensa**
Unirse y permanecer	*Recompensa individual.* Se opta por recompensas individuales para aquellos agentes externos que jueguen roles externos dentro del sistema +Cloud, ya sean agentes humanos o artificiales.
Esfuerzo sobre niveles mínimos	*Recompensa individual.* Dado que el patrón de la organización es de tipo coalición las recompensas deben ser siempre individuales.
Cooperación	*Recompensación de grupo.* Se implementan recompensas de grupo como premio a la colaboración en la consecución de los objetivos de la organización en su conjunto o de las unidades organizativas que la forman

Tabla 19.- Documento H.- Sistema de recompensas

Posteriormente, una vez definido como se llevarán a cabo las recompensas, (individuales o en grupo), la siguiente tarea consiste en analizar el sistema concreto de recompensas a utilizar. En este sentido, para recompensas individuales existes dos modelos de recompensas, en primer lugar, un sistema basado en puntos en el que un agente recibe una cantidad inicial de puntos y se

van incrementando los puntos periódicamente siempre y cuando el agente no haya sido sancionado. Estas sanciones están basadas en la retirada de puntos. En segundo lugar, el otro sistema de recompensa individual hace referencia al control de la honestidad del agente, en este sistema el agente recibe o se le retira crédito en función de la valoración de sus acciones. Por otro lado, con respecto a las recompensas de grupo se establecen medidas de rendimiento del grupo, que hagan posible que un controlador externo pueda controlar de forma objetiva el rendimiento del grupo, gratificando a todos los miembros por igual. Las recompensas del grupo pueden ser cooperativas cuando se gratifica a todos los miembros del grupo por igual, o competitiva cuando la gratificación de la recompensa varía en función del rendimiento individual.

En el sistema que se está desarrollando se selecciona un sistema de recompensas basado en puntos para gratificar a los agentes a título individual. En nuestro caso se aplican estas recompensas principalmente a los agentes externos que participan en el sistema para fomentar su permanencia en el mismo. Por otro lado, también se modelan recompensas de grupo, siguiendo un modelo cooperativo para compensar las interaciones del sistema. En este caso, se aplican este tipo de recompensas cuando la distribución de los recursos computacionales en función de la carga de los servicios y los acuerdos SLA se realizan adecuadamente.

Para finalizar esta última fase H de la guía metodológica GORMAS, se aplica el sistema de recomponensas, para ello se actualiza el documento normativo con la inclusión de normas relativas al sistema de recompensas según el modelo y tipo que se han detallado. En la Figura 87 se muestra un subconjunto de normas definidas durante esta fase del diseño de la dinámica del sistema +Cloud.

```
OBLIGUED ?a:LocalManager REQUEST ReadaptResources
AFTER ?a REQUEST ServiceResourcesInform
```

Figura 87.- Ejemplo de normas de normalización de resultados

ANEXO B
LA PLATAFORMA DE EVALUACIÓN DEL SISTEMA +CLOUD

CC [Mell *et* Grance, 2011] es el nombre por el que se conoce al novedoso paradigma tecnológico en el que cualquier forma de computación puede ser ofertada como servicio. Es decir, esta tecnología es capaz de ofertar no sólo *software* sino también infraestructura (*hardware*) como servicio a través de Internet. Dentro de este paradigma, tanto la lógica de negocio como la información que manejan las aplicaciones se almacenan en entornos computacionales remotos y, por tanto, no está vinculada a los terminales desde los que se accede. CC incluye tres niveles de servicio ampliamente conocidos: *software*, plataforma e infraestructura.

Sin embargo, la variedad de los servicios ofertados no son más que la vista externa de este paradigma computacional. Más allá de ellos, la característica clave reside en el nivel de calidad con la que se exponen estos servicios, la cuál es independiente de la demanda de los mismos por parte de los usuarios [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012]. En la práctica esto se traduce en que el usuario final hace uso de un conjunto de recursos (capacidades) en la nube, que siempre están disponibles y que a priori son ilimitados. Por tanto, el aspecto diferenciador de este paradigma computacional reside en la gestión automatizada de los recursos *hardware* y su abstracción en recursos virtuales, que hace posible la gestión elástica de los recursos en función de la demanda. Gracias a ello, es posible obtener una imagen externa de recursos ilimitados, mediante el sostenimiento prolongado del nivel de calidad de los productos ofertados como servicio.

El concepto no es algo nuevo, sin embargo, la forma de gestión de la infraestructura subyacente sí que lo es [Buyya *et al.*, 2009] [Armbrust *et al.*, 2010]. Mientras que en tecnologías anteriores se utilizaba el balanceo de carga entre los recursos, en un entorno CC se habla de elasticidad. Este novedoso concepto no sólo implica el balanceo de la demanda entre los recursos disponibles, sino también la creación y/o destrucción de los recursos en función de la demanda instantánea por parte de los usuarios. Esta nueva característica es posible gracias a la maduración de la tecnología de virtualización en los últimos tiempos [Che *et al.*, 2008]. Esta tecnología aporta innumerables ventajas como la reducción de la inversión inicial en *hardware*, la gestión dinámica (e incluso automática) de los recursos, el mantenimiento de la calidad en los servicios, la reducción de los costes (electricidad, refrigeración, etc.), así como la evolución hacia un entorno computacional mucho más

saludable con el medio ambiente en el marco de los sistemas *Green IT*[73] [Hooper *et al.*, 2008] [Bing *et al.*, 2009].

A lo largo de este anexo se presentará la plataforma CC desarrollada en el marco de este trabajo de investigación por el grupo de investigación BISITE[74] y que integra la arquitectura propuesta **+Cloud**, el cuál se propone en el marco de esta investigación como modelo organizativo de SMA para el control y monitorización de la plataforma CC. Esta plataforma ha sido el entorno de pruebas de los algoritmos, técnicas y herramientas propuestos y desarrollados en el marco de esta tesis doctoral.

Esta plataforma CC es capaz de prestar un conjunto de servicios de bajo nivel que habilitan el desarrollo y despliegue de aplicaciones web escalables orientadas a Internet. La plataforma agrupa los diferentes aspectos relevantes de este paradigma de computación incluyendo:

- La gestión de la infraestructura la cuál se encarga de gestionar los recursos *hardware*, reconfigurando los proveedores de servicio, gestionando el balanceo de carga y permitiendo el despliegue, tanto de los servicios que componen la plataforma, como de las aplicaciones que se desarrollen por terceros y se exploten a través de la plataforma CC.
- La creación de aplicaciones susceptibles de ser desplegadas en la plataforma desarrollada, haciendo especial énfasis en las características más problemáticas en cuanto a su escalabilidad, es decir, a los servicios de persistencia de datos. En este sentido, la plataforma proveé de mecanismos para la persistencia de la información y la gestión de usuarios.

Este anexo se organiza en dos grandes bloques, el primero de ellos contenido en el apartado B.1, el cuál describe las funcionalidades externas de la plataforma CC siguiendo los tipos de capacidades propuestos por el NIST [Mell *et* Grance, 2011]: el nivel *software* en el apartado B.1.1, el nivel plataforma en el apartado B.1.2, y finalmente el nivel de infraestructura en el apartado B.1.3. Por su parte, el segundo bloque describe los componentes tecnológicos que se han utilizado para hacer posible la funcionalidad de esta plataforma, incluyendo su persistencia (apartado B.2.1), el balanceo de carga (B.2.2), el sistema de comunicación (B.2.3), la capa de virtualización (B.2.4) y, finalmente, una breve descripción del sistema de control (o) que se explica con gran detalle en el Capítulo 4 de este trabajo de investigación.

[73] http://www.greenit.net/
[74] http://bisite.usal.es

B.1 SERVICIOS EXTERNOS

Se propone un modelo de computación en nube que abarca las tres vertientes fundamentales en cuanto a la oferta de servicios del paradigma CC. La descripción de la plataforma se llevará a cabo deacuerdo con su descomposición lógica y funcional en dichos servicios (*software*, plataforma e infrastructura).

B.1.1 LA CAPA SaaS (*SOFTWARE* COMO SERVICIO)

La capa *software* está constituida por las aplicaciones *software* a las que los usuarios finales del entorno de computación tienen acceso. Las aplicaciones SaaS delegan en la capa PaaS sus necesidades de persistencia, tanto en lo referente a datos, como a ficheros. También delegan en la capa PaaS la autenticación de usuarios y gestión sesiones. Las aplicaciones SaaS son aplicaciones web, y por lo tanto accesibles para los usuarios finales a través de Internet.

Para que las aplicaciones de esta capa sean realmente escalables, debe emplearse un estilo arquitectónico apropiado para su construcción. Toda la información almacenada, obligatoriamente debe hacerse mediante los servicios proporcionados por la capa inmediatamente inferior (PaaS). De esta manera, las aplicaciones pueden ser replicadas sin la posibilidad de que una réplica guarde información que no esté accesible por las demás. Esta sencilla norma permite efectuar el balanceo de las peticiones y, por lo tanto, el reparto de la carga entre las distintas réplicas de un servicio sin que se pierda información o se produzcan incongruencias en la información disponible en cada una de las réplicas, lo que derivaría en resultados erróneos e impredecibles.

Para que el usuario sienta que está interactuando con una aplicación real y no con una página web estática, se han empleado las últimas técnicas en el desarrollo de aplicaciones web. Utilizando las novedades ofrecidas por HTML5 [75] y CSS3 [76]. Así como el novedoso protocolo de comunicación WebSockets[77] que permite actualizar los contenidos visualizados por el cliente sin necesidad de recargar páginas o realizar sondeos periódicos. También se hace uso intensivo de bibliotecas como jQuery[78] que faciliten la manipulación del árbol del documento estático de las páginas web, y finalmente, también se

[75] http://www.w3.org/html/
[76] http://www.w3.org/Style/CSS/
[77] https://tools.ietf.org/html/rfc6455
[78] http://jquery.com/

integra la tecnología AJAX[79] que garantiza la compatibilidad del código entre navegadores.

Los servicios a este nivel se dividen en dos tipos, en primer lugar, las aplicaciones nativas, que permiten la gestión del propio entorno, de sus usuarios. En segundo lugar, las aplicaciones desarrolladas por terceros y desplegadas en el entorno CC. Todas las aplicaciones y servicios desarrollados se integran dentro de un escritorio virtual que sirve de punto de acceso a la plataforma, de forma que es posible interactuar con él de una forma similar a como se haría a través de escritorio en un sistema operativo tradicional. A continuación se introducen las aplicaciones que se enmarcan dentro del entorno cloud que se ha desarrollado para su uso en la plataforma.

B.1.1.1 EL ESCRITORIO VIRTUAL

El escritorio virtual, Figura 88, emula el escritorio de un sistema operativo tradicional y sirve de punto de acceso unificado a todas las aplicaciones y funcionalidades de la plataforma. A través de este escritorio, los usuarios tienen acceso directo a las aplicaciones favoritas de cada usuario individual, así como un menú desplegable que permite acceder a la lista completa de aplicaciones disponibles para cada usuario. Finalmente, también incorpora un conjunto de opciones de configuración y personalización.

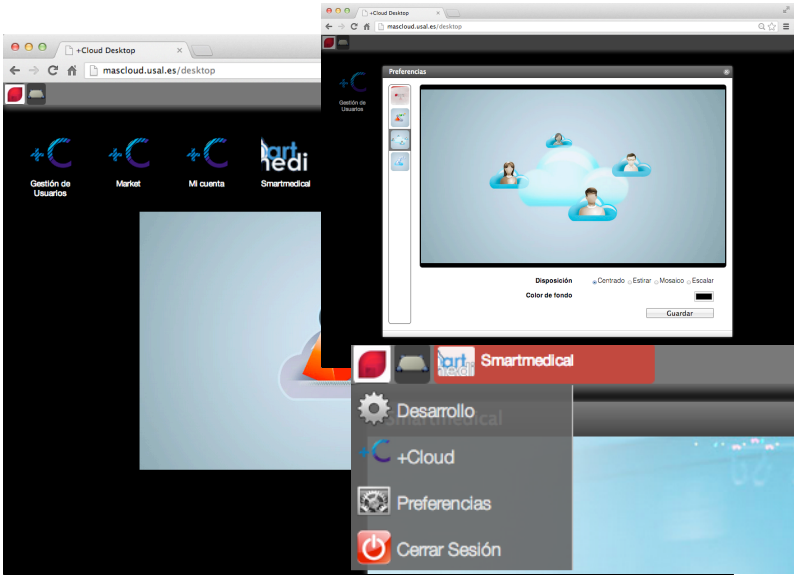

Figura 88.- Escritorio virtual de la plataforma CC

[79] http://www.w3.org/TR/XMLHttpRequest/

Dado que la experiencia de usuario es un aspecto de vital importancia y, en cierta medida, determina al grado de aceptación de la plataforma por parte de los usuarios, el aspecto de presentación del escritorio es altamente personalizable ya que se pueden ajustar parámetros como la posición de los iconos, el tamaño y posición de las ventanas, así como el fondo del espacio de trabajo.

B.1.1.2 LAS APLICACIONES NATIVAS

Las aplicaciones nativas están especialmente orientadas a la gestión del entorno CC. Permiten actualizar la configuración de la plataforma, tanto a nivel global de toda la plataforma computacional por parte de los administradores de la misma; como a nivel individual por parte de los usuarios, en función del nivel de privilegios

A continuación se describen las aplicaciones nativas de la plataforma:

- El **panel gestión personal** que está disponible para todos los usuarios de la plataforma. Desde él, se puede realizar la gestión de la información individual de cada usuario, así como gestionar las compras o el pago por uso de las aplicaciones en el entorno CC. En lo relativo a su cuenta de usuario, se les permite visualizar sus datos personales y modificarlos. Del mismo, también podrán visualizar un listado de las aplicaciones que tienen disponibles en un determinado momento.
- El **catálogo de aplicaciones** de la plataforma está disponible para los usuarios a través de un *mercado de aplicaciones*. Este mercado sigue la metáfora de tienda, a la que los usuarios se han acostumbrado en los últimos tiempos debido al cambio de paradigma en la distribución de *software* en las plataformas móviles y los proveedores de aplicaciones para la web.
  Esta aplicación nativa consiste, fundamentalmente, en un catálogo de aplicaciones, a través del cuál es posible consultar las aplicaciones disponibles y la información más relevante sobre éstas. La información provista por los desarrolladores de la aplicación se complementa con información provista por los usuarios que la utilicen (opiniones).
  Cuando un usuario indica que desea adquirir una aplicación, el control se transfiere a dicha aplicación por si fuera necesario realizar un pago o aceptar algún tipo de licencia de uso. Si el usuario acepta las condiciones de uso, un servicio web permite a la aplicación comunicar al entorno CC la suscripción del usuario, utilizando para ello el gestor de identidad que posteriormente se presentará en el apartado B.1.2.1.1.
- La aplicación de **gestión de usuarios** permite la manipulación de cuentas de usuario y sus roles asignados. En este sentido, en la plataforma se distingue entre roles de aplicación y roles globales. Los roles de aplicación sólo tienen efecto en cada aplicación concreta, sin

afectar a las demás. Por su parte, los roles globales son efectivos a nivel de todo el entorno CC, pero son independientes de los roles que cada usuario tenga en las aplicaciones.

- El **panel de administración de aplicaciones** permite gestionar opciones referentes a las aplicaciones desplegadas en la plataforma, así como a la interacción que hacen los usuarios con esas aplicaciones.
  Las aplicaciones del entorno CC pueden tener tres niveles de acceso (público, privado o restringido). Estos niveles de acceso especifican qué conjunto de usuarios y bajo qué condiciones pueden utilizar la aplicación concreta que se esté configurando.

En la Figura 89 se muestra una captura de pantalla que incluye todas las aplicaciones nativas de la plataforma.

