CARMiCLOC: Context Awareness Middleware in Cloud Computing

Jose Aguilar, Marxjhony Jerez, Ernesto Expósito, Thierry Villemur

▶ **To cite this version:**

Jose Aguilar, Marxjhony Jerez, Ernesto Expósito, Thierry Villemur. CARMiCLOC: Context Awareness Middleware in Cloud Computing. 2015 Latin American Computing Conference (CLEI), Oct 2015, Arequipa, Peru. pp.1 - 10, 10.1109/CLEI.2015.7360013 . hal-01464457

HAL Id: hal-01464457
https://laas.hal.science/hal-01464457

Submitted on 10 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CARMiCLOC: Context Awareness Middleware in CLOud Computing

Jose Aguilar* & Marxjhony Jerez
CEMISID, Dpto. de Computación, Facultad de Ingeniería, Universidad de Los Andes
Mérida - Venezuela
aguilar@ula.ve, marxjhony@ula.ve
*Prometeo researcher, Universidad Técnica Particular de Loja, Ecuador

Ernesto Exposito & Thierry Villemur
CNRS, LAAS, 7, avenue du Colonel Roche, F-31400
Univ de Toulouse, INSA, LAAS, F-31400
Toulouse, France
ernesto.exposito@laas.fr, thierry.viellemur@lass.fr

***Abstract*— Technological advances such as miniaturization of sensors, mobile devices and increased computational capabilities have given way to Context Aware Applications (ACC), which constantly monitor and change depending on the context. These applications require a variety of features and services, particularly on the management of context. When applications/services lack part or some of these services, they must be consumed from other suppliers who can provide them via a middleware. In addition, due to the changes undergone by the context mean that such services consumed by the ACC must be dynamic; and due to the amount of data involved to define the context, it may require more computation capacity than that provided by a mobile device. For these reason it is necessary the use of the Cloud. In this way, to support these requirements is proposed CARMiCLOC (Context Awareness in Reflective Middleware Cloud Computing), a web-service based middleware, which can behave as a SaaS (Software as a Service) or as a PaaS (Platform as a Service).**

Keywords—Context-Awareness; Reflective Middleware Web Services; Ontologies.

I. INTRODUCCIÓN

Los adelantos tecnológicos, la miniaturización de sensores y dispositivos móviles y el aumento de capacidades de computo han dado espacio a la creación de un nuevo tipo de aplicaciones que constantemente están monitoreando y reaccionando a la información contextual. Estas aplicaciones emergentes, las aplicaciones conscientes del contexto (o *Context-Awareness Applications (CAAs)* en inglés), poseen características claves como, que son intensamente conscientes del contexto y continuamente se adaptan a los cambios en el contexto. La Consciencia de Contexto es una de las funcionalidades claves de los sistemas de computación ubicua, para darle cabida a la Internet de las Cosas.

El desarrollo y la ejecución de las CAAs típicamente son soportados por middleware denominados *Context-Awarenes Middleware (CaM)*, [01] [02] [03] [04] [05]. A nivel de la literatura, en las CaMs se presenta una gran diversidad en las estructuras, arquitecturas y técnicas que se usan en el ciclo de vida de gestión del contexto: adquisición del contexto, modelado del contexto, razonamiento del contexto y distribución del contexto [06] [07], como en los soportes en tiempo de ejecución respecto al procesamiento, comunicación y almacenamiento.

En cada una de las etapas del ciclo de vida del contexto podemos encontrar diversos problemas y retos. Uno de los grandes retos en el área es diseñar una abstracción de programación para aplicaciones conscientes de contexto, bajo un enfoque de abajo hacia arriba y descentralizado. Eso permitirá el desarrollo de infraestructuras que soporten el rápido desarrollo de aplicaciones conscientes del contexto, con medios de gestión de servicios que hagan la gestión del contexto para dichas aplicaciones.

En el área de adquisición de contexto los servicios están vinculados al monitoreo y captura de información sobre el contexto/entorno, de tal manera de capturar todo el acontecer en el contexto, y permitir a las aplicaciones ser más adaptativas. La recolección del contexto del usuario se está realizando en base al comportamiento de los objetos del entorno y al tipo de comunicación que este use en el mismo, para lo cual diversas disciplinas de investigación están siendo usadas en esta tarea (como inteligencia artificial e ingeniería de software) [7] [8]. Uno de los problemas por resolver es garantizar la interoperabilidad e integración de la gran diversidad de sensores que se presentan en los contextos.

Dentro del área de modelos de contexto, se debe garantizar que dichos modelos contengan información sobre el tipo de contexto, el tiempo de ocurrencia o valides de la información, las fuentes de información, etc. Dicha información en el modelo debe ser confiable y significativa, resolviéndose todas inconsistencias y ambigüedades en su contenido [6] [7]. Igualmente, el modelado puede requerir almacenar un gran volumen de datos, por lo que eventualmente se podrán explotar técnicas de minería de datos para extraer conocimiento.

En cuanto al razonamiento del contexto, las aplicaciones conscientes de contexto requieren la inferencia del contexto de manera distribuida. Además, las aplicaciones van a requerir de procesos de razonamiento inexacto. Por la inconsistencia y ambigüedad que pueda existir en el contexto. Algunos aspectos que se vienen considerando aquí tienen que ver con el uso de patrones o esquemas para la detección, resolución de inconsistencia y diagnóstico de situaciones de interés en el contexto, en ambientes heterogéneos, dinámicos y distribuidos

978-1-4673-9143-6/15/$31.00 ©2015 IEEE

[7] [10]. La incorporación del razonamiento semántico en la gestión del contexto es una nueva área de estudio [8].

Otros problemas por resolver tienen que ver con las necesidades de cómputo y de comunicación, de almacenamiento de contexto en tiempo de ejecución, por lo que se requiere optimizar esos procesos a través de paradigmas como la concurrencia de procesos, la computación de la nube, la programación orientada a servicios, entre otros. También, la seguridad y privacidad del contexto son problemas por resolver.

En este artículo se propone una arquitectura de un middleware reflexivo consciente de contexto basado en la computación autonómica, para proporcionar adaptabilidad, y escalabilidad haciendo uso del paradigma computación de la nube. Es un middleware basado en servicios web, para prestar servicios de gestión del contexto a aplicaciones conscientes de contexto. El artículo presenta una sección de trabajos relacionados, después se presenta la arquitectura base de CARMiCLOC (Context Awareness Reflective Middleware in Cloud Computing), para luego presentar cada uno de los servicios que componen cada una de las fases del ciclo de gestión del contexto, para finalizar con un caso de uso y las conclusiones.

