Universidad Nacional
Federico Villarreal

Vicerrectorado de
INVESTIGACIÓN

EUPG
ESCUELA UNIVERSITARIA DE POSGRADO

ARQUITECTURA DE SOFTWARE PARA EL DESARROLLO DE APLICACIONES SENSIBLES AL CONTEXTO – PROPUESTA

Tesis para optar el Grado Académico de Doctor en Ingeniería de Sistemas

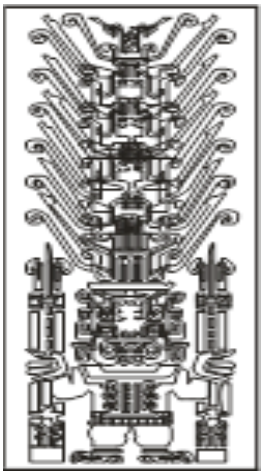
AUTOR (A)
Iparraguirre Villanueva, Orlando Clemente

ASESOR (A)
Andrade Arenas, Laberiano Matias

JURADO
Solís Fonseca, Justo Pastor
Mayhuasca Guerra, Jorge V.
Contreras Aranda, Santiago E.

Lima - Perú
2017

UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
ESCUELA UNIVERSITARIA DE POSGRADO



TESIS

“ARQUITECTURA DE SOFTWARE PARA EL DESARROLLO DE APLICACIONES SENSIBLES AL CONTEXTO – PROPUESTA”

PRESENTADO POR:
ORLANDO CLEMENTE IPARRAGUIRRE VILLANUEVA

PARA OPTAR EL GRADO ACADÉMICO DE:
DOCTOR EN INGENIERÍA DE SISTEMAS

LIMA – PERÚ
2017

Resumen

Durante el desarrollo de esta Tesis Doctoral se ha realizado una propuesta tecnológica de Arquitectura de Software Para el Desarrollo de Aplicaciones Sensibles al Contexto. En respuesta a la problemática que generalmente se encuentran los desarrolladores de software, al momento de establecer comunicación con el contexto sin la participación de una acción humana, las arquitecturas de software para el desarrollo de este tipo de aplicaciones aún están en propuestas, en el mejor de los casos están en proyectos y aún están en proceso de madurez. Este problema se enmarca dentro de las áreas de investigación de: desarrollo de aplicaciones móviles y las fuentes de información del contexto. Tradicionalmente la integración entre estos dos campos se ha desarrollado de forma independiente, que aíslan a las aplicaciones móviles de conocer los procedimientos de acceso a la información del contexto, limitando la interacción con otras aplicaciones de forma autónoma. Esta problemática coincide con el actual escenario de la inteligencia ambiental, en donde se busca que en un futuro cercano los objetos inteligentes como los teléfonos móviles "internet de las cosas" sean capaces de interactuar de forma autónoma con el contexto y obtener información para suministrar a otros objetos y personas creando una interacción inteligente y natural con el usuario. En la presente Tesis se ha diseñado una arquitectura de software para el desarrollo de aplicaciones sensibles al contexto, para el cual se ha tenido en cuenta el actual escenario tecnológico. Para ello, se ha realizado un análisis de propuestas tecnológicas ya existentes "Arquitecturas y patrones" más relevantes para el diseño de la propuesta, asimismo de las principales plataformas de suministro de información contextual. Luego del análisis se ha determinado utilizar el Patrón "Event-Control-Action-ECA" como plataforma base para el diseño de la propuesta. A esto se ha añadió una extensión denominado "dominioX-DX" que complementa y refuerza la propuesta, creando la arquitectura "ECA-DX" idónea para el desarrollo de aplicaciones sensibles al contexto, permitiendo procesar e integrar información de diferentes fuentes híbridas y por consiguiente diseñar aplicaciones de interacción autónoma - sensibles al contexto.

Palabras claves: Contexto, Arquitectura, Patrón, Plataforma, Software, fuentes de información, información contextual.