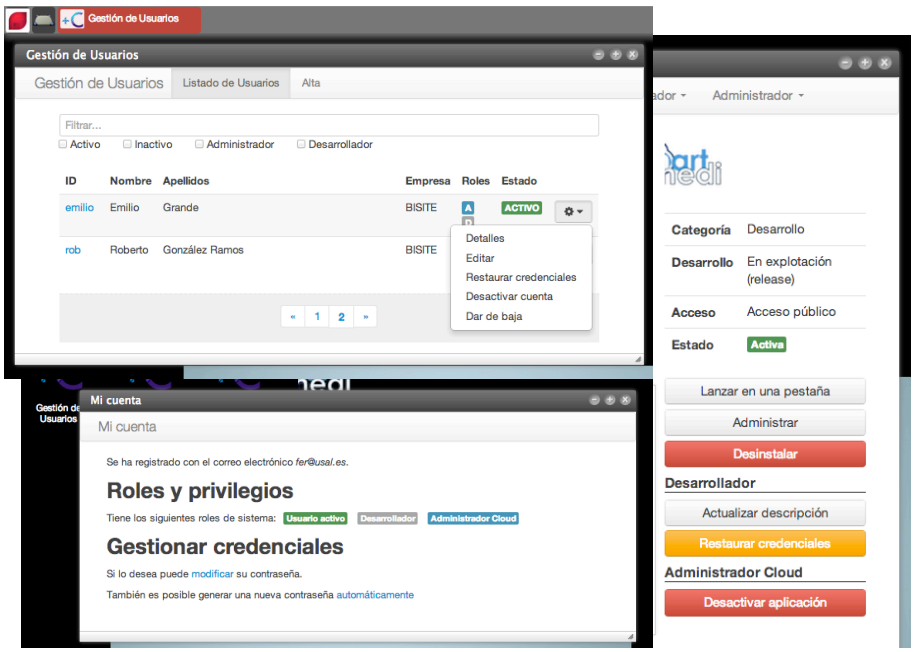

Figura 89.- Aplicaciones nativas de la plataforma CC

B.1.1.3 EL PANEL DE CONTROL DE LA INFRAESTRUCTURA

El panel de control de la infraestructura de la plataforma CC permite monitorizar y gestionar tanto la infraestructura subyacente (máquinas físicas y virtuales), como los servicios desplegados en la plataforma.

A través de la *vista de monitorización y control* de la infrastructura (Figura 90) es posible obtener una idea general del estado de la plataforma CC incluyendo servidores físicos y virtuales. Gracias a esta vista, se facilita en gran medida la supervisión de la infraestructura interna de la plataforma CC. Entre

las funcionalidades destacables de esta vista destacan la visualización de cada máquina física o virtual de forma individualizada incluyendo estadísticas acerca del uso recursos (CPU, memoria, disco duro y ancho de banda). Además, también es posible alterar manualmente la distribución de estos recursos entre las diferentes máquinas virtuales de una máquina física, o la migración de máquinas virtuales entre máquinas físicas. Finalmente, esta vista también permite migrar máquinas entre diferentes servidores físicos.

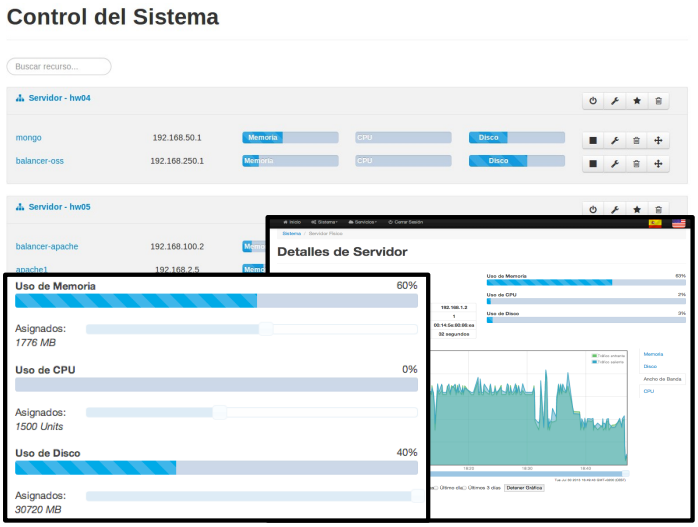

Figura 90.- Panel de control de la infraestructura en la plataforma CC (*Vista de monitorización y control*)

Por otro lado, la *vista de servicios* permite monitorizar y gestionar cada servicio de forma individualizada, así como visualizar su rendimiento en tiempo real. Se presenta una captura de pantalla de esta vista en la Figura 91.

Figura 91.- Panel de control de la infraestructura en la plataforma CC (*Vista servicios*)

B.1.2 LA CAPA PaaS (PLATFORM AS A SERVICE)

Una vez que se ha presentado la capa de *software* como servicio de la plataforma CC, la siguiente capa está compuesta por un conjunto de servicios orientados a hacer posible la elasticidad de estas aplicaciones de alto nivel. Cada componente de esta capa proveerá un conjunto servicios en términos de servicios web sin estado, para que las aplicaciones de la capa superior puedan utilizarlos como mecanismo para la persistencia de información (tanto de ficheros, como de datos). Así mismo, también proporcionará un método para la autenticación de usuarios unificado y homogéneo para todas las aplicaciones desplegas en toda la plataforma CC.

A continuación se describe los módulos principales de la plataforma CC, que son dos servicios de persistencia: objetos (B.1.2.3) y ficheros (B.1.2.2); y un servicio de gestión de la identidad que se presenta en el apartado siguiente.

B.1.2.1 SERVICIO DE GESTIÓN DE LA IDENTIDAD

Este servicio permite la autenticación de usuarios y aplicaciones para asegurar que los clientes de los servicios proporcionados por el entorno CC tienen las credenciales correctas para el acceso a los mismos, así como su nivel de privilegios en la plataforma. El servicio de gestión de la identidad implementa la funcionalidad relacionada con la autenticación de usuarios y aplicaciones, incluyendo los dos siguientes servicios principales:

- **Autenticación de usuarios** que permite una gestión de la identidad unificado a todas las aplicaciones y servicios del sistema CC, centralizando el control de las credenciales y privilegios para el acceso al sistema CC en su conjunto y a cada aplicación.
  Esta gestión incluye la asignación de roles a usuarios concretos dentro de las aplicaciones, que es un servicio que permite agregar o eliminar roles a usuarios dentro de una aplicación, previa autenticación de los mismos. Así, si por ejemplo, tenemos una aplicación que necesita un rol moderador, mediante este servicio podemos asignar a un usuario ese rol de forma automática, siempre y cuando el administrador haya configurado la existencia de ese servicio para la aplicación concreta.
- **Autenticación de aplicaciones** que permite validar las credenciales de las aplicaciones desplegadas dentro del sistema CC y así poder utilizar los servicios proporcionados por la plataforma. Para ellos, las aplicaciones disponen de un identificador y una clave que se le proporcionan al desarrollador. Cuando deseen utilizar un servicio, sus peticiones deberán ir acompañadas de estas credenciales junto con la solicitud.

Este servicio está construido haciendo uso del framework de aplicaciones Django[80] que actúan sobre una base de datos MaríaDB[81], que es la que mantiene la persistencia de la información de los usuarios, roles, aplicaciones y sesiones.

B.1.2.1.1 PROCESO DE AUTENTICACIÓN DE USUARIOS

El proceso de autenticación de usuarios se inicia cuando un visitante (usuario anónimo) intenta acceder a cualquiera de las aplicaciones del entorno CC, ya sean aplicaciones nativas, el escritorio virtual o aplicaciones desarrolladas por terceros y desplegadas en la plataforma CC. En ese momento se desencadenan un conjunto de eventos que dan lugar a la autenticación (o no) del usuario. Para este propósito el sistema CC proporciona un conjunto de pasos que hacen posible validar las credenciales del usuario no sólo en la aplicación individual, sino en todo el sistema CC.

Este proceso de autenticación, que se presenta en la Figura 92, se desarrolla tal y como sigue:

1. El usuario anónimo hace una petición a la aplicación, la cuál detecta que este usuario no ha iniciado sesión en el sistema CC.
2. La aplicación solicita un identificador de autenticación mediante el servicio web *GetAuthenticationToken*, indicando su identificador y clave de aplicación, así como la URI (*Uniform Resource Identifier*) a la que desea redirigir al usuario una vez se haya autenticado.
3. Una vez recibido el identificador, la aplicación redirige al usuario a la página de inicio de sesión en el CC, único para todo el sistema.
4. El usuario introduce sus credenciales en el formulario de inicio de sesión. El sistema CC comprueba la validez de los datos de autenticación proporcionados por el usuario.
5. Si los datos son válidos se le redirige a la página a la que intentaba acceder, acompañado de un segundo identificador.
6. El navegador del usuario hace una nueva petición, ahora con su identificador.
7. La aplicación recibe al usuario, y mediante una llamada a *TokenIsValid*, la cuál comprueba la validez del segundo identificador y recibe como respuesta los datos del usuario, sus roles, etc. La aplicación muestra entonces la página a la que el usuario deseaba acceder.

[80] https://www.djangoproject.com/
[81] https://mariadb.org/

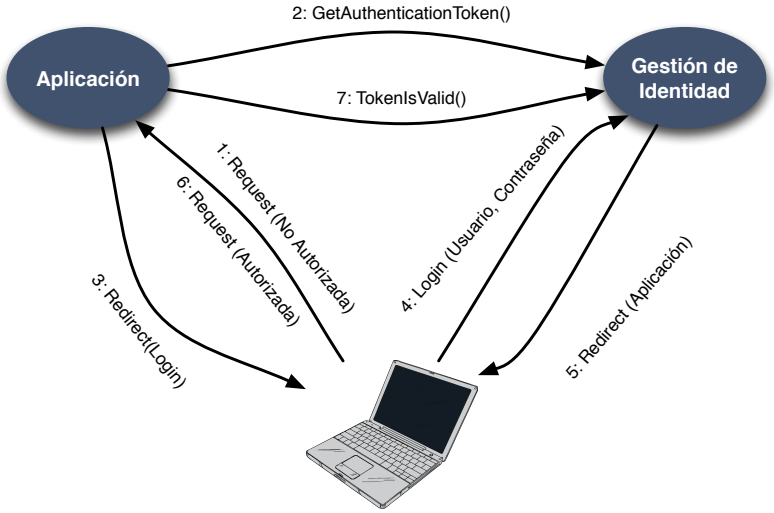

Figura 92.- Proceso de autenticación de usuarios en la plataforma CC

A pesar de que la autenticación se produce en varios pasos, es importante señalar que el proceso es interno y el usuario sólo verá la interfaz de inicio de sesión y después la interfaz a la que quería acceder en un primer momento, es decir, la de la aplicación. El resto de interacciones se produce entre la aplicación y el servicio de identidad, y las redirecciones las procesa automáticamente el navegador web del usuario.

Al igual que el inicio de sesión unificado, se puede también realizar un cierre de sesión múltiple. Si un usuario desea terminar de utilizar el entorno CC completamente, puede cerrar la sesión activa de forma centralizada con este sistema.

B.1.2.1.2 PROCESO DE AUTENTICACIÓN DE APLICACIONES

Cada aplicación dentro de la platafoma CC deberá tener un identificador único y una clave que le permiten acceder a los servicios nativos de persistencia y gestión de identidad de la plataforma. Cuando la aplicación desea utilizar un recurso la solicitud debe ser acompañada por este identificador y clave. Al igual que sucedía con la autenticación de usuarios, la plataforma proporciona un conjunto de servicios web que permiten autenticar a los usuarios.

El proceso de autenticación de aplicaciones, que se detalla en la Figura 93, es el siguiente:

1. Cuando cualquier petición llega a un servicio de la capa PaaS, lo hace acompañado de la información de autenticación de la aplicación (identificador y clave de la aplicación).

2. El servicio de la capa PaaS extrae la información de autenticación y la coteja con el sistema de gestión de identidad, mediante el servicio web *ApplicationIsValid*.
3. El servicio de gestión de identidad comprueba la información de autenticación y responde con éxito o fracaso.
4. Si la autenticación es válida, el servicio PaaS realiza la operación de la petición y devuelve la respuesta al cliente.

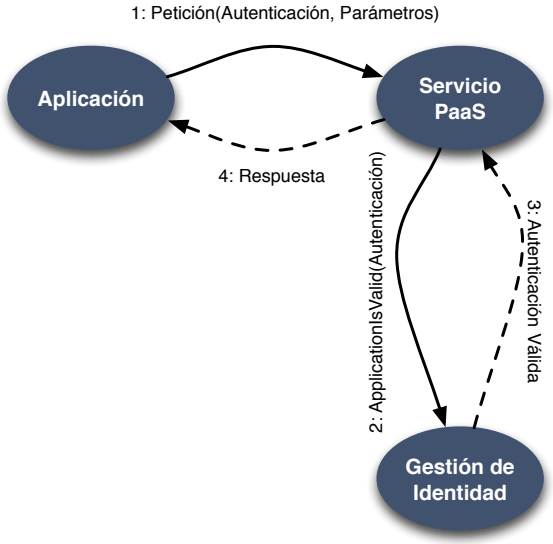

Figura 93.- Proceso de autenticación de aplicaciones en la plataforma CC

B.1.2.1.3 API DEL SERVICIO DE AUTENTICACIÓN

A continuación, en la Tabla 20, se recogen los métodos expuestos por el servicio de gestión de identidad. Estos servicios son asíncronos y están basados en al tecnología REST[82].

Método	**UserPasswordIsValid**
	Permite a una aplicación comprobar si las credenciales de un usuario son válidas.
Método	**GetAuthenticationToken**
	Obtiene un *token* temporal para la autenticación de usuarios en el entorno CC, iniciando el proceso de inicio de sesión unificado.
Método	**TokenIsValid**

[82] http://www.w3.org/TR/ws-arch/

	Permite comprobar la validez del *token* de usuario a las aplicaciones que hacen uso del mecanismo de inicio de sesión unificado.
Método	**ApplicationIsValid**
	Permite a los servicios de la capa PaaS comprobar si las credenciales de una aplicación son válidas.
Método	**ApplicationRestrictionsAccepted**
	Permite a las aplicaciones informar al entorno de que un usuario ha aceptado sus condiciones de uso.
Método	**GrantRoles**
	Permite a las aplicaciones configurar el rol de un usuario dentro de una aplicación concreta.
Método	**RevokeRoles**
	Permite a las aplicaciones revocar roles de aplicación a un usuario.

Tabla 20.- API del servicio de identidad

B.1.2.2 SERVICIO DE ALMACENAMIENTO DE FICHEROS

El servicio de almacenamiento de ficheros proporciona una interfaz a un contenedor de ficheros accesible mediante servicios web, que emula una estructura de directorios tradicional. Junto con los ficheros se almacena un conjunto de metadatos que facilitan la gestión y manipulación de los elementos individuales. Las aplicaciones pueden guardar y recuperar ficheros sin conocimiento acerca de dónde se encuentran físicamente almacenados, y por lo tanto, tan sólo tiene que preocuparse por no exceder el espacio que se haya contratado.

Sus características más relevantes son:

- Emulación de una estructura de directorios tradicional, con la que el desarrollador de *software* puede interactuar de manera similar a como lo haría con un sistema de ficheros normal.
- Un mecanismo para el control de versiones, de forma que cuando se sobrescribe un fichero, es decir, se produce la carga de un fichero a una ruta ocupada por otro, y que no se borre el contenido del anterior, sino que se genera una nueva versión. Siendo este proceso transparente al usuario. Del mismo modo en el caso de la eliminación, en lugar de borrar el contenido del fichero, se marca como eliminado. En cualquier momento se puede acceder a las versiones disponibles del fichero.
  El control de versiones puede ser desactivado mediante una llamada al servicio para aquellas aplicaciones clientes que no lo requieran.