II. TRABAJOS RELATIVOS

Diversos estudios relacionados a las aplicaciones conscientes de contexto han sido realizadas en [6] [7] [8] y presentan taxonomías, y marcos de trabajos de arquitecturas ideales de middleware conscientes de contexto. Por ejemplo, [6] muestra una arquitectura conceptual ideal de una aplicación consciente de contexto, y una taxonomía de la funciones y componentes que se presentaron en 50 trabajos estudiados entre el periodo comprendido entre 2001 y 2011 acerca de aplicaciones conscientes de contexto en Internet de las Cosas. En [7] se presenta un marco de trabajo sistemático referencial para las aplicaciones conscientes de contexto, también se realiza un estudio del estado del arte en cada uno de los ciclos de vida de gestión del contexto, y propone los retos para futuros trabajos en el área. En [8] se muestra un estudio de middlewares que gestionan contextos, presentan los diferentes marcos de trabajos e implementaciones en base a la representación, gestión y razonamiento del contexto, además realizan una evaluación de los mismos, lo que permite tener idea de las diferentes maneras de implementar un middleware de gestión de contexto.

En COCAAAL [5] se presentan un diseño de un modelo basado en la nube para ambientes conscientes de contexto. Se presenta el desarrollo de una arquitectura orientada a servicios, que soporta servicios conscientes de contexto, y sugiere una nueva propuesta de modelado de contexto así como un algoritmo para clasificar el contexto.

En [9] se muestra una arquitectura de un middleware para el procesamiento de contexto y servicios conscientes de contexto sobre un PaaS (Plataform as a Service), se señalan retos a considerar en la implementación de una aplicación consciente de contexto dentro de la nube (como el decidir qué datos y procesos colocar en la nube y cuáles deben ser locales), y propone un modelo funcional de implementación de su propuesta.

En [3] se presenta una arquitectura consciente de contexto para Internet de la cosas (*Context Awareness for Internet of Things* (CA4IoT)), que resuelve el problema de selección de sensores de acuerdo a los problemas/tareas a manejar. Se enfoca en la configuración automatizada de filtrado, fusión, y mecanismos de razonamiento que pueden ser aplicados para recoger el flujo de datos de los sensores seleccionados.

En [1] se presenta un middleware llamado CAMPUS (*Context-Aware Middleware for Pervasive and Ubiquitous Service)*, para la adaptación automática del contexto para la toma de decisión en tiempo real. Para ello, este middleware se apoya en Ontologías y en el análisis del dominio de aplicación y del dominio de contexto, permitiendo la adaptación dinámica de acuerdo al contexto y a las aplicaciones que estén recibiendo servicio de él.

En [4] se introduce un Middleware multiplataforma centrado en el humano, es llamado Webinos (Secure WebOS Application Environment), y fue uno de los proyectos de la comunidad europea para estandarizar las interconexiones entre el hardware y el software consciente del contexto. Ese middleware introduce los conceptos de *Zona Personal* como una red sobrepuesta, también introduce el concepto de *Objeto de Contexto*, que contiene toda la información del contexto para transferirlo de manera sencilla sobre la Zona Personal. Es basado en RPC y posee un API para ser usado en dominios de Entretenimiento, Vehículos, y Hogar, entre otros.

En [2] se presenta el problema de Middleware Reflexivo Conscientes de contexto en los sistemas embebidos. Se propone un middleware llamado MARCHES (*Middleware for Adaptive Robust Collaborations across Heterogeneous Environments and Systems*), que soporta la reflexión a nivel de componentes y a nivel del sistema. A nivel de componentes, la reflexión gestiona el comportamiento y el contenido de los componentes. A nivel del sistema, la reflexión gestiona la estructura de las conexiones entre los componentes, para descubrir y operar la cadena de componentes activa. Esto permite a MARCHES, a través de la reflexión, examinar el estado interno del sistema en tiempo de ejecución, y reconfigurar dinámicamente la arquitectura de la aplicación para mejorar su adaptabilidad.

Por último en [11] se propone una arquitectura para la computación de la nube en aplicaciones móviles, llamada *MapCloud* la cual provee funcionalidades de ubiquidad a los dispositivos móviles, además, presentan una manera de resolver el problema de qué información debe ser local y cual debe estar en la nube. Definen un ambiente autónomo en el cual los diferentes dispositivos móviles obtienen recursos de computación, almacenamiento, y otros servicios y recursos, eficientemente en cualquier momento.

En general, en los middleware conscientes de contextos se encuentran un buen número de trabajos en los que se enfatiza la adaptación en tiempo real, sin embargo, en ellos falta la auto-adaptación, como por ejemplo bloquear a las aplicaciones hasta que no se actualice el contexto. Otro ejemplo de auto-

adaptación es la capacidad de hacer composiciones de servicios de gestión de contexto en tiempo de ejecución, en función de las necesidades del momento de las aplicaciones.

En CARMiCLOC se propone una arquitectura de middleware reflexivo basado en la computación autonómica, soportado por la computación de la nube, que es soportado sobre la computación de la nube, para proporcionar escalabilidad, auto-adaptatibilidad, integración e interoperabilidad, a diferencia de [1], [2], [3], [4] y [5], tal que CARMiCLOC será descentralizado y basado en servicios web. CARMiCLOC integra un conjunto de servicios que le permitirá ser autónomo y escalable, de igual manera podrá prestar los servicios de manera individual para los consumidores que lo requieran, o comportarse como un middleware de gestión completa de entornos basados en contexto. El problema de decisión de que tareas dejar locales o en la nube presentado en [6], se resolverá haciendo uso de los principios de Mapcloud [11]. Así, CARMICLOC permitirá gestionar el contexto de la zona en donde se encuentra la aplicación que lo instancio, y cada servicio que interactúe será supervisado por una instancia de CARMiCLOC para asegurar consistencia del contexto.

III. ARQUITECTURA BASE DE CARMICLOC

CARMiCLOC es una arquitectura de Middleware Reflexivo [12] que incorpora la computación autonómica [13] para la auto-adaptación, y usa el paradigma de la computación de la nube [14] para permitir escalabilidad. Es una arquitectura basada en servicios web para que puedan ser consumidos sus servicios por las aplicaciones conscientes de contexto o no.

A. Bases filosoficas de la arquitectura.

Las bases filosóficas de la arquitectura de CARMiCLOC son la computación reflexiva, la computación autonómica, y la computación de la nube.

La reflexión es la habilidad de nuestro middleware para monitorear y modificar su propia conducta, así como también aspectos de su implementación (sintaxis, semántica, etc.), permitiéndole la habilidad de ser sensible a su contexto. En este sentido, CARMiCLOC posee un comportamiento dinámico y una arquitectura adaptativa, es decir, puede manejar servicios que son capaces de cambiar dinámicamente o evolucionar. En general, este tiene dos procesos [12]:

- **La introspección:** La habilidad de observar y razonar acerca de su propio estado de ejecución.
- **La intersección:** Es la habilidad de modificar su propio estado de ejecución, o alterar su propio significado o interpretación.