Abstract

In this Doctoral thesis, it was elaborated a technological proposal for an architectural software aimed at developing sensitive context applications. Its purpose was to tackle the problem that software and developer programmers usually have to face at the moment of establishing a communication with the context without human participation, though the development of this kind of applications is still a proposal and in its process of reaching maturity. This problematic issue is placed under research areas such as the development of mobile applications and context information sources. Traditionally the research in the above-mentioned areas has been developed independently, isolating mobile applications from the process of accessing environmental information, limiting the interaction with other autonomous applications. At the same time, this problematic issue coincides with the actual environmental intelligence scenario, in which it is sought to achieve that intelligent objects such as mobile phones and other appliances may interact with the environment in an autonomous way, obtaining data which are able to provide information to other objects or people, thus creating an intelligent and natural interaction with the user. In this research, it was designed an architectural software for the development of applications sensitive to the environment, taking into account the actual technological scenario. In order to achieve that, the most relevant technological proposals in existence: "architectures and patterns" were analyzed, as well as the main platforms which provide contextual information. After the analysis, it was established to use the pattern "Event – Control- Action- ECA" as basic platform in order to design the proposal, to which it was added an extension called "dominoX-DX" that complements and strengthens the proposal. Finally, the architecture "ECA–DX" was created, which is suitable to develop applications sensitive to the context and apt to process and integrate information from different hybrid sources, thus enabling the design of autonomous interactive applications, sensitive to the context.

Keywords: Context, Architecture, Pattern, Platform, Software, Information sources, Contextual information

INDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

El desarrollo de software sensible al contexto comprende la construcción de aplicaciones que se ejecutan en entornos dinámicos, con la capacidad de ejecutarse en cualquier momento, en cualquier lugar, y en dispositivos capaces de requerir la mínima atención posible por parte del usuario(Kun, Shumao, Manooch, & Nektarios, 2005). Una característica principal de las aplicaciones sensibles al contexto, es que deben ser soportados por dispositivos compatibles con este tipo de aplicaciones, de tal forma que permita obtener información relevante para adaptar el comportamiento de la aplicación dinámicamente en función de su contexto.

El desarrollo de aplicaciones sensibles al contexto aun es complejo, dado que además de la adquisición de la información del contexto requiere el adecuado procesamiento de dicha información. A la fecha se han desarrollado de forma genérica distintas herramientas para la implementación de aplicaciones sensibles al contexto como son: *frameworks*, interfaces de programación de aplicaciones y kits de desarrollo de software. Teniendo en cuenta los distintos esfuerzos analizados y estudiados en esta línea, se observa que la información contextual es un aspecto que los desarrolladores de software recientemente están empezando a tener en cuenta en sus proyectos, aunque en esta área se carezca de estándares adoptados.

En aras de alcanzar una interacción más natural entre el usuarios y los dispositivos, la gran mayoría de estudios e investigaciones que se han desarrollado en esta línea han tratado de abordar este objetivo con la introducción de la comunicación multimodal, dotando así a los usuarios la libertad de poder elegir la forma o formas de interacción que cada uno considere más apropiada (Izquierdo, 2013).

La presente Tesis tiene como objetivo principal contribuir al diseño y desarrollo de aplicaciones móviles, proponiendo el modelo "ECA-DX" de Arquitectura de Software para el desarrollo de Aplicaciones Sensibles al Contexto, considerando la problemática asociada a los desarrolladores, al no contar con herramientas idóneas para el desarrollo de aplicaciones sensibles al contexto. En donde la información contextual pueda ser proporcionada por redes de sensores y otras fuentes avanzadas de información, y, a la vez pueda integrarse con fuentes proveedoras de información para su consumo por otras entidades. Bajo este escenario se ha definido los objetivos específicos para el desarrollo de la propuesta:

1. Analizar los diferentes patrones, middlewares y arquitecturas de software relacionadas con el desarrollo de aplicaciones contextuales en dispositivos móviles, entre otras tecnologías asociadas a la inteligencia ambiental.

2. Integrar la extensión “dominioX- DX” a la arquitectura “Event-Control-Action-ECA” para descubrir la funcionalidad que es necesaria para llevar acabo la integración con las fuentes de información del contexto.