- Gestión eficiente de los recursos, manteniendo una baja carga en el servicio tanto para la subida, como descarga de ficheros grandes.
- Almacenamiento de metadatos asociados los ficheros, lo que permite una gestión más eficiente de los ficheros.

En la Figura 94 se presenta el modelo de datos que hace posible el conjunto de características descritas para el sistema de almacenamiento de ficheros. Los nodos del almacén de ficheros pueden ser de dos tipos: directorios o ficheros. A todos ellos se les asignará un identificador único en el sistema y un elemento contenedor o padre (la raíz del sistema de ficheros virtual o cualquier otro directorio). Se guardará su nombre y se registrará su fecha de creación. En el caso de los ficheros, se guardará también su fecha de actualización.

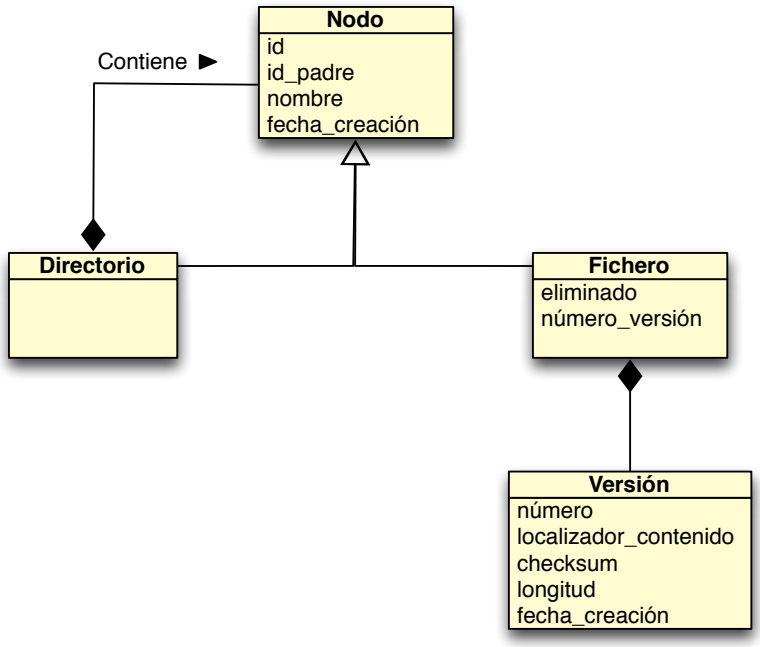

Figura 94.- Modelo de datos del almacén de ficheros en la plataforma CC

Si el sistema de control de versiones está activado, se guardará también el número de la versión actual y un atributo que determine si el fichero está eliminado (su contenido sigue accesible para ser restaurado si se desea). Para cada versión de un fichero se guardará su número, un resumen en formato *md5*[83] que permita comprobar la integridad del fichero en las transferencias y, finalmente, su longitud en bytes. Además, se guardará un localizador de

[83] http://www.ietf.org/rfc/rfc1321.txt

contenido para uso interno que permita encontrar el contenido del fichero en el sistema de almacenamiento en la capa de infraestructura.

Este servicio de almacenamiento de ficheros debe ser capaz de hacer frente con subidas y descargas de ficheros muy grandes. Para reducir la carga del servidor y mantener al mínimo el consumo de memoria, se emplea una versión modificada del proxy inverso Nginx[84] como punto de entrada al servicio. Cuando se recibe una petición de subida, el proxy almacena el contenido del fichero en su ubicación definitiva y redirige la petición, ya sin el contenido, al servicio web para que la procese. Cualquier condición de respuesta distinta a la de éxito provoca la eliminación automática del fichero.

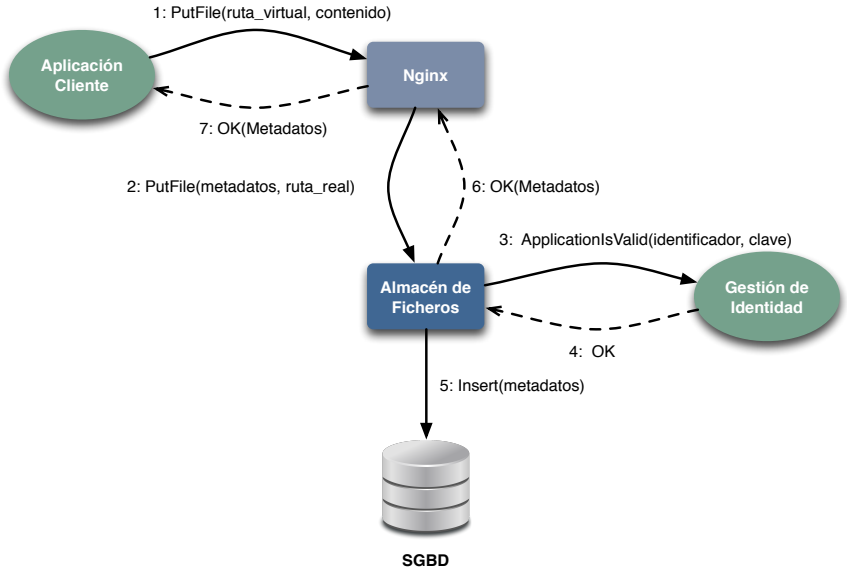

Figura 95.- Procesamiento de una llamada a *Putfile* en el servicio de persistencia de ficheros de la plataforma CC

Así, en la Figura 95, se presenta una descripción de una llamada al método *PutFile*, la cuál desencadena las siguientes acciones:

1. La aplicación cliente envía la petición al método *PutFile*, junto al contenido del fichero y la ruta virtual en la que desea almacenarse.
2. El proxy intercepta la petición y extrae el contenido del fichero de ella, colocando el contenido en su ubicación final en el sistema de ficheros distribuido sin pasos intermedios. Posteriormente, transfiere la petición al almacén de ficheros, que sólo contiene la información de

[84] http://nginx.org/

autenticación, la ruta virtual y la ruta real del contenido en el sistema de ficheros distribuidos.

3. El almacén de ficheros extrae la información de autenticación de la petición y valida estos datos contra el servicio de gestión de identidad.
4. Si el sistema de gestión de identidad valida los datos de autenticación , el almacén de ficheros guarda los metadatos generados del fichero en la base de datos distribuida y devuelve una respuesta que contiene los metadatos.
5. El proxy transfiere la respuesta a la aplicación cliente.

Por otro lado, al recibir una petición de descarga, el servicio web no devuelve el contenido del fichero, sino un comando especial al proxy indicando qué fichero debe proporcionar como respuesta a la llamada por parte del usuario del servicio. Así, tal y como se aprecia en la Figura 96, cuando se recibe una petición al método *DownloadFile*, se produce un comportamiento similar al de la llamada descrita anteriormente:

1. La aplicación cliente envía la petición al método *DownloadFile*, que contiene la ruta virtual o el identificador del fichero y la información de autenticación.
2. El proxy redirige la petición al almacén de ficheros.
3. El almacén de ficheros extrae la información de autenticación y chequea con el gestor de identidad la veracidad de los sistemas
4. Si el sistema de gestión de identidad confirma la autenticación, el almacén de ficheros recupera la información de los nodos desde la raíz de la ruta virtual hasta el nodo buscado, recuperando la ruta real de la última versión del fichero.
5. El almacén de ficheros envía una respuesta especial al proxy, indicándole la ruta del fichero que debe servir.
6. El proxy lee el fichero y lo envía a la aplicación cliente de manera eficiente.

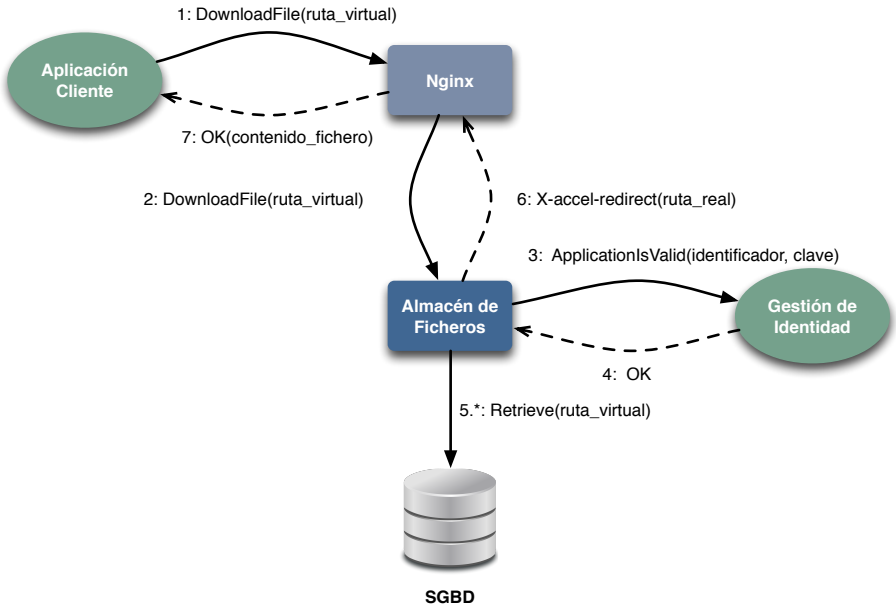

Figura 96.- Procesamiento de una llamada a *DownloadFile* en el servicio de persistencia de ficheros de la plataforma CC

B.1.2.2.1 API DEL SERVICIO DE PERSISTENCIA DE FICHEROS

A continuación, en la Tabla 21 se recogen los métodos expuestos por la capa de servicio web del almacén de objetos. Como ya se ha indicado, cada petición realizada debe ir acompañada de la información de autenticación de la aplicación que utiliza el servicio.

Método	**CreateFolder**
	Crea un directorio vacío en el dispositivo virtual.
Método	**DownloadFile**
	Recupera el contenido del fichero especificado.
Método	**GetMetadata**
	Devuelve los metadatos de un fichero o directorio.
Método	**GetAvailableVersions**
	Devuelve las versiones disponibles de un determinado fichero.
Método	**GetFolderContents**
	Recupera el contenido de un directorio
Método	**GetSize**

	Obtiene el tamaño en *bytes* de un fichero, o de la suma de tamaños del contenido de un directorio.
Método	**Copy**
	Copia un fichero o directorio de una ubicación a otra.
Método	**Move**
	Mueve un fichero o directorio de una ubicación a otra.
Método	**Delete**
	Elimina un fichero o directorio (recursivamente). En caso de que el control de versiones esté activado, los ficheros pueden ser recuperados. Si este comportamiento desea evitarse, se puede indicar la opción de borrado permanente.
Método	**DeleteVersion**
	Elimina de manera permanente una versión determinada de un fichero.
Método	**Undelete**
	Restaura la última versión de un fichero que haya sido eliminado o restaura un directorio, siempre que el control de versiones esté activado.
Método	**RestoreVersion**
	Si el control de versiones está activado, restaura una versión anterior de un fichero, configurándola como versión más reciente.
Método	**GetConfigurationParameter**
	Devuelve el valor de un parámetro de configuración del servicio. Por el momento el único parámetro contemplado es el estado (activado o desactivado) del mecanismo de control de versiones.
Método	**SetConfigurationParameter**
	Establece el valor de un parámetro de configuración del servicio.

Tabla 21.- API del servicio de persistencia de ficheros

B.1.2.3 SERVICIO DE ALMACENAMIENTO DE OBJETOS

El servicio de almacenamiento de objetos permite a las aplicaciones cliente almacenar sus datos persistentes a través de un servicio web que oculta la tecnología concreta de la base de datos utilizada para la persistencia final de la información.

Este servicio proporciona una base de datos orientada a documentos [Han *et al.*, 2011], simple y flexible. En este contexto, un documento es un objeto

diccionario, entendido como un conjunto de pares clave/valor. Las claves son cadenas cortas de texto, y son el equivalente de los nombres de columna para una base de datos relacional. En cuanto a los valores, pueden ser de tres tipos:

- Valores de tipos básicos, como números y texto.
- Valores de tipo conjunto ordenado.
- Valores de tipo documento, de forma que cualquier documento es anidable.

Estos documentos se agrupan en colecciones. Las colecciones son conjuntos de objetos agrupados semánticamente. Son el equivalente a las tablas de una base de datos relacional. Sin embargo, a diferencia de lo que ocurre en los sistemas relacionales, los documentos de una colección no están obligados a compartir un conjunto de atributos. En este sentido, es importante destacar que no hay que definir de antemano el conjunto de atributos o claves de los objetos de una colección. Para insertar un documento, bastará con enviar al servicio sus claves y valores, y el nombre de la colección en la que se desea insertar. Si la colección no existiese, se crearía en el momento. Esta flexibilidad en la definición de los documentos facilita ampliamente la migración entre versiones de una aplicación, y la definición de relaciones entre los datos.

El servicio ofrece las operaciones básicas de una base de datos sobre conjuntos de documentos: creación, eliminación, actualización y recuperación. Cuando se inserta un documento en una colección, se le asignará un identificador único, equivalente a una clave primaria. Gracias a la cuál, es posible identificar unívocamente al documento, para su selección o enlazado con otros documentos.

Cuando una petición de creación llega al servicio, se desencadena el siguiente comportamiento:

1. La aplicación cliente envía la petición de creación y el documento, acompañados de su información de creación y la colección en la que desea realizar la inserción.
2. El almacén de ficheros extrae la información de autenticación que se adjunta en la petición, y solicita al servicio de gestión de la identidad la validación de las credenciales.
3. El sistema de gestión de identidad procesa la petición y responde, indicando si la autenticación es válida o no.
4. El almacén de ficheros inserta el documento en la colección correspondiente del sistema gestor de bases de datos distribuido. Automáticamente se genera un identificador.
5. El almacén de objetos devuelve el documento insertado a la aplicación cliente, incluyendo el identificador generado.

La secuencias de pasos que se acaba de describir se presenta gráficamente en la Figura 97.

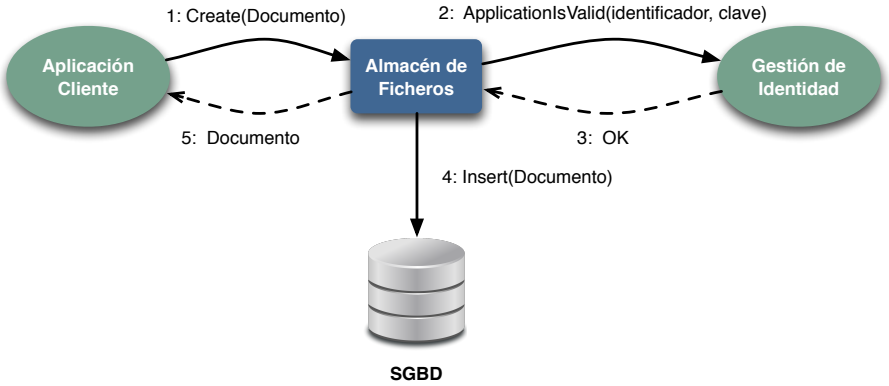

Figura 97.- Procesamiento de una petición a *Create* en el servicio de persistencia de objetos de la plataforma CC

B.1.2.4 API DEL SERVICIO DE ALMACENAMIENTO DE OBJETOS

A continuación, en la Tabla 22, se recogen los métodos expuestos por la capa de servicio web del almacén de objetos. Cada petición realizada debe ir acompañada de la información de autenticación de la aplicación que la genera.

Método	Create
	Crea un documento, o conjunto de documentos, dentro de una determinada colección, asignándoles identificadores únicos. Si la colección especificada no existe, se creará.
Método	Delete
	Elimina un documento o conjunto de documentos de acuerdo a los criterios proporcionados.
Método	Update
	Actualiza un documento o conjunto de documentos de acuerdo a unos criterios de selección y a unos criterios de actualización.
Método	Retrieve
	Recupera un conjunto de documentos de una colección.