Así, CARMiCLOC tiene una conducta dinámica y adaptativa, completamente distribuida, con el fin de tener un mejor conocimiento de los cambios que puedan ocurrir dentro del contexto gestionado. CARMiCLOC se divide en los dos niveles clásicos de todo middleware reflexivo.

El **Nivel Base**: Se refiere a los usuarios/aplicaciones que se encuentren en la zona de contexto compartido [4]. El nivel base del middleware debe conocer y monitorear las interacciones en el contexto, así como los cambios en él. El nivel Base puede pertenecer a la porción local de la aplicación/servicio o la porción de la nube de la misma.

El **Nivel Meta**: Esta es la parte del middleware que ofrece la capacidad de reflexión. La introspección (observación) del nivel base se hace analizando los cambios en el contexto. En específico, en el nivel meta se tienen los servicios que presta CARMiCLOC, el gestor de servicios del contexto, y un registro de todos los servicios en uso en la zona de interacción.

La implementación de la introspección y adaptación de CARMiCLOC ha sido pensada basada en el paradigma de computación autonómica. La Computación Autonómica es un modelo de computación de auto-gestión inspirado en el sistema nervioso autónomo del ser humano [13]. Son sistemas capaces de auto-administrarse, para lo cual se define una arquitectura compuesta por 6 niveles [13]:

- **Recursos Manejados:** Puede ser cualquier tipo de recurso (hardware o software), y puede tener incluido atributos de autogestión. Corresponde al registro de servicios, sensores y contextos.
- P**untos de Enlace:** Implementa los sensores y/o actuadores requeridos para la gestión de los recursos.
- **Gerente Autonómico:** Implementa los lazos de control inteligentes que automatizan las tareas de autorregulación de las aplicaciones. Está compuesto por cuatro módulos que caracterizan el lazo de control autónomo: *Monitoreo, Análisis, Planificación y Ejecución (MAPE)*. El módulo de monitoreo recolecta los eventos/datos de los sensores. El módulo de análisis analiza las situaciones de interés, el módulo de planificación decide y organiza las tareas a realizar, y el módulo de ejecución envía las órdenes a los componentes del sistema autonómico.
- **Orquestador de los Gerentes Autonómicos:** Proporciona la coordinación entre los gestores autonómicos locales. Coordina cada uno las instanciaciones de CARMiCLOC.
- **Gerente Manual:** Crea la interfaz hombre máquina para tener acceso a los gestores autonómicos. Se implementa para realizar configuraciones manuales de algunos sensores o reconfiguraciones de usuarios.
- **Fuentes de Conocimiento:** Proporciona acceso a los conocimientos requeridos para la gestión autonómica del sistema. En ella se registran el histórico y los meta-contextos de dominio.

La fuente de conocimiento podrá hacer uso de la nube para guardar y procesar data cuando los dispositivos no tengan capacidades de cómputo para estas tareas computacionales.

El diseño de CARMiCLOC se muestra en la figura 1, en donde se observan los niveles Base y Meta de la computación reflexiva. Los puntos de enlace se encuentran entre estos niveles. Dentro del Nivel meta se observa el Gerente

Autonómico y la Base de Conocimiento, cada de uno de ellos está compuesto por uno o más servicio **Se_i**. Los puntos de enlaces se encuentran entre los niveles meta y base.

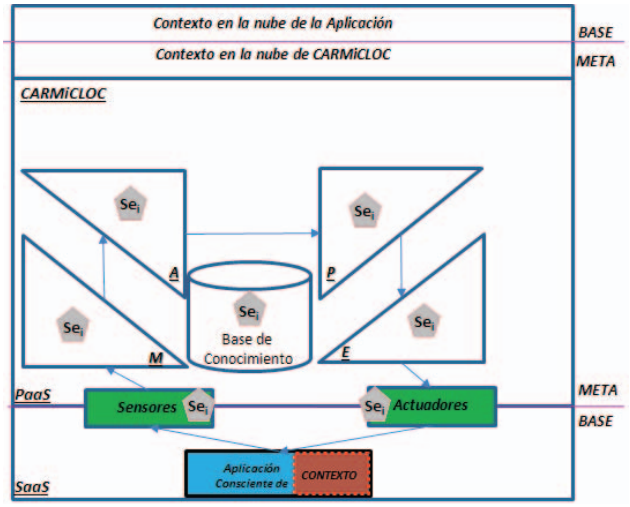

Figura 1. Arquitectura base de CARMICLOC

B. *Servicios de CARMICLOC*

CARMiCLOC incorpora siete servicios para realizar la gestión del contexto, los servicios pueden prestarse de manera interna (I) (requerido por los mismos servicios de CARMiCLOC) o llamados independientemente por servicios externos (E). Los Servicios de CARMiCLOC pueden ser consumidos por:

A. Aplicaciones Consciente de Contexto (CAA)

B. Aplicaciones No Conscientes de Contexto.

CARMiCLOC ofrecer servicios precisos que son requeridos en para la gestión del ciclo de vida del contexto. En la tabla 1 se muestra un resumen de los servicios que presta CARMICLOC, estos servicios son:

Se_1: Adquisición de Contexto/Pre configuración. Dentro de este servicio se encuentra las actividades de descubrimiento del contexto y anotación semántica del contexto [8]. Además, da soporte a las fuentes de datos (capa de sensores (físicos, virtuales o lógicos o dispositivos móviles)) o puntos de enlace de la computación autonómica. También, una tarea fundamental es el pre-procesamiento de datos, en este caso debe realizar tareas de predicción de datos perdidos, detección de valores atípicos y coincidencia de datos, entre otros. También, realiza la configuración automática de los sensores, así como su registro. Utiliza diversas técnicas de adquisición de datos y contexto.

Se_2: Modelado de Contexto.

Es uno de los servicios más importantes de CARMICLOC, en él se realiza la gestión del conocimiento, a través del modelado del conocimiento, la extracción del conocimiento, el razonamiento e inferencia (Se_3: Servicio 3), y la gestión de conocimiento de un dominio especifico. Existen diversas técnicas de modelado, pero en CARMICLOC se hará uso de las Ontologías [6] [8]. Otra función dentro de este servicio involucra el almacenamiento de data, y la gestión de herramientas para su explotación, tales como la minería de datos, reconocimiento de patrones, entre otros. Otra tarea dentro del modelado es la gestión datos con problemas cuando se invoca a Se_1 (en este caso, debe manejar data imperfecta, data ambigua, etc.).

Se_3: Razonamiento de Contexto.