3. Crear una aplicación como prototipo para demostrar las abstracciones del contexto, modelado de la situación y la plataforma de manejo de contexto, usando las herramientas como: Jess, jre, jdk, WYSIWYG, eclipse, entre otros componentes necesario para la implementación

Se pretende abordar los objetivos anteriores centrando la investigación dentro del entorno presentado. Considerando cada una de las particularidades de la problemática para el diseño de la propuesta. Puesto que la información para el desarrollo de aplicaciones móviles sensibles al contexto, se clasifican en tres grupos: computacional (procesadores, dispositivos de entrada y salida, red, recursos de interacción, conectividad), usuario (localización, preferencias/perfil del usuario, situación social) y físico (iluminación, nivel de ruido). Por lo tanto, una aplicación adaptable o adaptativa se reconfigura por si misma dinámicamente de acuerdo a su contexto de uso. Éste a su vez facilita la utilización del ambiente en una forma que garantice la calidad del servicio al usuario(Canelón, Losavio, Matteo, & Chirinos, 2009). Actualmente el uso masivo de este tipo de aplicaciones y servicios en dispositivos móviles ha suscitado considerables iniciativas de investigación y desarrollo en la comunidad académica, científica y en el sector privado, orientados a implementar esta tecnología en variados ámbitos de la vida de los usuarios(Cangrejo, 2014). Según(Weiser, 1991b), uno de los principios fundamentales de la computación ubicua es que los dispositivos dejen atrás su comportamiento de artefactos que el usuario puede utilizar, y pasen a tener un comportamiento activo, con consciencia del entorno en el que se encuentran, en definitiva, un comportamiento inteligente.

La distribución de esta Tesis ha sido estructurada como sigue:

- En el capítulo 1 se brinda un panorama general de la realidad problemática del desarrollo de aplicaciones sensibles al contexto con las arquitecturas de software tradicionales, antecedentes, formulación del problema, justificación, alcance y limitaciones. Asimismo, se presenta los objetivos que se pretenden lograr en la presente tesis.

- En el capítulo 2 se analiza las teorías generales sobre arquitecturas de software como: arquitectura cliente servidor, modelo vista controlador, orientada a servicios (SOA), etc. Seguidamente se identifican los problemas de interacción de los dispositivos móviles con el contexto. Y finalmente se presenta las bases teóricas y específicas como: la sensibilidad de las aplicaciones al contexto, herramientas para el

desarrollo de aplicaciones contextuales en dispositivos móviles. Y se introducen los patrones y arquitecturas específicas para este tipo de aplicaciones como: la arquitectura Event-Control-Action (ECA), patrón de manipulación del contexto, patrón de fuentes de contexto, gestores de jerarquía y patrón de acciones. Asimismo, se presenta el marco conceptual de la presente investigación

- En el capítulo 3 se presenta el tipo de investigación, diseño, variables y técnicas de investigación como: el procesamiento y análisis de datos. Asimismo, en este apartado se demuestra la viabilidad de la arquitectura para el desarrollo de aplicaciones sensibles al contexto. Con el fin de demostrar la viabilidad se ha construido un prototipo de aplicación usando como escenario la asistencia médica de un paciente con epilepsia. Para ello se ha modelado el contexto, la situación, y se ha diseñado la estructura de la aplicación en un contexto del paciente.
- En el capítulo 4 se presenta los resultados de la aplicación prototipo, para ello se ha implementado los componentes utilizando Java como el lenguaje de programación, Java RMI para permitir la distribución de los componentes, y Eclipse como el entorno de desarrollo. Se ha implementado en el prototipo todos los componentes que se identificaron en el trabajo de diseño, las fuentes de contexto, los gestores de contexto, el controlador, el revolvedor de acciones y los servicios de acción.
- Por último, en el capítulo 5 se presenta la discusión, se exponen las conclusiones y recomendaciones desarrolladas en esta tesis Doctoral. Además, se presenta los trabajos futuros que pueden desencadenarse como resultado de este estudio.

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA

1. Antecedentes

Para el desarrollo del proyecto de investigación se hizo necesaria la revisión bibliográfica y trabajos de investigación en cuanto a modelos arquitecturales de Software para el desarrollo de aplicaciones contextuales en dispositivos móviles, con el fin de crear un grupo de antecedentes validos relacionados con las variables de estudio. Esta investigación se desarrolla de una manera más comprensible si se utilizan distintos modelos, a continuación, se citan los antecedentes de los modelos arquitecturales para aplicaciones contextuales en dispositivos móviles.