Tabla 22.- API del servicio de persistencia de objetos

B.1.3 LA CAPA IaaS (INFRASTRUCTURE AS A SERVICE)

Aunque en menores capacidades que en los niveles descritos en los apartados anteriores, el sistema CC desarrollado también permite ofertar servicios de infraestructura en términos de VPS (*Virtual Private Server*)

[Beloussov *et al.*, 2008]. Este tipo de productos están orientados a que los usuarios de la plataforma puedan disponer de un servicio de *hosting* básico, que les permita el despliegue de sus aplicaciones en la propia plataforma, lo más próximo posible al resto de servicios de la misma.

En la versión actual del sistema CC desarrollado, las capacidades de configuración e interacción de los usuarios con los productos ofertados en esta capa son mínimas. Ya que, hoy en día, tan sólo se ofertan un pequeño conjunto de máquinas virtuales preconfiguradas, que se instancian en base a plantillas. El usuario puede utilizar una de estas plantillas para el despliegue de sus aplicaciones web, pero en ningún caso podrán ser utilizadas para otros fines, como puede ser la computación de alto rendimiento o la persistencia de grandes volúmenes de información.

En definitiva, este producto aunque tiene limitaciones, hace posible el despliegue de aplicaciones de terceros en el mismo entorno de computación. Las aplicaciones web desplegadas en esta capa, pasan a formar parte de la capa SaaS del sistema CC, integrándose con las aplicaciones nativas y el escritorio virtual. Para garantizar su elasticidad tienen que seguir un modelo arquitectónico similar al de las aplicaciones nativas, almacenando su estado e información que en los servicios de persistencia que proporciona la plataforma y que ya se han descrito en los apartados B.1.2.1, B.1.2.3 y B.1.2.3.

B.2 LOS COMPONENTES INTERNOS

Un entorno CC es uno de los entornos más complejos dentro del panorama tecnológico actual. En él se encuentran una gran variedad de sistemas *hardware* y *software* altamente heterogéneos. Además, la amplia mayoría de estos componentes presentan características heterogéneas. Por ello, el desarrollo de una plataforma de tipo CC es una tarea compleja, ya que exige que todos los componentes trabajen de forma coordinada en un entorno altamente impredecible. Además, el usuario final no tiene por qué ser consciente de la complejidad subyacente. Por lo que la plataforma debe ser capaz de trabajar aislando su complejidad intrínseca, así como las características del entorno en el que se despliegan.

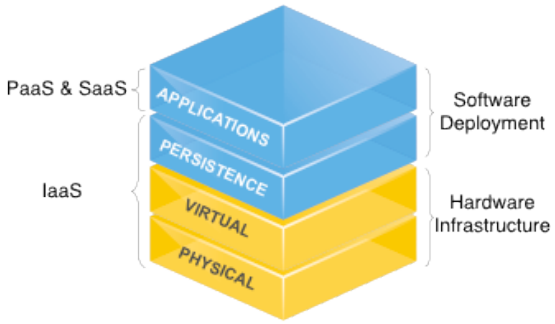

Figura 98.- Estratos de un entorno Cloud Computing

Así en la Figura 98 se presenta a muy alto nivel la pila tecnológica de cualquier entorno CC. Como se puede apreciar, se subdivide en dos grandes grupos, la capa de despliegue *software* y la infrastructura *hardware*, tal y como sigue:

- La capa de despliegue *software* se subdivide a su vez en la capa de aplicaciones incluyendo las aplicaciones alojadas en la capa PaaS y SaaS. Y, la capa de persistencia que engloba las bases de datos que hacen que la información alojada en el CC sea persistente.
- La capa de infraestructura que a su vez incluye los componentes físicos, es decir, el *hardware* real del entorno CC y una una capa de virtualización a nivel superior que contiene un conjunto de abstracciones de estos recursos físicos, denominadas máquinas virtuales que permiten una gestión dinámica de esta capa. Gracias a este modelo de computación, se consigue una visión externa de recursos físicos ilimitados, pese a que los recursos *hardware* son siempre limitados.

En este apartado se presentarán los principales componentes tecnológicos de la plataforma CC, empezando con los sistemas que proporcionan la persistencia de datos en el apartado siguiente. Posteriormente, se describirán

otros subsistemas que permiten el balanceo de carga, la comunicación y la gestión dinámica del entorno de virtualización. Finalmente, se hará una breve descripción de la estructura del sistema de control y monitorización del entorno CC.

B.2.1 LA CAPA DE PERSISTENCIA

La capa de persistencia de la plataforma se utiliza para guardar el estado y la información de las aplicaciones a nivel SaaS y PaaS, asegurando de este modo que los algoritmos de elasticidad propuestos en el marco de esta investigación no producen inconsistencias en los datos. Con tal fin, esta capa tiene dos componentes principales, el primero de ellos orientado a la persistencia de información en base de datos, que se asocia al servicio de almacenamiento de objetos descrito en el apartado B.1.2.3. Por su parte, el segundo componente es un sistema de ficheros distribuido utilizado por el servicio de almacenameinto de ficheros descrito en el apartado B.1.2.2.

B.2.1.1 LA PERSISTENCIA DE INFORMACIÓN

El paradigma relacional ha sido el más popular en las últimas décadas, pero presenta problemas para la escalabilidad horizontal; es decir, para incrementar la capacidad de respuesta del sistema mediante la replicación de proveedores del servicio. Estos problemas son derivados en gran medida por las restricciones impuestas por medio de relaciones basadas en identificadores entre las diferentes tablas de una base de datos. En los últimos tiempos, debido a la proliferación de aplicaciones orientadas a Internet de uso masivo, los problemas de escalabilidad del paradigma relacional han empezado a ser evidentes [Strauch *et al.*, 2011]. Los desarrolladores de aplicaciones deben recurrir a la separación de datos en varios servidores que trabajen en paralelo, siendo las propias aplicaciones las encargadas de descubrir qué servidor posee la información a la que desean acceder. Este método es en sí mismo es un factor limitante, tanto para la escalabilidad como para el aislamiento de las capas que forman la aplicación.

Además de estos problemas de escalabilidad, el modelo relacional adolece de excesiva rigidez. Las aplicaciones orientadas a Internet están en constante evolución, y esa evolución tiene un efecto directo sobre su modelo de datos. Las bases de datos relacionales requiere que sea necesario conocer de antemano la estructura de la información que deben almacenar, siendo necesario reconstruir la base de datos con cada cambio en el modelo de información. A esto se le une la dificultad de hacer coexistir varios modelos de datos en la fase de transición entre versiones de la aplicación.

Debido a estas limitaciones, en los últimos años han aparecido una gran cantidad de propuestas de modelos alternativos al relacional [Han *et al.*, 2011],

tanto desde el ámbito académico como desde el ámbito empresarial. Estos nuevos modelos, caracterizados por un enfoque más relajado en cuanto a las relaciones entre los datos, y unas restricciones de integridad más laxas, han sido aglutinadas bajo la etiqueta de bases de datos *NoSQL* [Strauch *et al.*, 2011].

El modelo de base de datos que más éxito ha cosechado dentro de las alternativas al modelo relacional es el orientado a documentos. En este modelo, la unidad fundamental de información es el documento, que está constituido por una colección de pares clave-valor. Comparado con el modelo relacional, los documentos corresponderían a las tuplas y las claves a las columnas de una tabla. Los valores asignados a las claves se corresponden a los valores de los campos de la tupla. Los documentos son anidables, es decir, un valor para una clave de un documento puede ser otro documento; y se agrupan en colecciones. Las colecciones son agrupaciones semánticas de documentos, al igual que en el modelo relacional las tuplas se agrupan en tablas. No obstante, no se observan más semejanzas, ya que los documentos de una colección no están obligados a compartir un conjunto de atributos, aunque en la práctica si que suelen hacerlo. En cualquier momento se puede insertar e integrar en una colección un documento con un atributo del que no dispongan los demás, sin perjudicar el rendimiento del sistema. Así mismo también se puede añadir nuevos atributos a voluntad a los documentos ya creados.

Con el modelo de documentos, el mapeo de objetos *software* a base de datos resulta trivial. Basta con crear un documento por objeto, con claves y valores obtenidas de los atributos del objeto. Las relaciones pueden representarse mediante referencias, o mediante anidamiento. En este sentido, cabe destacar que los documentos suelen tener toda la información que se precisa en las búsquedas, bien como atributos, o bien como documentos anidados. Con el fin de garantizar la distribución de las colecciones a través de los nodos de la base de datos, los sistemas de este tipo no suelen soportar sentencias de unión (*join*).

Para la persistencia de objetos en la plataforma CC, se utiliza la base de datos orientadas a documentos MongoDB[85], que es un sistema gestor de bases de datos de código abierto orientado a documentos, distribuido y escalable. La orientación a documentos constituye un modelo de base de datos distinto al relacional, con su propio conjunto de abstracciones y restricciones.

[85] https://www.mongodb.org/

MongoDB es uno de las bases de datos orientadas a documentos con mayor éxito [86]. Es un proyecto que se encuentra en constante desarrollo, con una gran comunidad de usuarios, y que está siendo utilizado en una gran cantidad de proyectos de gran envergadura por empresas que van desde Disney[87] hasta EA Sports[88], pasando SourceForge[89], y proveedores de PaaS como SAP[90].

Para conseguir la escalabilidad y la alta disponibilidad de las bases de datos MongoDB, se emplean dos mecanismos distintos, pero complementarios:

- La **replicación de la información**, en el que se definen varios servidores o nodos que forman parte de un conjunto de replicación o *ReplicaSet*. Dentro del conjunto de replicación, los servidores escogen un primario mediante votación. El servidor primario debe ser el objetivo de las operaciones de escritura. Por su parte, las operaciones de lectura se pueden hacer a cualquier servidor del conjunto de replicación. Cuando un servidor primario deja de funcionar, se produce una reelección de primario de manera transparente al usuario.
- El ***sharding* o partición de la información**, en el que la información se divide utilizando funciones de dispersión, entre varios nodos. Cuando este mecanismo se utiliza en combinación con el de replicación, la información se divide entre conjuntos de réplicas. De esta forma, la carga computacional y de red se divide de manera uniformemente entre estos conjuntos.

Un servicio especial se encarga de encaminar las peticiones al nodo correcto, por lo que las aplicaciones no necesitan tener conocimineto en ningún momento de la opción de escalabilidad elegida. En MongoDB, estas máquinas reciben el nombre de *routers*.

B.2.1.2 LA PERSISTENCIA DE FICHEROS

Los sistemas de ficheros tradicionales no pueden cumplir con las requerimientos de los grandes sistemas que cuentan con una gran cantidad de datos. En este sentido es necesario emplear un sistema de ficheros distribuido que permitan la replicación y la partición de datos [Clifford *et al.*, 2007]. En un entorno CC, todos los recursos virtuales del usuario comparten un mismo área de almacenamiento, utilizando un sistema que se conoce como un entorno de almacenamiento virtual, implementado en forma de SAN (*Storage Area Network*).

[86] http://lineofthought.com/tools/mongodb
[87] http://disney.com/
[88] http://www.easports.com/
[89] http://sourceforge.net/
[90] http://www.sap.com/

La replicación de datos en diferentes nodos *hardware* y diferentes discos permite que el acceso a la información sea más rápido, y que además, el servicio no se interrumpa por fallos en un nodo, lo que permite que no se pierda información en el caso de que un disco deje de funcionar. La partición de datos aumenta la velocidad de lectura y escritura de información, y permite aumentar la capacidad del sistema mediante la adicción de nuevos nodos *hardware*.

Como entorno de almacenamiento en red, existen diferentes opciones, quizás la más conocida es NFS (*Network File System*)[91] que es un sistema de almacenamiento de ficheros distribuido. No obstante, la plataforma utiliza GlusterFS [92] debido a que ofrece mejores características en cuanto a escalabilidad y a las mayores posibilidad de configuración y readaptación. GlusterFS es un sistema de archivos escalable para NAS (*Network Attached Storage*). Se puede encontrar siendo utilizado en una gran variedad de ambientes y aplicaciones como computación en nube, ciencias biomédicas y almacenamiento de archivos [Noronha *et al.*, 2008] [Písačka *et al.*, 2012]. GlusterFS está licenciado bajo GPL (*General Public License*).

Su gestión de la infraestructura de persistencia se basa en la creación de volúmenes de almacenamiento virtuales que pueden ser montados desde otro dispositivo. Estos volúmenes pueden ser distribuidos o replicados, tal y como se muestra en la Figura 99, según las necesidades que se tengan:

- Un **volumen distribuido** es cuando dos o más nodos de almacenamiento de ficheros, cada uno con su información, integran un único volumen distribuido, que el cliente ve como un todo. Gracias a este modelo se proporciona gran escalabilidad, ya que es posible agregar servidores en caliente según se requiera.
- Un **volumen replicado,** se tiene cuando varios nodos tienen réplicas de un único volumen distribuido. Gracias a este modelo es posible asegurar la disponibilidad de los datos.

GlusterFS permite configurar el número de servidores, réplicas e incluso fraccionar un fichero y distribuirlo en varios servidores para ganar en velocidad. Además se pueden combinar los tipos de volúmenes como se desee.

[91] http://nfs.sourceforge.net/
[92] http://www.gluster.org/

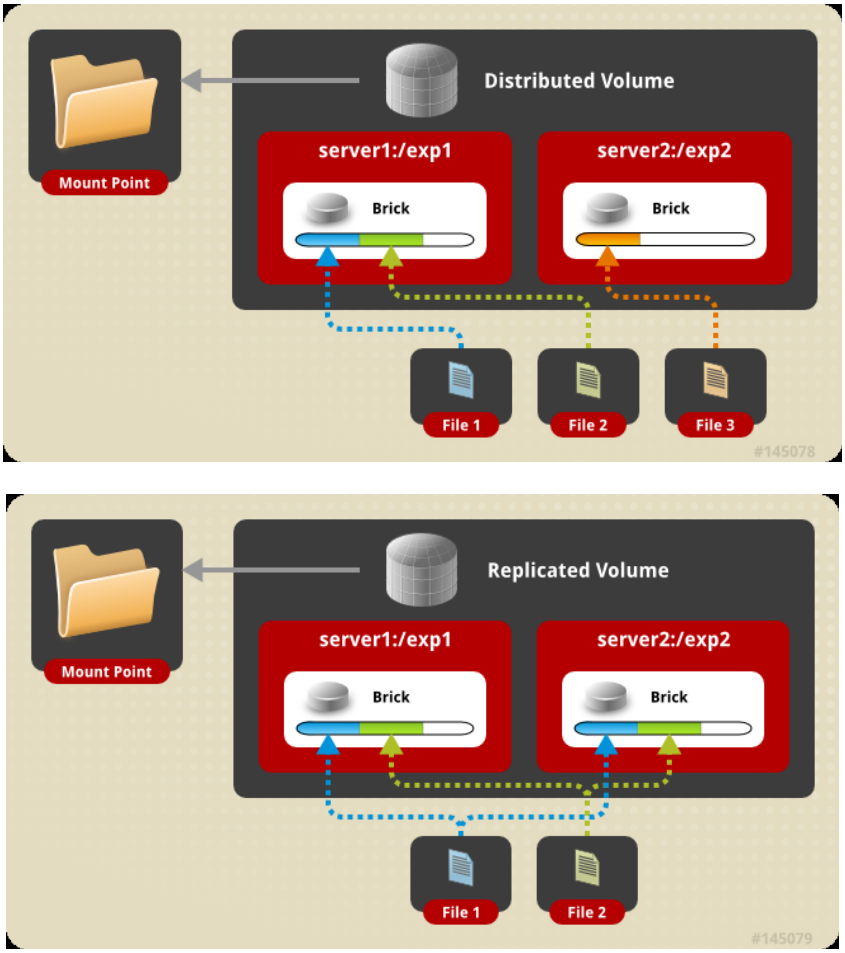

Figura 99.- Volúmenes de almacenamiento virtual en GlusterFS (Arriba: Volúmen distribuido; Abajo: Volúmen replicado)

B.2.2 EL BALANCEO DE CARGA

Un *proxy* inverso es un componente *hardware* o *software* que recibe peticiones de clientes y las delega a otros servidores, los proveedores reales del servicio. Los *proxies* se utilizan normalmente para filtrar peticiones, registrar históricos o transformar en las peticiones que pueden incluir modificaciones de la ruta de acceso al servicio, encriptación o desencriptación, modificaciones de los encabezados del protocolo de aplicación, etc.