Es responsable del procesamiento de la data, de aplicar las técnicas de razonamiento, de la gestión de eventos (análisis de patrones de comportamiento, análisis de tendencias), de la fusión de data, del análisis de la incertidumbre del contexto y de la Calidad del Contexto. Además, se encarga de razonar, hacer cambios al contexto, y sugerir las posibles acciones a realizar en el contexto. Es parte importante del planificador de MAPE. En este servicio se encuentran los operadores de fusión de datos que son usados para procesar data.

Se_4: Distribución de Contexto. Es responsable de hacer entrega del contexto a los usuarios, hace llamada al servicio 2 (Se_2). Se encarga de aplicar las técnicas de distribución, y entre sus funcionalidades tenemos: el formateo de los datos (incluye la transformación de los datos, la conversión de unidades, la personalización de datos, etc.), también la accesibilidad (permite facilitar su uso, ofrece documentación, alternativas y opciones múltiples de acceso), manejo de requerimiento de usuarios (procesamiento de consultas, construcción del conocimiento y entendimiento semántico) y la gestión de consumidores de servicios (registro, mantenimiento de perfiles: historial, preferencias, etc.).

TABLA I. TABLA RESUMEN DE LOS SERVICIOS DE CARMICLOC.

Id	Servicio a prestar	Interno(I)/ Externo (E)	Descripción
Se_1	Adquisición/Pre configuración de Contexto	I, E(B)	Servicio encargado de la agregación y filtrado de data, el descubrimiento y anotación del Contexto
Se_2	Modelado de Contexto	I, E	Servicio encargado de realizar la gestión del conocimiento (Modelo), almacenamiento de la data, compartir el Contexto y configuración en tiempo de ejecución
Se_3	Razonamiento del Contexto	I, E	Servicio encargado del gestión e inferencia del conocimiento, gestión de eventos, fusión de data
Se_4	Distribución del Contexto	E(A)	Servicio de la diseminación del contexto y servicios
Se_5	Verificación de inconsistencia del Contexto (QoC)	E(B), I	Servicio encargado de la Validación o resolución de conflictos del contexto
Se_6	Seguridad del Contexto	I	Servicio encargado de la seguridad y privacidad del usuario, aplicación y data
Se_7	Gestor de servicios de contexto		Servicio encargado la gestión de servicios

Se_5: Verificación y resolución de inconsistencia del Contexto.

Es responsable de detectar inconsistencias en el contexto y tomar acciones para la resolución de los mismos, a través de técnicas de conflictos. Hace uso de Se_3 para sus tareas.

Se_6: Seguridad y privacidad del contexto.

Es responsable de estas tareas dentro de todo el ciclo de vida del contexto, en particular, de la seguridad y privacidad de los datos

Se_7: *Gestor de Servicios de Contexto.*

Es responsable de la gestión de los servicios dentro del contexto, estos incluyen los servicios que puedan estar en la nube. Este es un orquestador de los servicios internos de CARMiCLOC. En este servicio se realiza la invocación de los servicios después del análisis del razonador Se_3, o la adaptación de un servicio en base a la semántica y contexto de los requerimientos.

C. *Diseño de los Servicios de Carmicloc*

Los servicios a implementar están diseñados basados en los trabajos de [3] [6] [7]. En esta sección presentamos el diseño de dos de ellos, algunos de los más relevantes.

Servicio 1: Adquisición de Contexto/Pre configuración

Este servicio es importante para los sistemas conscientes de contexto [9] [11], y representa uno de los puntos de enlace entre el exterior y CARMiCLOC, Las funciones de este servicio se muestran en la figura 2.

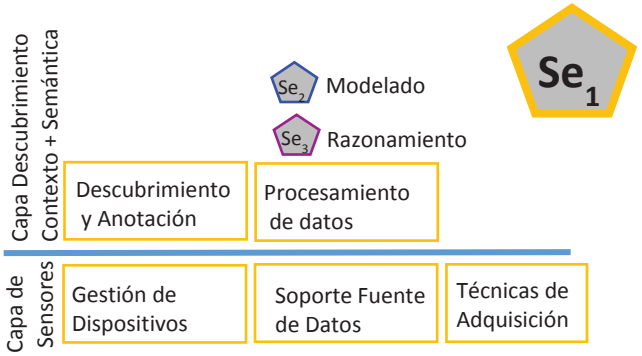

Figura 2. Diseño del Servicio 1 (Se_1): Adquisición/Pre configuración de datos.

El servicio de Adquisición/Pre configuración (Se_1) tiene dos capas, y llama a los servicios Se_2 (Modelado) y Se_3 (Razonamiento). La primera capa es de sensores, y realiza las siguientes funciones:

a. Soporte a la Fuente de Datos: esta función da soporte a los diferentes fuentes de datos (permite la interconexión a los mismos): Sensores físicos, Sensores de Software (virtuales o lógicos), cualquier dispositivo Móviles, etc.

b. Gestión de Dispositivos: Esta función se encarga de la configuración automática de los sensores. Se hace uso de herramientas que usan Archivos de Definición de sensores (ADS) y herramientas de redes sensores como Global Sensor Network (GSN)[1].

c. Técnicas de adquisición: se encarga de la captura de los datos, puede ser instantánea o a intervalos, desde las diferentes fuentes del Contexto (sensores directos, a través de Middleware, a través de servidores de Contexto).

La otra capa realiza el Descubrimiento de Contexto y el análisis semántico, para lo cual realiza las siguientes funciones:

a. Procesamiento de datos, que tiene dos subfunciones

   a. Filtrado de Data: Se encarga de depurar y limpiar los datos, eliminando los datos obsoletos, datos no coherentes, datos duplicados, etc.

   b. Agregación de Data: Agrupa los datos que son del mismo contexto

b. Anotación de contexto: permite agregarle al contexto y a la data cruda de sensores información, en particular, semántica. Algunas informaciones que se agregan en relación al contexto son: Tipo de contexto, valor del contexto, estampilla de tiempo, fuente y confidencia.

c. Descubrimiento de contexto: se basa en procesar la información de las fuentes de datos, a través de la fusión de datos, procesar el contexto adquirido con contexto en el histórico (Se_2), razonar sobre su vinculación a algunos de los contexto (Se_3), para descubrir el contexto actualmente usado.

En la figura 3 se observa en diagrama de actividades de la capa de sensores, se encarga de identificar los sensores que realizan llamadas al servicio, solicitando la información de los sensores para poder identificarlos según su naturaleza (sensores directos (físicos o lógicos), middleware de contexto o servidor de contexto, y buscar los Envoltorios de Sensores (ES) dentro de la base de conocimiento de CARMiCLOC. De no encontrarse envoltorio, se procede a buscar el Archivo de Definición del Sensor (ADS) en el repositorio local o en la Nube, y de no encontrarse se agrega manualmente. El ADS permitirá generar el Envoltorio para cada sensor para que pueda utilizar la técnica de adquisición del ADS.