Se comienza citando a, (Morillo, 2011). Desarrolló una investigación en el cual se creó un *Middleware para aplicaciones ubicuas en dispositivos móviles*, que facilita la comunicación de las aplicaciones entre sí, con el objetivo de poder gestionar la información del entorno del usuario de una manera eficaz y segura, y concluye con lo siguiente:

Respecto al middleware que se ha desarrollado, proporciona no sólo el intercambio de información entre aplicaciones, sino la disminución de los recursos necesarios para ejecutar dichas aplicaciones contextuales. Debido a que el desarrollo en dispositivos móviles debe tener en cuenta las limitaciones de estos sistemas, especialmente en términos de batería y capacidad de cómputo, se hace necesario ahorrar recursos de una manera masiva por parte de las aplicaciones. Esto permite conseguir mediante el uso del middleware, dado que el procesamiento para la obtención de información procedente de los sensores sólo será realizado una vez por una aplicación, la cual difundirá sus resultados a través del middleware con el resto de aplicaciones que están ejecutándose en el sistema.

(Gómez, 2011). Desarrolló un proyecto de investigación “Modelo Arquitectural para Aplicaciones Móviles Usando el Enfoque de Líneas de Producción Dinámica de Software”, el propósito de esta investigación radica en proponer un modelo arquitectural para aplicaciones móviles. Para el cual define como modelo la infraestructura que soporta los requisitos comunes que comparten los productos de software de la línea de producción, se hace énfasis en que la línea sea dinámica precisamente por la presencia de la adaptabilidad y se pretende medir los atributos de calidad de los productos a través de un modelo de calidad, que se obtiene en etapas tempranas del desarrollo.

Asimismo, el autor señala que en la ingeniería de las líneas de producción de software se distinguen dos procesos principales: a) Ingeniería del dominio: cuyos productos son componentes de software, requisitos reutilizables y configurables, modelos de análisis y diseño y b) Ingeniería de aplicación: del cual se derivan productos individuales de la

familia de productos, construidos usando un subconjunto de artefactos de software compartidos.

Finamente el autor en su investigación llega a las siguientes conclusiones: Los sistemas de software están sujetos a la evolución, a las actualizaciones dinámicas, debido a cambios en los requisitos del cliente, contexto de uso o la tecnología.

Una línea de Producción Dinámica de Software se diseña para apoyar explícitamente, los escenarios de actualización dinámica. Como consecuencia, la variabilidad y el modelo de variabilidad asociado a estos sistemas no son estáticos, Gracias al proceso para la ingeniería del dominio basado en calidad de software InDsCaS, es posible obtener una arquitectura base con calidad para una familia de productos.

(Haya, 2006). El planteamiento principal de esta investigación se centra en estudiar la manera más eficaz de gestionar el contexto dentro de un entorno inteligente, en esta investigación se considera el contexto en el mismo sentido que le dan los trabajos realizados en aplicaciones sensibles al contexto. Esta área engloba aplicaciones y dispositivos que consideran como una entrada más del sistema la información sobre las circunstancias bajo las cuales operan. Estas circunstancias pueden ser la localización, la tarea que está realizando el usuario, otros recursos que se encuentran cerca y/o condiciones ambientales del entorno.

El alcance del estudio queda acotado a las aplicaciones sensibles al contexto que operan dentro de un entorno inteligente. Este consiste en una infraestructura restringida por unas barreras físicas y compartidas por un conjunto de aplicaciones, dispositivos y personas. El adjetivo inteligente se emplea para indicar la habilidad de adquirir información de forma autónoma y de emplearla para adaptarse a las necesidades de sus ocupantes. El entorno se convierte en una aplicación sensible al contexto más, contribuyendo activamente en la interacción con el usuario.

Los autores concluyen en su investigación: Que la Computación Ubicua se presenta como una nueva área de investigación que nace como una rama de los sistemas distribuidos y la computación móvil incorporando requerimientos propios de la interacción hombre-máquina. La fusión de estas dos áreas se resume en dos objetivos propuestos por la Computación Ubicua: ubicuidad y transparencia.