Los balanceadores de carga pueden ser considerados un tipo especial de *proxy*. Además de modificar las peticiones de los clientes, pueden distribuir las peticiones que llegan al servidor entre un conjunto de proveedores de

servicios, proporcionando además un punto de acceso unificado a los conjuntos de servidores. De esta manera, los clientes no tienen que conocer el número, ni la ubicación de proveedores del servicio, sólo deben conocer el nombre de red del punto de entrada al sistema, es decir, del *proxy*. El *proxy* que sí conoce esta información, será el encargado de redirigir la petición a uno de los servidores reales que atienden el servicio. De manera interna, el balanceador puede utilizar diferentes algoritmos para elegir a qué servidor encaminar la petición.

En la plataforma CC, como implementación del proxy y balanceador de carga se ha optado por Nginx[93], un proyecto de *software* libre con un amplio espectro de utilización, y que se adaptaba perfectamente a las necesidades del proyecto:

- Un sistema de registro de estado completo, que permita la inspección constante de los ficheros de registro para obtener información relativa a la demanda de los distintos servicios que provee la plataforma.
- Un sistema de configuración basada en ficheros con una sintaxis clara, que pueda ser generada automáticamente por los agentes que gobiernan la plataforma.
- Configuración actualizable en tiempo de ejecución, de forma que pueda ser alterada sin que se pierdan peticiones. Este es un aspecto imprescindible de cara a los mecanismos de elasticidad de la plataforma, ya que debe permitirse la adición o eliminación de nodos asociados a un servicio concreto sin que los clientes adviertan problemas en la disponibilidad.
- Balanceo de carga basado en *Round-Robin* que admite ponderación.
- Soporte para HTTPS (*Hypertext Transfer Protocol Secure*), de manera que el cifrado se produce sólo entre el proxy y los clientes, lo que permite reducir la carga computacional de los servidores de aplicación, que ya no deben encargarse de descifrar las peticiones encriptadas.

En la plataforma, para el balanceo de carga se ha optado por una variación ponderada del algoritmo *Round-Robin*, de manera que las peticiones se entreguen a los servidores alternativamente (siguiendo un orden de lista circular), pero tratando de mantener el nivel de calidad contstante en todos los nodos o grupos de nodos. Además, la arquitectura basada en *plug-ins* de Nginx permite que sea fácilmente expandible si surgiera una necesidad no contemplada en el diseño inicial del sistema.

[93] http://nginx.org/

B.2.3 SISTEMA DE COMUNICACIÓN

Este componente ofrece el soporte necesario para que los diferentes agentes que forman parte de las organizaciones del sistema ofrezcan y descubran servicios, así como para permitir la comunicación entre cada uno de los componentes individuales. En este sentido, este componente proporciona un lugar donde las entidades autónomas pueden registrar la descripción de servicios como entradas en un directorio y suscribirse a los canales de comunicación que ofrecen el resto de integrantes del SMA.

Dentro de la plataforma, el sistema de comunicación está formado por una plataforma de mensajería de alta disponibilidad basada en colas de mensajes e implementada a través de la plataforma RabbitMQ[94]. A través de esta plataforma se ofrece también un servicio de información que permite a los agentes descubrir los servicios de sus iguales. En un SMA clásico, las locuciones se realizan entre pares 1 a 1, a través de un sistema de comunicación. No obstante, en un SMA, basado en organizaciones es posible definir comunicaciones más complejas, tal y como sigue:

- Comunicaciones 1 a 1, que son comunicaciones entre agentes a título individual.
- Comunicaciones 1 a N que son comunicaciones de 1 a un conjunto de agentes.
- Comunicaciones *multicast*, orientadas a la comunicación entre todos los miembros de una organización de agentes.

Dentro del sistema de paso de mensajes basado en colas se destaca que se ha hecho posible la alta disponibilidad del mismo ya que está replicado en varios servidores que forman parte de la infaestructura del entorno CC. Para ello se ha hecho uso de las bondades que ofrece RabbitMQ. El SMA será el encargado de comprobar que este módulo está funcionando correctamente en todo momento. Sino lo estuviera, se instancia una réplica del servicio en un nuevo servidor físico y se determinará las causas que produjo la no disponibilidad del mismo.

B.2.4 EL ENTORNO DE VIRTUALIZACIÓN

La virtualización [Barham *et al.*, 2003] es uno de los denominadores comunes de las plataformas CC. Consiste, a grandes rasgos, en la emulación de un entorno *hardware* para un sistema operativo, en este caso virtualizado. De forma que el sistema operativo, en general, no conoce que está siendo ejecutado en un instancia virtual, denominada máquina virtual. En la provisión

[94] https://www.rabbitmq.com/

de servicios basados en CC, el aporte de la virtualización es especialmente importante debido a tres aspectos principales: la simplificación del mantenimiento de los servidores, el soporte para la elasticidad, y el soporte para seguridad.

Gracias a la virtualización, es posible que en un único servidor físico (anfitrión), pueden existir múltiples servidores virtuales (huéspedes), cada uno de ellos con su propia identidad, compartiendo de manera transparente los recursos del servidor físico. Los sistemas de virtualización modernos se agrupan en dos categorías [Crosby *et* Brown, 2006], los sistemas de virtualización completa y los de paravirtualización. En primer lugar, los sistemas de *virtualización completa* emulan un entorno *hardware* completo, de manera que el sistema operativo huésped no tiene conocimiento de estar ejecutándose en una plataforma virtual. Esta aproximación permite que los sistemas operativos virtualizados sean distintos al sistema huésped, y puedan ser ejecutados sin modificación alguna. Sin embargo, simular múltiples computadores completos dentro de otro computador lleva asociada una gran pérdida de rendimiento. Por otro lado, los sistemas de *paravirtualización* requieren que el sistema operativo anfitrión y los sistemas huésped sean el mismo o, al menos, compartan ciertas semejanzas. Los sistemas huéspedes deben ser modificados y deben ejecutar un *kernel* especial. El *kernel* modificado del huésped ya no interactúa con un entorno *hardware* emulado, sino directamente con el *kernel* de la máquina anfitrión. La pérdida de rendimiento provocada por la emulación del *hardware* se reduce en gran medida [Youseff *et al.*, 2006] [Habib, 2008].

Las características principales que ofrece la virtualización son:

- **Despliegue basado en plantillas.** Las máquinas virtuales pueden ser instanciadas a partir de una serie de plantillas preconfiguradas con los servicios que se desean incluir. Esto elimina necesidades de configuración manual y facilita la escalabilidad horizontal del sistema.
- **Facilidad para añadir nuevo *hardware*.** Los nuevos servidores físicos sólo deben ser preconfigurados con el entorno de virtualización e incluidos en el sistema de despliegue. Los servicios se añadirán a partir de plantillas de máquina virtual.
- **Optimización del *hardware* disponible manteniendo el aislamiento entre servicios.** Con frecuencia se busca desplegar cada servicio en máquinas separadas. Así, la virtualización permite que los servicios permanezcan aislados en máquinas virtuales mientras se reparten los recursos físicos del servidor anfitrión. De esta manera se puede aumentar el nivel de uso de los procesadores o de la memoria de los servidores físicos, ya que un único servicio no suele emplear todas las prestaciones.

- **Minimización de los tiempos de *downtime*.** La posibilidad de migrar máquinas virtuales sin cortes en el servicio permite realizar el mantenimiento de los servidores físicos sin que lo perciba el cliente.

Así mismo, también aporta un conjunto de funcionalidades que hace posible la elasticidad de los servicios:

- **Reubicación en vivo de las máquinas virtuales.** El servicio de elasticidad puede ejecutar algoritmos de razonamiento que decidan la manera más óptima de reagrupar las máquinas virtuales en servidores físicos atendiendo a distintos criterios.
- **Reasignación de recursos entre servicios.** Además de reubicar las máquinas, los algoritmos de razonamiento pueden variar en tiempo real la cantidad de potencia de cómputo, de memoria o de disco que recibe un determinado servicio, contribuyendo a la optimización de las calidades de los servicios. De esta manera ya no hay que predecir las cargas de los servicios de antemano para diseñar el despliegue, sino que el despliegue se adapte a las condiciones de uso en cada momento.

En cuanto a la utilización de una solución de virtualización para el desarrollo de la plataforma CC, se ha optado por el uso de entornos con licencias libres, utilizando finalmente dos plataformas OpenVZ[95] y KVM[96]. En una primera fase del desarrollo se utilizó el sistema el sistema de virtualización OpenVZ que está basado en contenedores y por lo tanto restringe su funcionamiento a sistemas Linux. Como contrapunto a esa rigidez, el rendimiento de OpenVZ es ligeramente mayor que otros sistemas de virtualización. Sin embargo, el gran problema de OpenVZ reside en el débil aislamiento de las máquinas, que como se ha indicado está basado en contenedores, y habitualmente provoca que los fallos en las máquinas virtuales se propaguen con facilidad a la máquina anfitrión

Por este motivo, se migró a KVM que es un sistema de virtualización basado en Qemu[97] [Bellard, 2005]. En lugar de proporcionar una capa de abstracción hacia el *hardware*, proporciona al *kernel* Linux una serie de infraestructuras orientadas a la virtualización. También permite la migración en vivo de máquinas virtuales entre sistemas anfitrión, pero mediante el uso de un sistema de almacenamiento compartido. La asignación de memoria RAM es menos rígida que en otros entornos de virtualización y funciona correctamente

[95] http://openvz.org/
[96] http://www.linux-kvm.org/
[97] http://wiki.qemu.org/

con versiones estables (2.6) del *kernel* de Linux. Para su administración se utiliza la librería Libvirt[98] que facilita en gran medida el trabajo.

B.2.5 EL SISTEMA DE CONTROL

Un entorno tecnológico basado en el paradigma CC está caracterizado por su heterogeneidad, el dinamismo y la incertidumbre. Es (i) heterogéneo, ya que puede incluir diversos elementos y componentes muy diversos entre sí. Es (ii) dinámico, ya que la demanda de sus servicios varía a lo largo del tiempo, y además, el funcionamiento de sus componentes tiene que adaptarse a los nuevos estados que puedan darse en función de la demanda de los usuarios. Así mismo se puede dar el caso, y de hecho es habitual que nuevos componentes se añadan a un entorno CC, o que se eliminen algunos de los ya existentes sin que se altere el funcionamiento normal de la plataforma. Finalmente, también se caracteriza por la (iii) incertidumbre del contexto, ya que cada uno de sus componentes conoce tan sólo un subconjunto del mismo; teniendo que tomar decisiones con la información parcial que está disponible en cada momento.

Para el desarrollo de la plataforma CC que integra el SMA denominado +Cloud, se opta por el uso de un modelo de diseño innovador, basado en el uso de OV de agentes inteligentes. Este modelo de organizativo se ha estructurado estratos diferentes. El objetivo es crear un diseño que pueda ser adaptado al incesante avance de la tecnología, o incluso, a las nuevas líneas de investigación que puedan surgir en el futuro. Así, tal y como se aprecia en la Figura 100, el sistema se divide a nivel conceptual tal y como sigue:

- **Nivel organizacional.** En este nivel, que es abstracto, se incluyen un conjunto de roles de alto nivel que describen las tareas y cómo se produce la interacción entre ellos, los grupos que se pueden crear y las normas que hay que cumplir.
- **Nivel agentivo.** En este nivel es donde se definen los agentes individuales, cada uno tiene un comportamiento autónomo, tomando decisiones y actuando en el entorno en función de la información que recibe a través de sus sensores o puertos del entorno. Cada agente está caracterizado por unas creencias, deseos, objetivos, intenciones y planes, así como, el modelo de razonamiento especializado en función del rol que juegue en la organización.
- **Nivel de entorno.** Engloba a un conjunto de componentes que los agentes de la capa superior pueden utilizar, aunque su comportamiento

[98] http://libvirt.org/

no es autónomo. Todos estos componentes pueden tomar información sobre el entorno, o actuar sobre él, pero no tienen capacidad para tomar decisiones sobre sus acciones. En este sentido, será el nivel superior, el de los agentes, el encargado de tomar las decisiones adecuadas.

- **Nivel externo.** En el que tan sólo se trata de describir cómo y dónde se sitúan los agentes en cada uno de los recursos físicos existentes.

Utilizando este conjunto de niveles es posible diseñar la organización de forma unificada, intuitiva y con un alto nivel de abstracción. Gracias a ello se facilita y se simplifica el proceso de diseño de sistemas, obteniendo modelos de OV que pueden ser implementados en diferentes plataformas y con diferentes tecnologías.

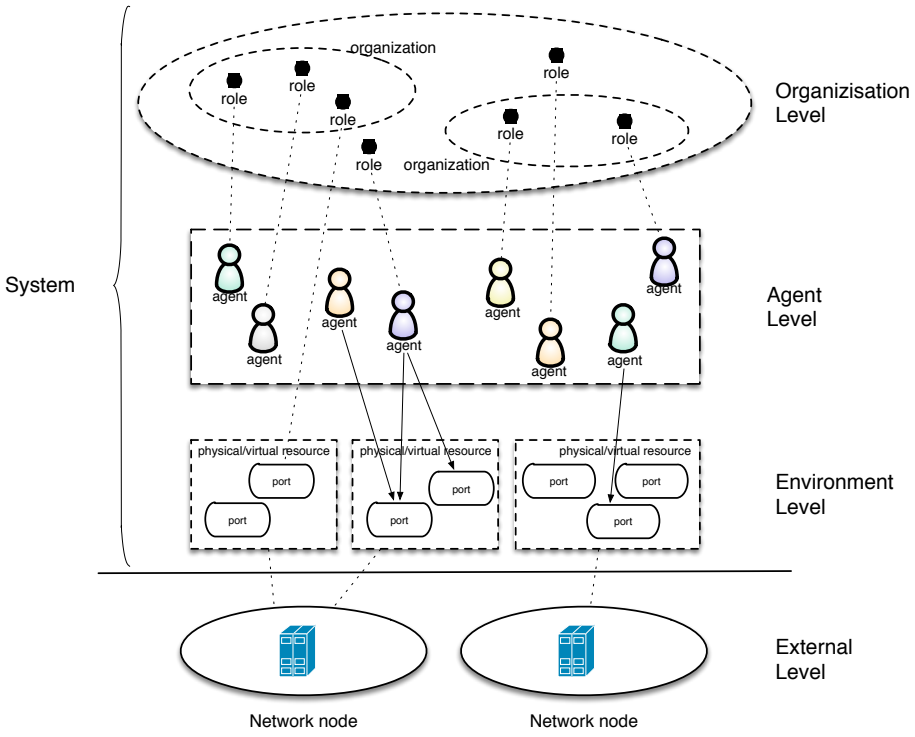

Figura 100.- Niveles del sistema multiagente de +Cloud

ANEXO C ESTADO DEL ARTE DE LAS TECNOLOGÍAS RELACIONADAS

En el aspecto tecnológico, la computación en nube constituye una nueva manera de construir aplicaciones orientadas a Internet y enfocadas a un número masivo de usuarios. El enfoque tecnológico del CC, es el de construir servicios (ya sean aplicaciones, servicios de *hosting* o servicios de almacenamiento) que puedan ofrecerse a un gran número de usuarios empleando el mínimo volumen de recursos hardware [Buyya *et al.*, 2010b]. Además, como ya se ha venido comentando a lo largo de todo este trabajo, es posible incrementar la capacidad o calidad del servicio de manera sencilla al aumentar los recursos computacionales disponibles para cada uno de ellos. Esta característica se denomina elasticidad de un servicio [Chiu, 2010].