Por ejemplo, un Corredor de Fondo puede tener un reloj que posee un sensor de monitor de pulso (RMP) y un GPS (RGPS). Por otro lado, en su teléfono móvil lleva la aplicación de CARMiCLOC. CARMiCLOC deberá reconocer los sensores presentes, buscar los envoltorios para cada uno, y seleccionar un Monitor de Pulso y un GPS en base a filtrado de data, seleccionando el RMP y el PGPS por ser más confiable y más usado según el histórico del sistema.

La figura 4 muestra el diagrama de actividades de la capa de análisis semántico y de descubrimiento de contexto, cuyos datos entrantes vienen de los sensores, y son filtrados para

[1] Global Sensor Network. http://open-platforms.eu/library/global-sensor-networks-gsn/

eliminar la data inexacta y obsoleta, luego se agrupa por similitud, de allí se realizan las anotaciones de tiempo, dominios, etc., para luego ser transferidos a la función de descubrimiento de contexto, esta función pasa la información al Servicio de Razonamiento, en donde se invoca la función de gestión del conocimiento para comparar con históricos y data actual, y generar los Descubridores Semánticos y de Contexto, esta información será almacenada en los repositorios correspondientes, para su posterior uso.

Para nuestro ejemplo anterior, teniendo la entrada de datos de un corredor de fondo, se agrupan los datos de localización y Pulso, y se analiza el contexto, para lo cual se determina la hora y lugar en el que el corredor realiza el entrenamiento desde el servicio de Descubridor de Contexto (DSC), y a partir de allí se comienza a monitorear la información.

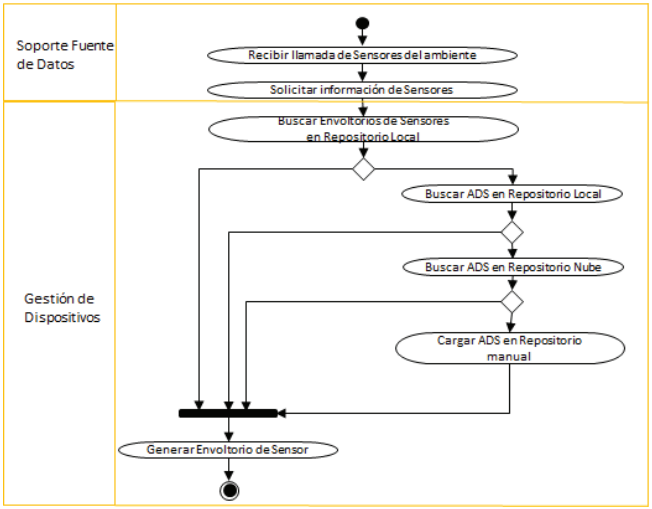

Figura 3. Diagrama de Actividades de la capa de Sensores

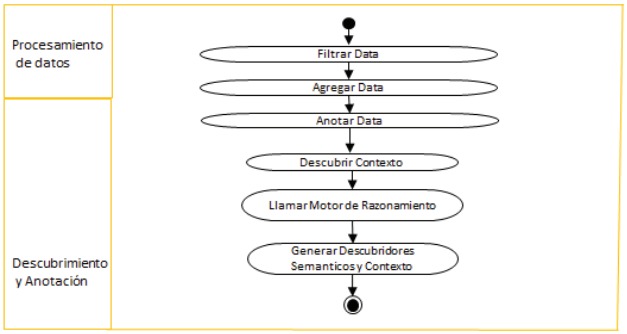

Figura 4. Diagrama de Actividades de la Capa de Análisis Semántico y de Descubrimiento de Contexto

Se2: Servicios de Modelado

El modelado consiste en el diseño de patrones para representar un objeto o dinámica del contexto. Involucra uno de los componentes más importantes del middleware, la base de conocimiento dentro de los componentes MAPE-K. Dentro de este servicio se desarrollan una serie de funciones importantes para comparar cambios de contexto en el middleware. Estas funciones se pueden apreciar en la figura 5, con los servicios complementarios y cada una de las funciones que se deben llamar.

Como esta capa está basada en las técnicas de Modelado, él es responsable de gestionar esas diferentes técnicas de modelado del contexto, entre ellas: esquemas de modelado basado en objetos, basado en lógica o basado en ontología. Esta última es la que será usada en nuestra implementación, sin embargo, será necesario poder transformar desde cualquiera de las otras técnicas a una ontología y viceversa, para permitir interoperabilidad con aplicaciones que tenga otras técnicas de modelado. En general, las tareas a realizar en este servicio son:

1. Pre procesamiento, es una función muy importante pues prepara la data para ser usada para construir el modelo. Las subfunciones son: Manejo de data imperfecta (no precisa, obsoleta, o incompleta), manejo de data ambigua (conflictos, inconsistencia, etc.), Predicción de datos perdidos, detección de anomalías, y la búsqueda de equivalencias de datos.

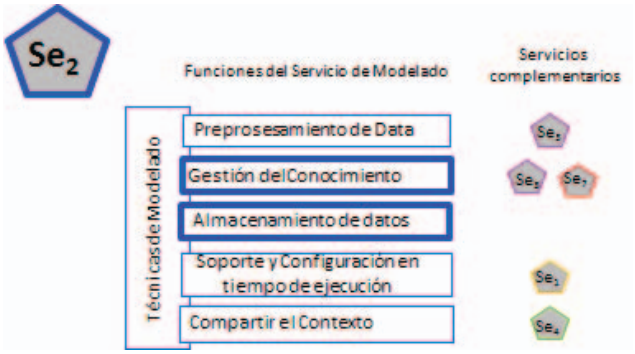

Figura 5. Funciones del Servicio 2 (Se2): Modelado del Contexto

2. Gestión del conocimiento: entre las funciones que poseen la gestión del conocimiento tenemos: Modelado del conocimiento, que consiste en construir el patrón o imagen del Objeto de Conocimiento deseado a modelar; extracción del conocimiento para poder entregar el conocimiento requerido en un momento especifico, y razonamiento o inferencia. Estas dos funciones requieren de la llamada al Servicio de Razonamiento (Se_3). Esta información del contexto puede estar almacenada localmente o en la nube.

3. Almacenamiento de datos (histórico de la data). A esta función le corresponde almacenar todos los datos que ya han pasado por el pre-procesamiento y modelado, para poder ser luego utilizados para obtener estadísticas o por el proceso de razonamiento. Las subfunciones son las siguientes: minería de datos (permite descubrir patrones desde grande volúmenes de datos), reconocimiento de patrones, procesamiento distribuido de bases de conocimiento y datawarehouse. La gestión de los repositorios de Envoltorios, de Contextos, etc., se encuentran en esta función. Puede hacer uso de la nube para guardar datos.