La ubicuidad lleva a que la capacidad de computación deje de ser dominio exclusivo de los ordenadores personales. Se propone desplegar multitud de dispositivos interconectados, entre los que se incluyen objetos de la vida cotidiana. Este cambio también afecta al software dando como resultado nuevas aplicaciones que se sustentan en Novedosas plataformas de software. Este avance, hasta el momento, se ha

materializado en mejoras en el hardware, quedando más camino que recorrer en la ubicuidad del software. La transparencia se aplica a la interacción entre el usuario y el sistema. El objetivo final persigue conseguir una interacción tan fluida como la comunicación entre personas. Para alcanzar este ambicioso objetivo se requiere de pequeños pasos: Intermedios que vayan añadiendo mayor transparencia a las interfaces actuales. Entre los múltiples pasos a realizar se señalan dos íntimamente relacionados.

Primero, crear aplicaciones que también tomen la iniciativa en el diálogo con el usuario, mayoritariamente, el papel de los ordenadores en la interacción con el usuario es pasivo. En los sistemas ubicuos la pro actividad se coloca en un primer plano. La iniciativa del dialogo se decide según la situación y las interfaces disponibles en cada momento.

Segundo, para poder alcanzar el propósito anterior se precisa de capturar la Información implícita en la interacción entre el usuario y el ordenador. El empleo del contexto permite construir aplicaciones que reaccionan activamente, ya que se pueden detectar situaciones en las cuales se requiere una determinada acción, pero el usuario no la expone explícitamente. El contexto es parte fundamental de la comunicación humana, y como tal tiene que ser considerado por una aplicación que busque una interacción transparente.

(Imen & Faouzi, 2013). Desarrollaron una Arquitectura de Software para entornos inteligentes. Los investigadores han estado profundamente interesados en la disposición óptima para la gestión y el modelado de la capa del usuario, por lo tanto, los primeros sistemas estaban involucrados en gran parte en la interfaz entre la gestión de los datos contextuales y las aplicaciones generalizadas. El objetivo principal de la arquitectura [CIS1] es recoger datos, para modelar y para proporcionar la información contextual. La arquitectura [SOCAM2] fue específicamente propuesta para convertir los espacios físicos en espacios semánticos y pueda ser compartido entre los servicios sensibles al contexto. Por otra parte, el componente clave de la arquitectura [Cobra3] es responsable de la gestión y trasmisión de la información contextual mientras se mantiene el modelo contextual.

La investigación describe la arquitectura de un sistema generalizado que permite la adaptación dinámica a los retos del entorno inteligente. Ayudar a los usuarios para hacer frente a su actividad en tiempo real es el objetivo principal del enfoque propuesto. Para esto, la estructura interna del sistema está fuertemente y principalmente influenciada por la actividad del usuario. Esto se puede lograr mediante la introducción de los modelos dinámicos, que puede ser construido en tiempo real para interactuar con el medio

ambiente. Este resultado es de considerable importancia por su propia cuenta, y es particularmente relevante para inferir las necesidades actuales de estos modelos dinámicos como una consecuencia una vez que el sistema llegó a crear el modelo más adecuado de la actividad actual. El sistema de hecho podría tener una representación adecuada y en tiempo real del usuario en una situación particular. Por lo tanto, se puede deducir el modelo adecuado o construir un modelo en tiempo real de acuerdo con el análisis de la interacción del sistema del usuario y el contexto actual de uso.

Finalmente, los autores concluyen que la arquitectura desarrollada no proporciona suficiente capacidad de adaptación a una serie de factores claves como es en el aspecto de tiempo real. De hecho, la adaptación en tiempo real de la información proporcionada para que coincida con las necesidades, preferencias y comportamientos distintos de los diferentes usuarios todavía constituye el desafío que aún no es abordado adecuadamente en estos sistemas.