Prácticamente la totalidad de los proveedores [Dewan *et* Hansdah, 2011] de soluciones informáticas se han sumado al cambio, convirtiéndose en proveedores y/o consumidores de servicios de CC. Este nuevo paradigma permite a las empresas dar servicio a un mayor número de usuarios, optimizando la utilización tanto de los recursos computacionales de los que disponen, como de los que son subcontratados a otras empresas.

Para los usuarios finales la transición ha sido más progresiva. Desde los comienzos de Internet el concepto de sitio web ha evolucionado de manera constante [Berners-Lee, 1996]. El aumento de la capacidad de procesamiento de los servidores y de los navegadores de los usuarios, unido a la introducción de lenguajes de programación del lado del cliente, motivó un cambio hacia una interactividad mayor entre el usuario y el sitio web. Las páginas ya no eran estáticas, el usuario podía tener un efecto sobre el sitio, y la manera de interactuar con él se iba pareciendo cada vez más a la manera de interactuar con las aplicaciones de escritorio. El incremento de la interactividad entre el usuario y el sitio desembocó en un cambio en la manera de utilizar Internet. Las aplicaciones web [Fraternalli *et al.*, 2010] permitían a los usuarios delegar la carga computacional y el almacenamiento de los datos a los servidores del proveedor de aplicación, y su ordenador personal sólo debía encargarse de mostrar la información.

En definitiva, el usuario de servicios CC percibe una capacidad de cómputo y almacenamiento aparentemente ilimitadas. Los proveedores de servicios gratuitos establecen cuotas de proceso y almacenamiento para cada usuario. Los proveedores de servicios de pago tarifican en función de los recursos empleados por los usuarios. Para hacer posible este modelo de comercialización donde los productos se fabrican a medida, no cabe duda de

que la tecnología que da soporte al paradigma CC ha tenido un papel fundamental.

Este apartado tiene como objetivo presentar un breve análisis del panorama CC actual, incluyendo proveedores, plataformas y tecnologías. Así pues, en el apartado siguiente (C.1) se presentan los principales proveedores CC existentes en la actualidad, los cuáles están amparados por las grandes compañías tecnológicas (Amazon Google y Microsoft, entre otras). A continuación en el apartado C.2 se presentan brevemente las plataformas CC existentes en la actualidad, recordar que el apartado 2.4 del Capítulo 2 incluía una completa revisión de las más importantes. Finalmente, el último apartado de este anexo (C.3) incluye una breve revisión a nivel tecnológico de los sistemas CC.

C.1 PROVEEDORES CLOUD COMPUTING

A lo largo de este apartado se presentará una descripción detallada y el catálogo de servicios que ofertan los principales proveedores de tecnología CC en la actualidad, es decir, Amazon, Google y Microsoft.

Posteriormente se hará una breve relación de algunos otros proveedores de menor tamaño. Se hará especial énfasis en los proveedores que dispongan al menos de servicios PaaS, ya que el número de proveedores de *software* como servicio es prácticamente ilimitado y su naturaleza muy heterogénea. En el estado del arte pueden encontrarse comparativas exhaustivas de los distintos proveedores de CC [Fisher *et al.*, 2010] [Rimal *et al.*, 2009] [Lenk *et al.*, 2009] [Lou *et al.*, 2011].

C.1.1 AMAZON WEB SERVICES

Amazon es, junto a Google, uno de los pioneros en la provisión de servicios basados en el paradigma CC, a través de su propducto *Amazon Web Services* (AWS)[99].

Al igual que otros proveedores, se ha especializado en los servicios relacionados con las capas de infraestructura. Su producto principal es *Amazon Elastic Compute Cloud (*EC2)[100]. El cuál es similar a un servicio de *hosting* privado virtual, pero que permite reservar y liberar recursos bajo demanda del usuario. Esta reserva o liberación de recursos de infraestructura se produce de manera automática deacuerdo a las necesidades de los usuarios. Su tecnología se basa en la virtualización de toda la pila tecnológica ofrecida al cliente, desde el sistema operativo a los discos duros. Para ello, utilizan una versión modificada del entorno de virtualización Xen[101] [Goth, 2007].

Los recursos sujetos a elasticidad abarcan desde la potencia de proceso y la memoria, al espacio de disco. Incluso es posible la elasticidad de nuevas instancias de máquinas virtuales siempre y cuando las aplicaciones de los clientes pueden escalar de manera horizontal. El servicio de *hosting* elástico se complementa con un sistema de configuración fácil de cortafuegos y balanceadores de carga.

[99] https://aws.amazon.com
[100] https://aws.amazon.com/es/ec2/
[101] http://www.xenproject.org/

Además del servicio de infraestructura, Amazon también ofrece un catálogo de servicios a nivel PaaS. Así, oferta una serie de productos disponibles en forma de servicio web que permiten construir aplicaciones orientadas a Internet, garantizando su escalabilidad en el ecosistema de Amazon. Estos productos van desde sistemas de bases de datos distribuidas (relacionales y no relacionales) a sistemas de coordinación de procesos basados en algoritmos tipo *Map Reduce* [Dean *et* Ghemawat, 2008]. A continuación en la Tabla 23 se presenta un listado de los principales productos ofertados en AWS.

Servicio	Nivel	Descripción
Amazon Elastic Compute Cloud EC2	IaaS	Servicio de hosting virtual con provisión elástica de recursos.
Amazon Elastic MapReduce	PaaS	Procesamiento de grandes volúmenes de datos.
Auto Scaling	IaaS	Escalado automático basado en uso, con límites máximos y mínimos, de los servicios.
Elastic Load Balancing	IaaS	Configuración semiautomática del balanceo de carga entre instancias de máquinas virtuales
Amazon Route 53	IaaS	Servicio DNS (*Domain name server*) de alta disponibilidad.
Amazon Virtual Private Cloud	IaaS	Segmentación privada de la infraestructura de Amazon para el uso empresarial. Permite mayor detalle en la configuración de la red.
AWS Direct Connect	IaaS	Establecimiento de una VLAN (*Virtual Local Area Network*) entre una organización y la infraestructura.
Amazon Cloud Front	IaaS	Servicio de entrega de contenido web de baja latencia.
Amazon Flexible Payments Service	PaaS	Servicio de transferencia automatizada de dinero entre entidades.
Amazon DevPay	PaaS, SaaS	Sistema de facturación y gestión de la contabilidad para el cobro por uso de las aplicaciones desarrolladas con AWS.
Amazon SimpleDB	PaaS	Servicio de base de datos NoSQL [Strauch *et al.*, 2011] de alta velocidad y escalabilidad.
Amazon Relational Database Service (RDS)	PaaS	Servicio de base de datos relacional.
Amazon ElastiCache	PaaS	Servicio de almacenamiento basado en pares

		clave-valor para la construcción de sistemas de cache
Amazon Simple Storage Service	PaaS	Servicio de almacenamiento de ficheros.
Amazon Elastic Block Storage	IaaS	Servicio elástico de provisión de unidades de disco virtuales para las instancias.
AWS Import / Export	PaaS	Servicio de importación y exportación de grandes cantidades de datos.
AWS Identity and Access Management	PaaS	Control de acceso a recursos por parte de los usuarios finales de las aplicaciones desarrolladas.
Amazon CloudWatch	SaaS	Monitorización de los servicios.
AWS Elastic Beanstalk	IaaS, PaaS	Despliegue automatizado de aplicaciones.
AWS CloudFormation	IaaS	Simplificación de la contratación de recursos computacionales.
Amazon Simple Queue Service	PaaS	Servicio de cola de mensajes entre servidores.
Amazon Simple Notification Service	PaaS	Servicio de envío de notificaciones multicanal.
Amazon Simple Email Service	PaaS	Envío de correo electrónico masivo.

Tabla 23.- Catálogo de servicios en Amazon Web Services

C.1.2 GOOGLE APP ENGINE

Google comenzó a especializarse en la computación distribuida y de alto rendimiento a través de su producto estrella, el buscador. Posteriormente, con la introducción productos ampliamente conocidos como Gmail[102] y Google Docs[103] se convirtió en uno de los primeros proveedores de SaaS basado en CC.

[102] https://gmail.google.com
[103] https://docs.google.com

En 2008 se incorporaron al mercado PaaS con su producto *Google App Engine*[104]. Esta plataforma permite el desarrollo de aplicaciones escalables que son desplegadas en el sistema CC de Google. Las aplicaciones se ejecutan en entornos controlados a través de diferentes servidores de la infraestructura, pero el cliente no tiene ningún control directo sobre los nodos que ejecutan la aplicación. Así pues, el cliente entrega el paquete de la aplicación a Google que automáticamente se despliega, por lo que el servicio es puramente PaaS.

Las aplicaciones deben estar desarrolladas en *Python*[105], *Java*[106] o *Go*[107]; y el soporte para bibliotecas es limitado debido a las restricciones de la infraestructura de despliegue. No obstante, Google ofrece una gran cantidad de servicios, que cubren la mayoría de las necesidades de las aplicaciones web. Debido a que no hay posibilidad de configurar el entorno de despliegue, es necesario suplir esa funcionalidad con subsistemas que normalmente se despliegan por separado, o por tareas ejecutadas sin interacción de los usuarios en el servidor.

A continuación en la Tabla 24 se presenta una descripción de los principales servicios ofertados

Servicio	Nivel	Descripción
Cloud Endpoints	PaaS	Interacción con aplicaciones Android.
Google Cloud Storage API	PaaS	Lectura y escritura de ficheros alojados en *Google Cloud Storage*
Search API	PaaS	Búsqueda de texto sobre los datos almacenados en Datastore API.
MapReduce API	PaaS	Ejecución de tareas de procesamiento de datos siguiendo el modelo *MapReduce* [Dean *et* Ghemawat, 2008].
Prospective Search API	PaaS	Servicio de búsqueda sobre flujos de datos en tiempo real.
ProtoRPC API	PaaS	Servicio de llamadas a procedimiento remoto entre aplicaciones.
Task Queue API	PaaS	Servicio para la ejecución de tareas asíncronas en el lado servidor.

[104] https://appengine.google.com
[105] https://www.python.org/
[106] https://www.java.com
[107] https://code.google.com/p/go/

OAuth API	PaaS	Integración con OAuth[108].
OpenID	PaaS	Integración con OpenId[109].
App Identity API	PaaS	Identificación de aplicaciones para el acceso a servicios provistos por otras aplicaciones.
Blobstore API	PaaS	Almacenamiento y servicio de ficheros.
Capabilities API	PaaS	Detección de fallos y paradas programadas en los servicios de *App Engine*.
Channel API	PaaS	Gestión de conexiones persistentes entre clientes Javascript y el servidor, de manera similar a los *Websockets*.
Datastore API	PaaS	Almacén de datos estructurados (servicio de base de datos NoSQL [Strauch *et al.*, 2011]).
Datastore Async API	PaaS	Versión no bloqueante del servicio datastore.
Images API	PaaS	Servicio de almacenamiento y manipulación de imágenes.
LogService API	PaaS	Servicio de logging de aplicación.
Mail API	PaaS	Servicio de envío de correo electrónico.
Memcache API	PaaS	Servicio de caché en memoria para las aplicaciones.
Namespaces API	PaaS	Servicio de fragmentación de datos para aplicaciones que deseen separar por clientes la información.
Remote API	PaaS	Uso de los servicios de App Engine desde aplicaciones no alojadas en la nube.
URL Fetch API	PaaS	Realización de peticiones HTTP y HTTPS.
Users API	PaaS	Gestión de cuentas de usuario mediante las cuentas de Google.
XMPP API	PaaS	Servicio de envío y recepción de mensajes mediante protocolo XMPP (*Extensible Messaging and Presence Protocol*) hacia y desde aplicaciones de mensajería instantánea.

Tabla 24.- Catálogo de servicios en Google App Engine

[108] http://oauth.net/
[109] http://openid.net/

C.1.3 WINDOWS AZURE

Finalmente, como último proveedor principal, la apuesta de Microsoft por el paradigma CC es Windows Azure[110], que es una plataforma que provee servicios de nivel infraestructura y de nivel plataforma. Permite la contratación de servidores VPS basados en Windows y en Linux, y dispone de una capa PaaS similar a la de Amazon.

Los servicios ofrecidos por Windows Azure se separan en tres grandes grupos [Redkar *et* Guidici, 2011]:

- **Cómputo.** Servicio de hosting virtual escalable (IaaS), con sistemas operativos huésped Windows Server o Linux.
- **Almacenamiento.** Servicios PaaS de almacenamiento de datos y ficheros y servicio de cola de mensajes, accesibles a través de servicios web de tipo REST[111].
- **Gestión.** Gestión automatizada de la infraestructura virtual contratada, desde las instancias de máquinas virtuales a los componentes de red.

En la Tabla 25 se presenta una relación de los servicios ofertados por este proveedor.

Servicio	Nivel	Descripción
Virtual Machines	IaaS	Servicio de hosting virtual escalable. Permite configurar los parámetros de elasticidad, las instancias, la configuración de red virtual y el balanceo de carga.
Hosted Cloud Services	PaaS	Despliegue automatizado de aplicaciones desarrolladas. Se admiten dos roles de aplicación: *website* y *worker*, optimizados para el servicio de datos o para el trabajo en segundo plano.
SQL Database	PaaS	Servicio de base de datos relacional escalable.
Tables	PaaS	Servicio de base de datos no relacional.
BLOB Storage	PaaS	Servicio de almacenamiento de ficheros.
SQL Reporting	PaaS	Generación semiautomática de informes de negocio a partir de los datos de la aplicación.

[110] http://azure.microsoft.com/
[111] http://www.w3.org/TR/ws-arch/

Marketplace	PaaS	Permite comercializar aplicaciones desarrolladas.
Hadoop	PaaS	Servicio de procesamiento de grandes volúmenes de datos empleando *Apache Hadoop*[112] siguiendo un esquema MapReduce [Dean *et* Ghemawat, 2008].
Active Directory	PaaS	Gestión de cuentas de usuario.
Access Control Service	PaaS	Autenticación de usuarios contra el servicio *Active Directory*[113].
Azure Service Bus	PaaS	Infraestructura de mensajería entre aplicaciones.
Queue Storage	PaaS	Servicio de almacenamiento de mensajes.
Media Services	PaaS	Creación, administración y distribución de contenido multimedia.

Tabla 25.- Catálogo de servicios en Microsoft Windows Azure

C.1.4 COMPARACIÓN Y OTROS PROVEEDORES

A continuación, en la Tabla 27 se presenta una tabla comparativa de los tres grandes proveedores de servicios CC que se acaban de presentar en los apartados anteriores (C.1.1, C.1.2 y C.1.3).