4. Configuración en tiempo de ejecución. Esta función en tiempo de ejecución en muy importante para resolver problemas emergentes. Así, se debe tener una arquitectura Plug-in y Hot pluggable, que permita la incorporación de cambios en tiempos de ejecución. De igual manera, se debe permitir interoperabilidad con otros servicios y contextos.

5. Compartir contexto. Esta función permite compartir de manera automatizada el contexto entre múltiples dispositivos. Debe tenerse en cuenta las técnicas de modelado que otros estén usando, y trabaja en conjunto con el servicio de distribución de contexto (Se_4).

Para un mejor entendimiento del uso de este servicio presentaremos un diagrama de actividades en la figura 6. Se presentan las funciones más importantes que siempre son llamadas en cada proceso: el pre procesamiento de la data, para que la data sea tratada con la mayor precisión posible; la Gestión del Conocimiento, para que la data sea modelada, extraída, etc.; y el almacenamiento de la data, lo que permite llevar un registro de todos los cambios que sufre la data y qué se requiere para poderla accesar. La data colectada por el Se_1 es tomada por este servicio y de nuevo procesada de manera más exhaustiva, una vez procesada la data se llama al servicio de Razonamiento, en donde se llama el histórico de data, se buscan patrones en diferentes dominios, etc.

Siguiendo nuestro ejemplo de un corredor de fondo, se ha detectado que realiza un entrenamiento, se ha modelado y se ha analizado la información del contexto, alertando al corredor en los momentos en los que la información monitoreada este fuera de los parámetros configurados para el entrenamiento (pulso, distancia y sitio de entrenamiento). Luego del entrenamiento se almacena la información y se le indica al corredor cual ha sido su rendimiento. Además, se compara con las sesiones de entrenamiento anteriores, y le sugiere un nuevo plan de entrenamiento para una nueva sesión.

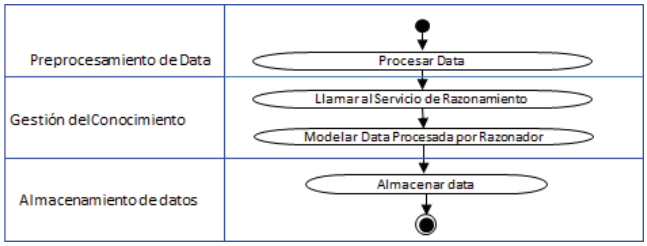

Figura 6. Diagrama de actividades de funciones mas importantes de Servicio de Modelado de Contexto

D. Especificacion de MAPE de CARMiCLOC

Nuestro Gestor Autonómico es compuesto por todos los elementos MAPE-K de la estructura de la Arquitectura de Computación Autonómica, los cuales están en el nivel Meta (ver Figura 7). Cada gestor autonómico puede trabajar de manera íntegra y distribuida con el resto (trabajando a nivel del middleware con el resto para proveer propiedades de contexto a aplicaciones que lo requieran), o simplemente proveer servicios específicos a consumidores que lo invoquen (internos o externos al middleware).

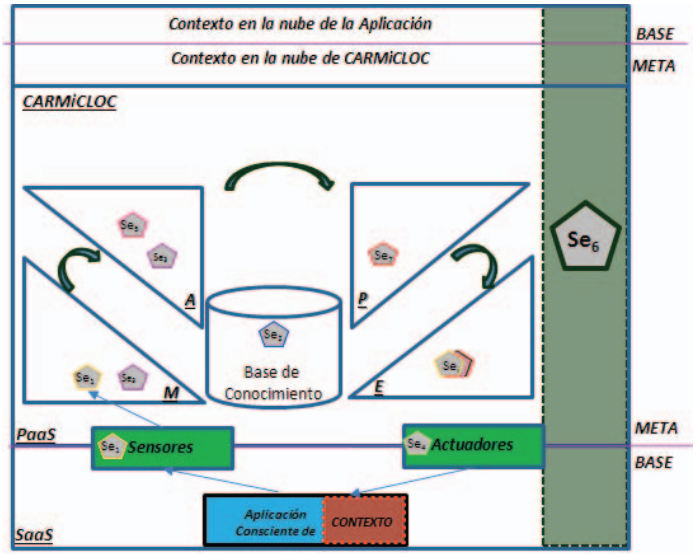

Figura 7. Arquitectura MAPE de CARMiCLOC

a. Componente Monitor

El Monitor es responsable de la observación de nuevos cambios en la data del contexto, o de nuevos requerimientos del usuario. Dentro de este componente se encuentra el servicio de Se_1 (Adquisición de Contexto), y siempre en sus comunicaciones actúa Se_6 (Seguridad y Privacidad).

b. Componente Analizador.

El Analizador es responsable de determinar si los eventos ocurridos han modificado el contexto, y de ser así pasan los resultados al Planificador para que este pueda decidir qué acciones tomar. En este componente se ejecuta los servicios Se_3 y Se_5. El servicio Se_3 se encarga de inferir los posibles patrones de comportamiento para suministrar información al planificador y determinar el plan a seguir, llamando al gestor de Servicios (Se_7). Sin embargo, los cambios en el contexto pueden deberse a inconsistencias en el contexto, por lo que el Servicio Se_5 detecta inconsistencias, y de haberlas, deberá resolverlas para no dar falsos positivos.

c. Componente Planificador

El Planificado del middleware recibe la llamada del analizador para que tome acciones. Para ello usa al servicio razonador (Se_3), que le permite inferir que hacer. El resultado será la orquestación de un conjunto de servicios a invocar con el Objeto de Contexto necesario. Para ello se invoca al servicio Se_7.

d. Componente Ejecutor

El ejecutor solo se encarga de ejecutar los servicios que han sido propuestos por el planificador para ser

invocados o adaptado según el nuevo contexto emergido

e. Componente Base de Conocimiento

Se refiere a los repositorios de contexto, Modelos, etc. que contiene toda la base de conocimiento del Middleware. Es un componente que interviene en todo los procesos de ejecución, y corresponde al servicio de Modelado (Se_2), y eventualmente al de verificación de contexto (Se_5).

f. Puntos de enlaces

Es donde se encuentran los Sensores y actuadores del sistema. En el punto de enlace de sensores se encuentran todas las fuentes de datos, y allí actúa el Servicio de Adquisición de Contexto (Se_1). Allí subyace la Capa de Sensores y actuadores. Por otro lado, el Servicio Se_4 (Distribución del Contexto) se encarga de entregar los datos según los requerimientos del usuario/Servicio/Aplicación, a los actuadores.