(González, Alejandres, & González, 2015). Desarrollaron una arquitectura de un Sistema de recomendación semántico sensible al contexto para entornos tipo campus. En este trabajo describen el uso de la arquitectura en el desarrollo del sistema Find-It, en el cual se consideran factores contextuales para recomendar personas, objetos de conocimiento, lugares, actividades, eventos, recursos tecnológicos y servicios a los visitantes de un centro de investigación. Para filtrar los ítems recomendables, de acuerdo a una situación específica, la arquitectura define un módulo de inferencia que considera diferentes dimensiones contextuales modeladas en una ontología, el grado de novedad de los ítems, la percepción de las recomendaciones por parte de los usuarios y las condiciones contextuales en que estas fueron recomendadas en ejecuciones previas. También se define una interfaz cliente móvil adaptable a las características del usuario y de su dispositivo, sobre la cual se ejecutan mecanismos implícitos y explícitos para identificar el grado de satisfacción con respecto a las recomendaciones recibidas y la calidad de las mismas. La experiencia obtenida en el desarrollo de Find-It indica que tiene un desempeño satisfactorio, por lo que se considera adecuado continuar implementando la arquitectura propuesta en el desarrollo de otros sistemas de recomendación organizacionales. Finalmente, los autores, concluyen exponiendo los componentes de una arquitectura para SRSSC organizacionales, los cuales ya han sido implementados en el proyecto Find.It. En la propuesta, las dimensiones contextuales y los elementos organizacionales, son modelados en una red de ontologías, que nos ha permitido mantener una Base de Conocimientos que considera múltiples aspectos para la inferencia de recomendaciones.

Según (González et al., 2015), en el artículo “Arquitectura de un sistema de recomendación semántico sensible al contexto para entornos tipos campus”. Analizan sus principales componentes para explorar sensibilidad del contexto de la composición de servicios de apoyo. A la vez los autores manifiestan que se han desarrollado un sin número de aplicaciones para demostrar la utilidad de la tecnología sensible al contexto, por ejemplo, “ParcTab”, fue uno de los primeros sistemas desarrollados para ofrecer un marco modular sensible al contexto, y con componentes reutilizables, permitiendo a los programadores construir más fácilmente las aplicaciones sensibles al contexto interactivo basados en sensores. Estos sistemas no tienen un modelo de contexto abierto porque el contexto se describe en base a la programación orientada a objetos y por lo que, la información está fuertemente acoplado al modelo de programación. El presente trabajo concluye presentando una arquitectura sensible al contexto orientada a servicios que utiliza un modelo de contexto ontológico para proporcionar información personal y contextual y apoyar la composición de los servicios sensibles del contexto. Las dos principales contribuciones del trabajo son la utilización conjunta de una semántica modelo de contexto (SeCoM), para describir y explorar la expresión de la información contextual, junto con el apoyo de la composición dinámica, de los servicios sensibles al contexto del usuario.

(Manuel & Coello, 2013). En su tesis de Master, presenta una Plataforma Middleware para aplicaciones Contextuales en Plataformas móviles. Este trabajo se centra en estudiar cómo se pueden obtener atributos contextuales definidos con las tecnologías móviles actuales. Para alcanzar el objetivo se ha realizado un estudio sobre el uso de las tecnologías móviles. Concretamente se analizan las posibilidades de las tecnologías móviles a los sistemas de recomendación basados en contexto. De esta forma este se enmarca en la clasificación y obtención de la información contextual a través de dispositivos móviles para posibilitar el desarrollo de sistemas de recomendación contextuales.

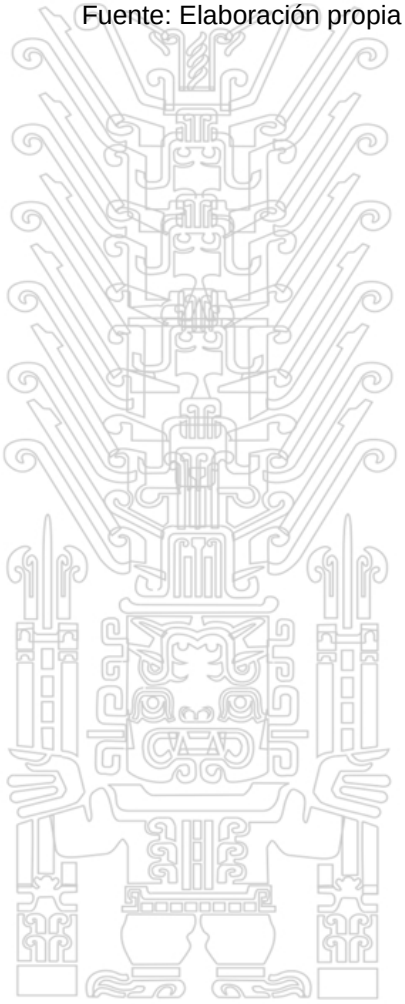
Finalmente, el autor concluye que, para identificar qué información contextual se puede obtener a través de los dispositivos móviles se han estudiado distintas plataformas móviles existentes en la actualidad, así como la tendencia que ha sufrido el mercado en este campo. Se han encontrado librerías similares, en concreto se ha estudiado ContextPhone, la cual se desarrolló para la plataforma Symbian y ofrece una gran cantidad de información para el desarrollo de aplicaciones en dicha plataforma. Por ello en este trabajo se ha estudiado utilizar los estándares de tecnología web para el desarrollo en dispositivos móviles con el fin de que puedan ser reutilizados en un futuro sin la necesidad de desarrollar completamente una nueva librería para otras plataformas.