Proveedor / Característica	Amazon	Azure	App Engine
VPS Elástico	Sí	Sí	No
Configuración de red y balanceo de carga	Sí	Sí	N/A
Cálculo Intensivo	Map-Reduce	No específico	No
BBDD Relacional	RDS (beta)	SQL Database	Table Storage
BBDD NoSQL	SimpleDB	Table Storage	Datastore
Almacén de Ficheros	S3	BLOB Storage	Cloud Storage Blobstore
Cola de Mensajes	SQS	Queue API	Task Queue API

Tabla 26.- Comparativa de proveedores Cloud Computing

[112] http://hadoop.apache.org/
[113] http://support.microsoft.com/kb/196464/es

Además de estos proveedores, existen otras muchas empresas que han decido apostar por este negocio. En este sentido, la modernización de la tecnología empleada en los proveedores de *hosting* virtual ha provocado un cambio natural hacia la infraestructura como servicio. Por otra parte, la necesidad de abstraer los mecanismos de escalabilidad de las grandes aplicaciones, provoca la aparición de proveedores de PaaS. Ambas características han dado lugar al nacimiento de diferentes compañías que centran su negocio en la oferta de servicios computaciales a través de internet. Así a continuación, en la Tabla 27, se presenta una breve reseña de los más importantes.

Proveedor	Nivel	Modelo de Despliegue	Dirección de Internet
BitRefinery	IaaS	Público	http://bitrefinery.com/
GoGrid	IaaS	Público, Híbrido	http://www.gogrid.com/
Hosting.com	IaaS	Público, Híbrido, Privado	http://www.hosting.com/
NephoScale	IaaS	Público	http://www.nephoscale.com/
OpSource	IaaS	Público, Privado	http://www.opsource.net/
RackSpace	IaaS	Público, Híbrido, Privado	http://www.rackspace.com/
ReliaCloud	IaaS	Público	http://www.reliacloud.com/
SoftLayer	IaaS, PaaS	Público, Privado	http://www.softlayer.com/
Terremark	IaaS, PaaS	Público	http://www.terremark.es/
AppFrog	PaaS	Indepen. de modelo	http://www.appfog.com/
Apprenda	PaaS	Indepen. de modelo	http://apprenda.com/
Cloud Foundry	PaaS	Indepen. de modelo	http://www.cloudfoundry.com/
CumuLogic PaaS	PaaS	Público	http://www.cumulogic.com/
Engine Yard	PaaS	Público	http://www.engineyard.com/
GigaSpaces Cloudify	PaaS	Indepen. de modelo	http://www.gigaspaces.com/
LongJump PaaS	PaaS	Público, Privado	http://www.longjump.com/
NetSuite SuiteCloud Platform	PaaS	Público	http://www.netsuite.com/
OpenShift	PaaS	Público	https://openshift.redhat.com/
Stackato	PaaS	Privado	http://www.activestate.com/stackato

Tabla 27.- Provedores de servicios Cloud Computing

C.2 PLATAFORMAS CLOUD COMPUTING

Hoy en día, existe una gran variedad de plataformas CC abiertas y distribuidas bajo la licencia *Open Source* [114] . A nuestro juicio las más importantes y, quizás, más utilizadas son OpenStack, OpenNebula y, en menor medida, Eucalyptus. No cabe duda que existen otras plataformas, sin embargo después de analizarlas no están diseñadas para cubrir un espectro generalista, o bien su único interés es el de facilitar la gestión de la capa *hardware* física y virtual, tarea que la mayoría de los sistemas de virtualización modernos ya incorpora como tarea por defecto.

En el estado del arte, como no podía ser de otro modo existen diferentes trabajos que comparan estas plataformas [Wen *et al.*, 2012] [Von Laszewski *et al.*, 2012] [Mahjoub *et al.*, 2011] [Voras *et al.*, 2011] [Sempolinski *et* Thain, 2010]. A continuación en la Tabla 28 se muestra una relación de las principales plataformas CC en la actualidad.

Prov.	Niv.	Empr.	Lic.	Plataf.	Descripción
vSphere	IaaS	VMWare	Comer	N/A	Sistema operativo orientado a la formación de infraestructuras CC a través de virtualización.
Cloud Foundry	PaaS	VMWare	Libre	vSphere EC2	Capa PaaS para *vSphere*, que puede ser desplegada en infraestructuras de otros proveedores.
CloudStack	IaaS	Citrix	Libre	VMWare, Oracle VM, KVM, Xen	Capa de gestión de infraestructura que abstrae la plataforma tecnológica de virtualización subyacente.
Eucalyptus	IaaS	Eucalyptus Systems	Libre	Xen, KVM, VMWare	Capa de gestión de infraestructura que provee una interfaz compatible con *Amazon EC2*.
OpenStack	IaaS, PaaS	Consorcio	Libre	KVM, Xen	Gestión de infraestructura y servicios PaaS.
AppScale	PaaS	Comun.	Libre	KVM, XEN, EC2, Eucalyptus	Ejecución en cloud privado de aplicaciones de *Google App Engine*.
Nimbus	IaaS	Comun.	Libre	KVM, Xen	Gestión de infraestructura.
OpenQRM	IaaS	Comun.	Libre	KVM, Xen, VMWare	Gestión de infraestructura de centros de proceso de datos.

Tabla 28.- Plataformas Cloud Computing

[114] http://opensource.org/

C.3 ENTORNO TECNOLÓGICO

La computación de tipo CC está caracterizada fundamentalmente por su escalabilidad, es decir, la capacidad del sistema de atender a un número mayor de clientes incrementando los recursos *hardware* y *software* sin afectar a su implementación y por su elasticidad, que hace referencia a la capacidad de escalar dinámicamente con la carga del servicio. Por ello, los paradigmas tradicionales de comunicación entre aplicaciones, de sistemas operativos, de almacenamiento de ficheros o de sistemas gestores de bases de datos no son directamente aplicables a este nuevo entorno. Las aplicaciones deben ser capaces de escalar a todos sus niveles, sin que su arquitectura impida aprovechar la flexibilidad de otras capas. Los pilares tecnológicos de la computación en nube son:

- El uso de la virtualización para facilitar la administración de grandes redes de computadoras.
- El uso de bases de datos distribuidas.
- El uso de sistemas de ficheros distribuidos.

A lo largo de este apartado ser revisarán las tecnologías que fundamentalmente se han revisado en el marco de este trabajo de investigación, incluyendo la virtualización (apartado C.3.1), bases de datos distribuidas (apartado C.3.2) y, finalmente, los sistemas de ficheros distribuidos (Apartado C.3.3).

C.3.1 VIRTUALIZACIÓN

En el Anexo B, anterior, ya se presentó una descripción detallada de la tecnología de virtualización (Apartado B.2.4). A continuación a lo largo de este apartado se hará una breve revisión de las principales tecnologías de virtualización que se han revisado, utilizado y considerado en el transcurso de este trabajo de investigación.

C.3.1.1 XEN

Xen[115] es un sistema de virtualización desarrollado inicialmente por la Universidad de Cambridge [Barham *et al.*, 2003] y liberado bajo licencia de *software* libre GPLv2. Actualmente es mantenido por la compañía Citrix[116], que mantiene

[115] http://www.xenproject.org/
[116] http://www.citrix.com/

además XenServer[117], una versión comercial del entorno de virtualización.

Se trata de un hipervisor del tipo *baremetal* [Ekanayake *et* Fox, 2010], este modelo de hipervisores se ejecutan directamente sobre el *hardware*, así es posible aislar los sistemas operativos huéspedes y anfitrión entre sí. Permite paravirtualización y virtualización completa:

- En caso de que se desee emplear la paravirtualización el sistema operativo huésped debe ser modificado para soportar las operaciones de bajo nivel de Xen. Para ello se emplea también una versión modificada del *kernel* de la máquina huésped.
- La virtualización completa permite ejecutar versiones no modificadas de sistemas operativos, incluyendo entornos Windows.

Permite la migración de máquinas virtuales entre diferentes servidores anfitriones sin que sea necesario detenerla. Pero, para ello, requiere que el disco virtual de la máquina huésped esté alojado en un sistema de ficheros distribuido al que ambos anfitriones tengan acceso. Así mismo, también permite la asignación dinámica de memoria RAM y procesadores virtuales entre las máquinas virtuales. Sin embargo, para ello se requiere que el núcleo de la máquina huésped cuente con opciones especiales que aparecieron en la versión 3.1 del kernel de Linux.

En la actualidad Xen es utilizado por una gran cantidad de proveedores de hosting y de infraestructura como servicio, incluyendo AWS[118] y Rackspace[119].

C.3.1.2 KVM

KVM o *Kernel-based Virtual Machine* [120] es un sistema de virtualización basado en Qemu [Bellard, 2005] que permite la virtualización completa y la paravirtualización de sistemas operativos en entornos Linux. Actualmente, su desarrollo está apoyado por Red Hat Inc[121].

En lugar de proporcionar una capa de abstracción hacia el *hardware*, proporciona al *kernel* Linux una serie de infraestructuras orientadas a la virtualización. De esta manera, el núcleo de la máquina anfitrión tiene acceso a los procesos que se ejecutan en los nodos virtuales.

[117] http://www.citrix.es/products/xenserver/overview.html
[118] http://aws.amazon.com/
[119] http://www.rackspace.com/
[120] http://www.linux-kvm.org/
[121] http://www.redhat.com/

Al igual que Xen, permite la migración en vivo de máquinas virtuales entre sistemas anfitrión, también precisando un sistema de almacenamiento compartido. La asignación de memoria RAM es menos rígida que en otros entornos de virtualización y funciona correctamente con versiones estables (2.6) del *kernel* de Linux.

C.3.1.3 OPENVZ

OpenVZ [122] es un sistema de virtualización basado en contenedores. Es un proyecto más joven que el resto de entornos de virtualización.

Para su funcionamiento, OpenVZ requiere que la máquina huésped y las máquinas anfitrión ejecuten la misma versión del núcleo de Linux. Esto restringe su funcionamiento a sistemas Linux con la misma versión del kernel. Actualmente, el soporte se encuentra restringido a las versiones 2.6 del kernel.

Como contrapunto a esa rigidez, el rendimiento de OpenVZ es ligeramente mayor que otros entornos de virtualización y, además, es más sencillo de administrar. La reasignación de RAM y CPU a las máquinas virtuales no requiere modificaciones en el núcleo de los sistemas huésped.

Sin embargo, en cuanto al aislamiento de las máquinas, los contenedores únicamente crean un sistema virtual de ficheros Linux (creando ficheros especiales que representen los dispositivos virtuales). A través del sistema virtual de ficheros y una capa de abstracción, las máquinas virtuales sólo tienen acceso a sus procesos y tienen sus propias cuentas de usuario. Sin embargo, un aislamiento tan débil provoca que los fallos en las máquinas virtuales se propaguen con facilidad a la máquina anfitrión.

C.3.1.4 COMPARATIVA Y OTROS SISTEMAS DE VIRTUALIZACÓN

En la literatura pueden encontrarse numerosas comparativas entre estos sistemas [Xu *et al.*, 2008] [Camargo *et al.*, 2008] [Regola *et* Ducom, 2010]. Los tres entornos de virtualización que se acaban de presentar exhiben un conjunto de funcionalidades bastante similares, ya que están están basados en Linux y muestran rendimientos similares.

A continuación en la Tabla 29 se muestra una comparación de las principales características de estos tres sistemas de virtualización de cara a ser utilizadas para proporcionar servicios elásticos.

[122] http://openvz.org/

	Xen	KVM	OpenVZ
Paravirtualización	Sí	Sí	Sí
Virtualización completa	Sí	Sí	No
Modificación de CPU	Con kernel modificado	Con kernel modificado	Sí
Modificación de RAM	Limitada	Sí	Sí
Modificación de disco	Sí	Sí	Sí
Migración en vivo	Sí, pero requiere almacenamiento compartido	Sí	Sí
Ventajas	Estabilidad, proyecto	Estabilidad, proyecto, pero más joven que Xen	Rendimiento y flexibilidad.
Desventajas	Problemas con la elasticidad	Bajo soporte a la paravirtualización	Bajo aislamiento

Tabla 29.- Comparación de los entornos de virtualización analizados

Además de los sistemas presentados, también existen otras alternativas para la virtualización de sistemas operativos; distribuidos tanto con licencias libres, como comerciales. A continuación se enumeran algunas de ellas:

- La suite **VMWare** [123] es el entorno de virtualización comercial de referencia. Permite migración en vivo e implementa características de elasticidad, pero requiere el pago de licencias proporcionales al volumen del despliegue.
- El entorno de virtualización **Oracle VM** [124] está optimizado para el despliegue de aplicaciones basadas en tecnologías de la marca, es decir, Oracle: base de datos y aplicaciones desarrolladas en Java. No es libre ni gratuito.
- **Hyper-V**[125] es el sistema de virtualización desarrollado por Microsoft y optimizado para la virtualización de entornos Windows, que también permite ejecutar instancias de Linux. No es libre ni gratuito.
- **VirtualBox** es el entorno de virtualización libre desarrollado por Sun Microsystems/Oracle [126], para la virtualización completa de distintos

[123] http://www.vmware.com/
[124] http://www.oracle.com/us/technologies/virtualization/oraclevm/overview/index.html
[125] http://www.microsoft.com/oem/es/products/servers/Pages/hyper_v_server.aspx
[126] https://www.virtualbox.org/

sistemas operativos. Su uso suele estar restringido a entornos de escritorio.

C.3.2 BASES DE DATOS DISTRIBUIDAS

En el apartado B.2.1.1 del Anexo B se presenta los modelos de persistencia de información utilizados en los sistemas CC donde priman las bases de datos noSQL, frente al modelo relacional. A lo largo de este apartado se hace una breve revisión de las bases de datos que se han estudiado, utilizado o considerado en el transcurso de este trabajo de investigación.

C.3.2.1 MONGODB

MongoDB [127] es una base de datos orientada a documentos que en la actualidad se está convirtiendo en un estándar de facto en el campo de las bases de datos NoSQL. Es un proyecto que se encuentra en constante desarrollo, con una gran comunidad de usuarios, y usado en una gran cantidad de proyectos en empresas.

Para conseguir la escalabilidad y la alta disponibilidad de las bases de datos MongoDB, se emplean dos mecanismos distintos y complementarios:

- **La replicación de la información.** En el que se definen varios servidores que forman parte de un conjunto de replicación o *ReplicaSet*. Dentro del conjunto de replicación, los servidores escogen un servidor primario mediante votación. El servidor primario debe ser el objetivo de las operaciones de escritura. Por su parte, las operaciones de lectura se pueden hacer a cualquier servidor del conjunto. Cuando un servidor primario deja de funcionar, se produce una reelección de primario de manera transparente al usuario.
- **El *sharding* o partición de la información.** La información se divide, utilizando funciones de dispersión, entre varios ordenadores. Cuando este mecanismo se utiliza en combinación con el de replicación, la información se divide entre conjuntos de réplicas. De esta manera, la carga computacional y de red se divide de manera uniforme entre estos conjuntos.

Un servicio especial se encarga de encaminar las peticiones al servidor de bases de datos correcto, por lo que las aplicaciones no necesitan conocer en

[127] https://www.mongodb.org/

ningún momento la opción de escalabilidad elegida. Sencillamente necesitan conocer un subconjunto de las máquinas que prestan el servicio de encaminado. En MongoDB, estas máquinas reciben el nombre de *routers*.

C.3.2.2 APACHE CASSANDRA

Apache Cassandra [128] es una base de datos no relacional distribuida, desarrollada inicialmente por Facebook y mantenida como proyecto de la *Apache Software* Foundation [129]. Está diseñada específicamente para la gestión de grandes volúmenes de datos repartidos entre gran cantidad de servidores con *hardware* normal. Provee almacenamiento estructurado clave-valor, un modelo similar a los documentos pero con mayor rigidez respecto cambios en el esquema.

Es una base de datos descentralizada, que soporta replicación; y cuyo caudal de lectura/escritura incrementa linealmente con el número de máquinas añadidas. Es un proyecto similar a BigTable de Google [Chang *et al.*, 2008].