IV. CASO DE ESTUDIO

A. *Caracterizacion del caso de estudio*

Para probar nuestra propuesta se toma un caso de estudio de un adulto de 65 años con Hipertensión Arterial (HTA) y Diabetes Mellitus tipo II, y además sufre de episodios de sincope (desmayos repentinos). Para llevar un monitoreo y asistencia de vida de esta persona, lleva consigo una serie de sensores (MCG[2], MCPA[3] , MCPC[4]) que permiten conocer su condición actual de salud y cuadro clínico. Además, lleva un marcador de Radio Frecuencia (RF), el cual interactúa con un conjunto de lectores de RF instalado en cada uno de los ambientes de la casa, permitiendo poder conocer su ubicación dentro de la misma. El paciente ha tenido este sistema de monitoreo desde hace más de 2 años, lo que ha permitido realizar minería de datos y establecer patrones los cuales permiten determinar cuando el paciente se ve afectado por alguna condición, como lo es el *sincope*. Existe instalado un sistema de altavoces que le alerta y da instrucciones a seguir en caso de una eventualidad. De manera automática, permite a un enfermero que monitorea hablar con el paciente. La figura 8 ilustra el caso.

B. *Instanciacion de CARMiCLOC*

La persona hace una vida normal, y sabemos que CARMiCLOC posee datos desde hace dos años en la nube. Además, suponemos que los sensores están funcionando y generando data, y que se desea agregar un nuevo monitor de pulso cardiaco.

[2] Monitor Continuo de Glucosa
[3] Monitor Continuo de Presión Arterial
[4] Monitor Continuo de Pulso Cardiaco

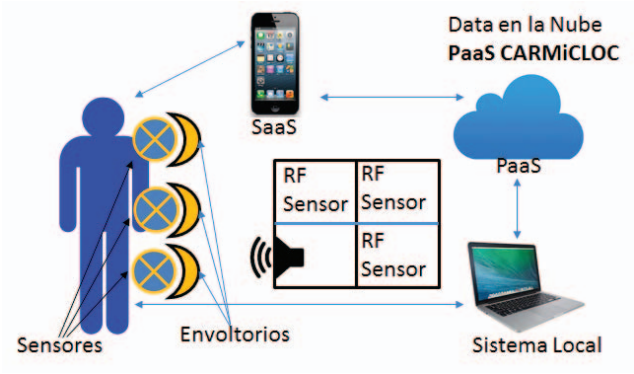

Figura 8. Caracterización del caso de estudio

1. Supongamos inicialmente que el monitor de pulso cardiaco está en su caja. Al encenderlo, se activa CARMiCLOC y detecta un nuevo sensor, invocando el servicio Se_1.
   a. Se activa la capa de Sensores. Se le da soporte a las fuentes de datos, se busca el envoltorio del nuevo sensor (ES). Si no se encuentra, se busca el Archivo Descriptor del Sensor en el Repositorio Local, de no estar se busca en la Nube.
   b. No se encontró el envoltorio del nuevo Monitor de Pulso. Se busca en el Archivo de Definición del Sensor, y se encuentra localmente en el mismo sensor y se crea el ES.
2. Monitoreo permanente por CARMiCLOC
   a. La capa de Descubrimiento y Semántica de Contexto esta siempre activa. Se activa la función de Procesamiento de data, se filtra la data eliminando la data obsoleta y no precisa. Se agrega la data con las mismas características.
   b. Se realizan las anotaciones y se obtiene un Objeto de Contexto preliminar que define una situación dada actual.
   c. Se coloca la información en la Base de conocimiento usando el servicio Se_2
   d. Se invoca a Se_3, este infiere información. Haciendo consultas en el histórico de Contexto obtiene un nuevo contexto y se genera un Descubridor de Semántico y de Contexto (Mantendrá cuales son los sensores a usar, y requerimientos del usuario)

TABLA II. COMPARACIÓN DE CARMICLOC CON OTROS TRABAJOS

Middleware	Arquitectura	Composición dinámica	Aplicaciones de Tiempo Real	Servicios llamados independientemente	Multi dominio
CAMPUS [10]	Web Services	Servicios	No	No	No
MARCHES [01]	Componentes	Componentes	Si	No	No
CA4IOT[19]	Web Services	Sensores	Si	No	Si
WEBINOS [11]	RPC		Si	No	Si
COCAMAAL[02]	Web Services	Semi-centralizado	Si	No	No
MAPCLOUS[07]	Web Services	Localización de Servicios (Locales/Nube)	-	No	-
CARMiCLOC	Web Services	Sensores, Localización de Servicios (Locales/Nube), Servicios	Si	Si	Si

e. Se sigue el monitoreo de la data con Se_1 y Se_3. La data se sigue guardando localmente por intervalos cortos, y luego pasada a la nube

3. Se descubre un Patrón de un posible sincope (lo hace el analizador a través del servicio razonador) y se reporta al planificador.
   a. El Planificador invoca al servicio Gestor de servicio y razonador para obtener información más precisa, la procesa, y llama al servicio que alerta,
   b. Se distribuye la alerta: se envía a través de un SMS y un mensaje de voz dentro de la casa por el altavoz, indicando que hay probabilidades de un Sincope. Para ello Se_7 Planifica e invoca al servicio Se_4. Así, los diferentes usuarios pueden tomar medidas para prevenir o mejorar su estado.

C. *Analísis de resultados*

Hemos visto como en el caso de uso cada uno de los servicios son usados e interactúan entre sí, para dar soporte al sistema de monitoreo de paciente, que debe ser consiente del contexto en el que se desenvuelve el usuario. La configuración y adaptaciones se realizan de manera autonómica, posibilitando que surjan contextos emergentes.

En el caso analizado vimos como las diferentes situaciones fueron manejadas flexiblemente por CARMiCLOC: la incorporación de nuevos dispositivos, la detección de patrones, el monitorio permanente y adaptación del contexto. Esta versatilidad es vital para las aplicaciones conscientes del contexto, en los entornos actuales tan dinámicos y cambiantes.

En la tabla II se observa la comparación de nuestro trabajo con otros, se destaca que CARMiCLOC posee servicios que pueden ser llamados independientemente. Es una middleware que puedo ser usado en diferentes dominios y por aplicaciones tiempo real.

Vemos que nuestro middleware se basa en los servicios web como modelo arquitectónica, como el resto, bajo una composición semántica distribuida y versátil (local y en la nube), además bajo un esquema de gestión de servicios independiente (lo que permiten ser usado individualmente por aplicaciones conscientes de contexto o como una plataforma de soporte completa a aplicaciones a los que esa habilidad se le quisiese añadir), todo ello en comparación con la previas propuestas, en los cuales no se cumplen simultáneamente esos aspectos, en particular esta última.