Tabla 1: Resumen de Antecedentes

Vari	Antecedente	Objetivos	Aportes
Aplicaciones Contextuales en Dispositivos Móviles	(Soria 2011). Desarrolló una investigación en el cual se creó un "Middleware para aplicaciones ubicuas en dispositivos móviles"	Gestionar la información del entorno del usuario de una manera eficaz y segura.	Contribuye no sólo el intercambio de información entre aplicaciones, sino la disminución de los recursos necesarios para ejecutar dichas aplicaciones contextuales.
	(Vargas -2011). Desarrolló un proyecto de investigación "Modelo Arquitectural para Aplicaciones Móviles Usando el Enfoque de Líneas de Producción Dinámica de Software".	Medir los atributos de calidad de los productos a través de un modelo de calidad, que se obtiene en etapas tempranas del desarrollo.	Contribuye con un diseño explícitamente para apoyar los escenarios de actualización dinámica. Como consecuencia, la variabilidad y el modelo de variabilidad asociado a estos sistemas no son estáticos.
	Pablo A., Haya Coll (2009), En su investigación presenta un *Middleware* para tratamiento de Información Contextual en Entornarnos Inteligentes.	Gestionar la información de manera más eficaz el contexto dentro un entorno inteligente.	Contribuye al crear un middleware para gestionar la información del contexto.
	Imen Ismail & Faouzi Moussa, Desarrollaron una Arquitectura de Software para entornos inteligentes, en el año 2013.	Recoger datos, para modelar y para proporcionar la información contextual.	Introducción de los modelos dinámicos, que puede ser construido en tiempo real para interactuar con el medio ambiente.
	(González et al., 2015). Arquitectura de un Sistema de recomendación semántico sensible al contexto para entornos tipo campus.	Definir una interfaz cliente móvil adaptable a las características del usuario y de su dispositivo.	Implementación del proyecto Find.It, con las dimensiones contextuales y los elementos organizacionales.
	João P. Sousa y Otros. En su artículo "Composición del contexto de servicios móviles conscientes utilizando un modelo de contexto semántico" describe una arquitectura	Explorar la sensibilidad del contexto de la composición de servicios de apoyo.	Contribuye con la creación de una arquitectura sensible al contexto orientada a servicios que utiliza un modelo de contexto ontológico.

que permite la creación de servicios sensibles al contexto sobre la marcha, en el año 2009.		
Manuel y Coello (2013). En su tesis de Master, presenta una Plataforma Middleware para aplicaciones Contextuales en Plataformas móviles.	Estudiar cómo se pueden obtener atributos contextuales definidos con las tecnologías móviles actuales	Creación de un Middleware para aplicaciones contextuales en plataformas móviles.

Fuente: Elaboración propia



2. Planteamiento del Problema

Es indiscutible que la tecnología ha contribuido en el desarrollo de la sociedad, ha cambiado el estilo de vida de las personas, la forma de percibir la misma, sin embargo, el modelo de interacción entre los ordenadores personales y dispositivos móviles requiere toda la atención del usuario.

Uno de los puntos débiles de las aplicaciones actuales es la insensibilidad ante los cambios del contexto. A causa de las actuales arquitecturas de desarrollo de software, como: cliente servidor, arquitectura de tres niveles, modelo vista controlador, orientada a servicios, dirigida por eventos, entre otras, no se adaptan para este tipo de desarrollo de aplicaciones.

El modelo de interacción entre los dispositivos móviles actuales requiere de una acción humana para ejecutar un evento. Actualmente éste modelo genera una distracción en los usuarios, del mismo modo el aprendizaje de la tecnología está haciendo que el usuario dedique tiempo para dicho aprendizaje, la interacción con la tecnología debe ser inconsciente (Weiser, 1993).