C.3.2.3 MYSQL CLUSTER

MySQL Cluster CGE [130] es la versión distribuida de la base de datos de licencia libre MySQL[131], actualmente desarrollada y mantenida por Oracle. Para su funcionamiento emplea un motor de base de datos *ad-hoc* denominado NDB (*Network Database*).

Los mecanismos para la escalabilidad de MySQL Cluster se basan, al igual que en el caso de MongoDB, en la replicación y la partición de información:

- **Replicación**, que permite la redundancia síncrona, que garantiza que los datos escritos se propagan a todos los nodos que forman parte de la réplica al producirse el commit.
- **Partición**, que permite que los datos se particionan de manera automática empleando un algoritmo que distribuye las tuplas a los

[128] http://cassandra.apache.org/
[129] http://www.apache.org/
[130] https://www.mysql.com/downloads/clustercge/
[131] http://www.mysql.com/

nodos basándose en la clave primaria de la tabla. También se puede especificar un esquema de partición manualmente.

Estos mecanismos se implementan mediante tres tipos de nodos. En primer lugar, el (i) gestor que se encarga de la puesta en marcha del sistema, la conexión de nuevos servidores y al ejecución de comandos de administración. En segundo lugar, el (ii) nodo de datos, que se encarga del almacenamiento final de los datos. Son las máquinas que más potencia requieren, por mantener los índices total o parcialmente en memoria y los datos en disco. Los nodos de datos deben ser uniformes para evitar la creación de cuellos de botella. Finalmente, los (iii) nodos API son los puntos de acceso a la base de datos, que dirigen y coordinan las peticiones generadas por las sentencias que reciben.

C.3.2.4 MARÍA DB GALERA CLUSTER

MariaDB[132] Es un sistema de gestión de bases de datos derivado de MySQL[131], la cuál está distribuida con el mismo tipo de licencia. La principal ventaja que tiene MariaDB es su alta compatibilidad con MySQL ya que utiliza las mismas interfaces y bibliotecas.

Su principal ventaja, frente a MySQL es la capacidad de distribuir los datos persistentes entre diferentes nodos, de forma que es posible tener una base de datos relacional de alta disponibilidad. Para ello, sigue un modelo de replicación donde cada nodo sirve su propia base de datos y los cambios se replican en el resto.

C.3.2.5 COMPARATIVA

A continuación se muestra en la Tabla 30 una breve comparativa de las bases de datos distribuidas que se acaban de presentar, atendiendo a facilidad de configuración, paradigma empleado. En el estado del arte se pueden encontrar diferentes trabajos que comparan las características de todas ellas [Bunch *et al.*, 2010] [Zhang *et al.*, 2012].

[132] https://mariadb.org/

	MongoDB	Apache Cassandra	MySQL Cluster	María DB
Dificultad de configuración	Baja	Alta	Media/Alta	Media
Paradigma	Orientada a documentos	Almacenamiento clave-valor	Relacional distribuido	Relacional distribuido
Transacciones	Gestionadas por la aplicación.	Gestionadas por la aplicación	Sí	SI

Tabla 30.- Comparación de las bases de datos distribuidas analizadas

C.3.3 SISTEMAS DE FICHEROS DISTRIBUIDOS

Al igual que en apartados anteriores, las peculiaridades de los sistemas de ficheros distribuidos se presentaron en el Anexo B, concretamente en el apartado B.2.1.2, por lo que en esta sección se incluirán los sistemas de ficheros distribuidos revisados a lo largo del proceso de investigación.

C.3.3.1 NFS

NFS (*Network File System*) [Sandberg *et al.*, 1985] es un protocolo de nivel de aplicación, que es utilizado como sistema de ficheros distribuido en entornos de área local. Está ampliamente extendido ya que se incluye en la mayoría de sistemas operativos Linux.

A partir de la cuarta versión del protoclo (NFSv4[133]) se incluye soporte para la replicación de datos, pero no hay soporte para el fraccionamiento automatizado, por lo que la aplicación cliente es la responsable de encontrar los ficheros que precisa.

C.3.3.2 GlusterFS

GlusterFS[134] es un sistema de archivos escalable para NAS (*Network Attached Storage*). Es utilizado por una gran variedad de ambientes y aplicaciones como computación en nube, ciencias biomédicas y almacenamiento de archivos.

[133] http://tools.ietf.org/html/rfc3530
[134] http://www.gluster.org/

Su funcionamiento se basa en la creación de *volúmenes de almacenamiento virtuales* que se pueden montar desde otro dispositivo. Estos volúmenes pueden ser distribuidos y replicados. GlusterFS permite configurar el número de servidores, réplicas e incluso fraccionar un fichero y distribuirlo en varios servidores para ganar en velocidad. Además se pueden combinar los tipos de volúmenes como se desee.

C.3.3.3 COMPARATIVA

A continuación, en la Tabla 31 se incluye una breve comparativa de ambos sistemas de ficheros distribuidos. Puede encontrarse más información en la literatura especializada [Brunet *et al.*, 2012] [Dewan *et* Hansdah, 2011].

	NFS	**GlusterFS**
Facilidad de configuración	Baja	Baja
Replicación	A partir de NFSv4.	Sí
Partición automática de datos	No.	Sí

Tabla 31.- Comparación de los sistemas de ficheros distribuidos analizados

ANEXO D PROYECTOS, PUBLICACIONES Y TRABAJOS RELACIONADOS

D.1 PROYECTOS

Los dos proyectos de investigación que están directamente relacionados con el desarrollo de esta tesis son los siguientes.

- **iHAS: Intelligent Social Computing for Human-Agent Societies** (TIN2012-36586-C03-0)
  - Web: http://ihas.usal.es/
  - Entidad financiadora: Ministerio de Ciencia e Innovación y los Fondos FEDER
  - Descripción:
  - El proyecto se centra en el desarrollo de mecanismos, algoritmos, herramientas y modelos que permitan la creación de sistemas abiertos en los que conviven e interactúan agentes virtuales y humanos de forma transparente en un entorno totalmente integrado. Denominamos este tipo de sistemas sociedades humano-agente (human-agent societies HAS).
    El trabajo planteado pretende dar respuestas a preguntas como:¿Qué es necesario conocer y diseñar para que los humanos interactúen con los agentes *software*? y ¿Cómo se deben formalizar y estructurar estas interacciones para obtener productos *software* que sean efectivos en este tipo de entornos?
- **OVAMAH: Organizaciones Virtuales Adaptativas: Mecanismos, Arquitecturas y Herramientas** (TIN 2009-13839-C03-03)
  - Web: http://www.ia.urjc.es/OVAMAH/
  - Entidad financiadora: Ministerio de Ciencia e Innovación. Proyectos de Investigación Fundamental No Orientada.
  - Descripción:
    El principal objetivo de este proyecto es el estudio y propuesta de nuevos métodos, herramientas y/o mecanismos que permitan la evolución y organización de organizaciones virtuales basadas en tecnología de sistemas multi-agente. De tal manera que permita la detección de situaciones no cooperativas o no deseadas mediante la evaluación dinámica del sistema así como su posterior tratamiento para que la organización evolucione de forma autónoma y sea capaz de adaptarse a la nueva situación.

Por otro lado también se destacan los tres siguientes proyectos de investigación y desarrollo:

- **Cloud-IO: Plataforma Cloud Computing para la Integración y Despliegue Rápido de Servicios sobre Redes Inalámbricas de Sensores** (IDI-20111471)
  - Web: http://cloud-io-project.com/
  - Entidad financiadora: Centro para el Desarrollo Tecnológico e Industrial (CDTI). Ministerio de Economía y Competitividad. Fondos FEDER.
  - Descripción:
  - El proyecto que tiene como objetivo trabajar en el desarrollo de la interoperabilidad, accesibilidad y usabilidad de los sistemas basados en Redes Inalámbricas de Sensores, a través de una nueva plataforma de servicios Cloud Computing.
- **Smart Medical: Plataforma Cloud Computing para gestión de historiales clínicos en aseguradoras privadas** (IDI-20120794)
  - Web: http://smartmedical.es/
  - Entidad financiadora: Centro para el Desarrollo Tecnológico e Industrial (CDTI). Ministerio de Economía y Competitividad. Fondos FEDER.
  - Descripción:
  - El proyecto tiene como principal objetivo el análisis, diseño y desarrollo de una innovadora plataforma basada en el revolucionario modelo Cloud Computing y construida sobre la base de una arquitectura multiagente específicamente diseñada para gestionar historiales clínicos de pacientes por parte de Aseguradoras Privadas.
- **DoyFE.ES: Verificación y prevención de fraude en contenidos digitales** (IDI-20120798)
  - Web: http://www.doyfe.es/
  - Entidad financiadora: Centro para el Desarrollo Tecnológico e Industrial (CDTI). Ministerio de Economía y Competitividad. Fondos FEDER.
  - Descripción:
    - El objetivo del proyecto es diseñar y crear un sistema online para la verificación y prevención de fraude en documentos digitales. En este sentido, DOYFE.ES hará énfasis en la capacidad para interactuar con el sistema de manera sencilla y práctica, de modo que dejar constancia digital verificable de una serie de comunicaciones por correo electrónico, o de las condiciones de servicio de un determinado proveedor o vendedor en Internet sea sencillo, eficaz y barato.

D.2 PUBLICACIONES

Respecto a la publicaciones, el siguiente listado muestra algunas de las más relevantes relacionadas con la tesis. El listado completo de publicaciones está accesible en: http://bisite.usal.es.

D.2.1 ARTÍCULOS EN REVISTAS INTERNACIONALES

De la Prieta, F., Heras S., Palanca J., Rodríguez, S., Bajo, J., and Julián, V. (2013, In press) Real-Time Negotiation and Fulfillment of SLAs in Cloud Computing Environments AI Communications. ISSN: 0921-7126. IOS Press. In press. Impact factor: 0.5

De la Prieta, F., Rodríguez, S., Corchado, J.M. & Bajo, J. (2014, In press) Infrastructure to simulate intelligent agents in cloud environments. In: Journal of Intelligent & Fuzzy System. ISSN: 1064-1246. Impact factor: 0.788.

Heras, S., **De la Prieta, F.**, Julian, V., Rodríguez, S., Botti, V., Bajo, J., & Corchado, J. M. (2012). Agreement technologies and their use in cloud computing environments. Progress in Artificial Intelligence, 1(4), 277-290. Springer Verlag

De la Prieta, F., Rodríguez, F., Bajo, J. & Corchado, J.M. (2014, In press) +Cloud: A virtual organization of multiagent system for resource allocation into a Cloud Computing environment. In: LNCS Transactions on Computational Collective Intelligence. ISSN: 2190-9288. Springer Verlag

Jiménez, A., Casado, A., Bajo, J., **De la Prieta, F.**, & De Paz, J.F. (2014, In press). Labor Integration of Deaf People Through a Cloud-based Platform. In: International Journal of Artificial Intelligence. ISSN 0974-0635. Ceser Publications.

D.2.2 ARTÍCULOS EN ACTAS DE CONGRESOS DE RECONOCIDO PRESTIGIO

De la Prieta, F., Rodríguez, S., Bajo, J., & Corchado, J. M. (2013). A multiagent system for resource distribution into a Cloud Computing environment. In Advances on Practical Applications of Agents and Multi-Agent Systems (pp. 37-48). Springer Berlin Heidelberg.

De la Prieta, F., Rodríguez, S., Bajo, J., & Lopez Batista, V. F. (2013, October). Data integration in Cloud Computing environment. In Fourth International Workshop on Knowledge Discovery, Knowledge Management and Decision Support. Atlantis Press.

De la Prieta, F., Bajo, J. & Rodriguez, S. (2013) Sistema multiagente para la asignación de recursos en entornos distribuidos. Multiconferencia CAEPIA

2013. Amparo Alonso-Betanzos, Concha Bielza, Antonio Salmerón, Abraham Duarte, J. Ignacio Hidalgo, Luis Martínez, Edurne Barrenechea, Alicia Troncoso, Emilio Corchado, Juan M. Corchado, Francisco Herrera, José C. Riquelme (Eds.). ISBN: 978-84-695-8348-7

De la Prieta, F., Sánchez, A. J., Zato, C., Rodríguez, S., & Bajo, J. (2013). .Cloud: Unified Platform for Compilation and Execution Processes in a Cloud. InAdvances in Artificial Intelligence (pp. 219-227). Springer Berlin Heidelberg.

Heras, S., **De la Prieta, F.**, Rodríguez, S., Bajo, J., Botti, V. J., & Julián, V. (2012). The Role of Argumentation on the Future Internet: Reaching agreements on Clouds. In AT (pp. 393-407).

Heras, S., **De la Prieta, F.**, Rodríguez, S., Bajo, J., Botti, V. J., & Julián, V. (2012). A novel model of argumentation for Cloud Computing paradigm. In ISCIF'13 workshop. IBERAMIA 2012

González, R., Hernández, D., **De la Prieta, F.**, & Gil, A. B. (2013). +Cloud: An Agent-Based Cloud Computing Platform. In Distributed Computing and Artificial Intelligence (pp. 377-384). Springer International Publishing.

Tapia, D. I., Alonso, R. S., García, Ó., **De la Prieta, F.**, & Pérez-Lancho, B. (2013, January). Cloud-IO: Cloud Computing Platform for the Fast Deployment of Services over Wireless Sensor Networks. In 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing (pp. 493-504). Springer Berlin Heidelberg.

De la Prieta, F., Corredera, L.E., Sánchez-Martin, A.J. & Demazeau, Y. (2014, In press). An agent-based cloud platform for security services.In: Highlights in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. Springer Verlang

Jiménez, A., Casado, A., Bajo, J., **De la Prieta, F.**, & De Paz, J. F. (2013). Cloud-Based Platform to Labor Integration of Deaf People. In Distributed Computing and Artificial Intelligence (pp. 633-640). Springer International Publishing.

De la Prieta, F., Bajo, J., Marín, P. A. R., & Méndez, N. D. D. (2013). A New Generation of Learning Object Repositories Based on Cloud Computing. InManagement Intelligent Systems (pp. 99-106). Springer International Publishing.

De la Prieta, F., Gil, A. B., Rodríguez, S., Pérez, J. B., Coria, J. A. G., & Corchado, J. M. (2014, January). An Enhanced Approach to Retrieve Learning Resources Over the Cloud. In The 2nd International Workshop on

Learning Technology for Education in Cloud (pp. 193-203). Springer Netherlands.

Rodríguez, S., Tapia, D. I., Sanz, E., Zato, C., **De la Prieta, F.**, & Gil, O. (2010). Cloud computing integrated into service-oriented multi-agent architecture. In Balanced Automation Systems for Future Manufacturing Networks (pp. 251-259). Springer Berlin Heidelberg.

Bajo, J., Zato, C., **De la Prieta, F.**, de Luis, A., & Tapia, D. (2010). Cloud computing in bioinformatics. In Distributed Computing and Artificial Intelligence (pp. 147-155). Springer Berlin Heidelberg.

D.3 PROPIEDADES INTELECTUALES

Finalmente, también se ha realizado el registro de propiedad intelectual asociada a este trabajo de investigación. Los datos de esta propiedad intelectual se presentan a continuación:

- **WAREHOUSE 3.0: HERRAMIENTA DE ALMACENAMIENTO Y BÚSQUEDA DE INFORMÁCIÓN EN ENTORNOS CLOUD COMPUTING** (SA-200-13)
  - Autores: Daniel Hernández Alfageme, Ana Belén Gil González, Fernando De la Prieta Pintado, Juan M. Corchado Rodríguez, Juan F. De Paz Santana, Sara Rodríguez González y Antonio J. Sáncehz Martín.
  - Descripción:
    El objetivo principal de la aplicación es permita el almacenamiento y la búsqueda de ficheros en entornos CC, el entorno de explotación de la aplicación ha sido la platforma +Cloud desarrollada por el grupo de investigación BISITE.