V. CONCLUSIONES

En este artículo se propuso la arquitectura de un Middleware Reflexivo Autonómico Consciente de Contexto basado en la computación de la Nube. Nuestro Middleware ofrece servicios que deben poseer una aplicación consciente de contexto; estos servicios pueden ser usados por una aplicación/servicio como un todo, o hacer uso de un servicio en específico, permitiendo que un servicio/aplicación que no posea contexto lo tenga y pueda ser consciente del mismo. CARMICLOC se diseña para poder ser usado dentro de cualquier dominio, lo que permitirá interoperatividad gracias a meta ontologías de dominio que estarán en la nube. Nuestro Middleware utiliza la computación autonómica para realizar la introspección y la adaptación haciendo uso del concepto de MAPE-K, en donde la base de conocimiento juega un papel importante pues es la base de la gestión del conocimiento y modelado del Contexto. Esta base de conocimiento puede estar alojada en la nube, lo que permitirá tener gran cantidad de información para ser procesada posteriormente, y así ofrecer un análisis de contexto mejor al usuario.

Actualmente, la implementación de CARMiCLOC se inició con los servicios de Adquisición y Pre configuración de Contexto (Se1) y Modelado de Contexto (Se2). Para el primer Servicio Se1 se realiza un descubrimiento y registro de servicios web a través de su WSDL (web Service Descripcion Language), de donde se obtienen los datos de salida y entrada del Servicio, con esto se crea el ADS, lo que permite tener la interfaz para tomar la data y crear los Descriptores de

Semántica y Contexto (DSC). El servicio Se2 se desarrolla dentro de la base de conocimiento como una Meta Ontología, en donde se define la información básica de un contexto, la cual se enriquece de acuerdo al dominio de la aplicación. El enriquecimiento y determinación del dominio de la aplicación se realiza a través de un motor de inferencia que usa patrones de dominios para identificar en cual se está.

El gestor autonómico será implementado mediante la coreografía de los servicios de CARMICLOC, realizando consumos internos de los servicios dentro del middleware. El Middleware se implementó haciendo uso de servicios web, con estos resolvimos los problemas de escalabilidad de los recursos, por lo que favorece su implementación en sistemas de gran tamaño. Para ello hemos integrado parte de los trabajos previos sobre Middleware, aplicaciones conscientes de contexto, y aplicaciones en la nube, para resolver los problemas presentados en este artículo,

En el caso de estudio hemos mostrado como la arquitectura propuesta se adapta a diferentes situaciones, y se implementa de manera simple de manera distribuida, lo cual es un avance para esta área de estudio.

En trabajos futuros se diseñaran los servicios faltantes, así como se implementaran sobre un bus de servicios. Este trabajo todavía se encuentra en progreso, pero los resultados iniciales son interesantes y prometedores.

RECONOCIMIENTO

Dr. Aguilar ha sido parcialmente financiado por el Proyecto Prometeo del Ministerio de Educación Superior, Ciencia, Tecnología e Innovación de la República del Ecuador.

REFERENCIAS

[1] E. Wei, A. Chan. “CAMPUS: A middleware for automated context-aware adaptation decision making at run time”, Pervasive and Mobile Computing, Volume 9, Issue 1, February 2013, Pages 35-56, ISSN 1574-1192, http://dx.doi.org/10.1016/j.pmcj.2011.10.002

[2] S. Liu, L. Cheng, “A context-aware reflective middleware framework for distributed real-time and embedded systems”, Journal of Systems and Software, Volume 84, Issue 2, February 2011, Pages 205-218, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2010.09.049.

[3] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. 2012. “CA4IOT: Context Awareness for Internet of Things”. In *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications* (GREENCOM '12). IEEE Computer Society, Washington, DC, USA, 775-782. DOI=10.1109/GreenCom.2012.128

[4] C. Ntanos, C. Botsikas, G. Rovis, P. Kakavas, D. Askounis, “A context awareness framework for cross-platform distributed applications”, Journal of Systems and Software, Volume 88, February 2014, Pages 138-146, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2013.10.018.

[5] A. Forkan, I. Khalil, Z. Tari, “CoCaMAAL: A cloud-oriented context-aware middleware in ambient assisted living”, Future Generation Computer Systems, Volume 35, June 2014, Pages 114-127, ISSN 0167-739X, http://dx.doi.org/10.1016/j.future.2013.07.009.

[6] C. Perera, A. Zaslavsky, P. Christen; D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey," Communications Surveys & Tutorials, IEEE , vol.16, no.1, pp.414,454, First Quarter 2014 doi: 10.1109/SURV.2013.042313.00197

[7] D. Zhang, H. Huang, Ch. Lai, Liang, Xuedong; Zou, Qin;Guo, Miny, "Survey on context-awareness in ubiquitous media", Multimedia Tools and Applications, Vol 67, Num 1, 2013, Springer US, Pag. 179-211, http://dx.doi.org/10.1007/s11042-011-0940-9

[8] M. Knappmeyer, S.L. Kiani, E.S. Reetz, N. Baker, R. Tonjes, , "Survey of Context Provisioning Middleware," *Communications Surveys & Tutorials, IEEE* , vol.15, no.3, pp.1492,1519, Third Quarter 2013, doi: 10.1109/SURV.2013.010413.00207

[9] M. Zaigham; N. NayyabZia; P. Davy, B. Yolande, "Cloud Computing: A Mobile Context-Awareness Perspective", Cloud Computing: Computer Communications and Networks, Springer London, 2013, pag 155-175, http://dx.doi.org/10.1007/978-1-4471-5107-4_8

[10] M. Sama, S. Elbaum, F. Raimondi, D. Rosenblum, Z. Wang. 2010. Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification. IEEE Trans. Softw. Eng. 36, 5 (September 2010), 644-661. http://dx.doi.org/10.1109/TSE.2010.35

[11] M.R. Rahimi, J. Ren, C. Liu,;Vasilakos, A. Venkatasubramanian, , "Mobile Cloud Computing: A Survey, State of Art and Future Directions", Mobile Networks and Applications, Vol. 19, Num 2, 2014, Pag. 133-143, Springer US, http://dx.doi.org/10.1007/s11036-013-0477-4

[12] P. Maes, “Concepts and Experiments in Computational Reflection”, Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, pp. 147-155, 1987.

[13] IBM Corporation. "An architectural blueprint for autonomic computing". Autonomic Computing”, Fourth Edition, http://www.ginkgo-networks.com/IMG/pdf/AC_Blueprint_White_Paper_V7.pdf, 2006.

[14] T. Erl, Ricardo R. Puttini, Z. Mahmood. 2013. *Cloud Computing: Concepts, Technology & Architecture* (1st ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.