Los avances desarrollados son considerables con respecto a la modelización del usuario y la adaptación según perfiles y preferencias, sin embargo, las aplicaciones siguen mostrando el mismo comportamiento independiente de la situación en que se encuentre el usuario. Una de las características de la nueva generación de ordenadores móviles, es que, la maquina desaparece de la vista, y para ello un punto débil de las aplicaciones es la insensibilidad ante los cambios del contexto. El principal problema con lo que se encuentran los desarrolladores de software, es al momento de establecer la comunicación con el contexto sin la participación de una acción humana, las arquitecturas de software para el desarrollo de aplicaciones basadas en el contexto, no existen, en el mejor de los casos, están en proyectos y aún no han madurado. El desarrollo de aplicaciones basadas en el contexto depende del avance en los diferentes campos que éstos involucran, los cuales incluyen hardware, software, redes, protocolos, sensores y estándares; sin embargo, cada uno de estos campos avanza a su propio ritmo y genera sus propios productos, haciendo necesario establecer formas apropiadas para su interrelación, es decir, pensarlo como un software integrado y no como partes individuales (Mondragón & Solarte, 2004) .

Aun los dispositivos de última generación, requiere de una acción humana para gestionar la información del contexto tales como la localización, temperatura del ambiente, humedad, aceleración, entre otras. Con una aplicación basada en la arquitectura propuesta, permitiría tomar decisiones oportunas y automatizadas. Es por

el cual hace imprescindible establecer una arquitectura de desarrollo de aplicaciones sensibles al contexto. De tal forma que se obtenga un estándar para el desarrollo de aplicaciones de este tipo, de esta forma se tendrá aplicaciones con mayor eficiencia en la gestión de la información de forma automatizada.

Formulación de Problema

Problema Principal

¿En qué medida el diseño de una arquitectura de software para el desarrollo de aplicaciones contextuales en dispositivos móviles, permitirá un desarrollo estándar para la implementación de aplicaciones sensibles al contexto?

Problemas Secundarios

¿En qué medida diseñar una arquitectura de desarrollo de aplicaciones en capas, permitirá un desarrollo ordenado, eficaz y estándar para la implementación de sistemas sensibles al contexto?

¿En qué medida proponer una aplicación como prototipo para demostrar las abstracciones del contexto, modelado de la situación y la plataforma de manejo de contexto, permitirá evaluar el desempeño y la escalabilidad la arquitectura propuesta?

3. Objetivos

Objetivo General

Diseñar una arquitectura de Software para el desarrollo de aplicaciones sensibles al contexto, que permita un desarrollo estándar para la implementación de sistemas basados en el contexto**.**

Objetivos Específicos

1. Analizar los diferentes patrones, middlewares y arquitecturas de software relacionadas con el desarrollo de aplicaciones contextuales en dispositivos móviles, entre otras tecnologías asociadas a la inteligencia ambiental.
2. Integrar la extensión "DX" a la arquitectura "Event-Control-Action-ECA" para descubrir la funcionalidad que es necesaria para llevar acabo la integración con las fuentes de información del contexto.
3. Crear una aplicación como prototipo para demostrar las abstracciones del contexto, modelado de la situación y la plataforma de manejo de contexto.

4. Justificación

La presente investigación fundamenta su justificación en la obtención de una arquitectura de software que garantice la adaptabilidad de las aplicaciones móviles basadas en el contexto, y su vez que ayude a los desarrolladores de software solucionar el problema al momento de estructurar y comunicar las aplicaciones, así como la necesidad de obtener la información el contexto mediante el procesamiento de las señales provenientes de los diferentes sensores presentes en el dispositivo móvil.

La arquitectura con las que se desarrolla las aplicaciones tradicionales, dista mucho de las arquitecturas empleadas para desarrollar aplicaciones basadas en el contexto. Actualmente la información no solo proviene de bases de datos o servidores, sino de una serie de sensores y procesos que adaptan la información que obtienen del contexto. La arquitectura propuesta busca asegurar el proceso de adaptación de los componentes que obtiene información del contexto con la información proporcionada por bases de datos en tiempo de ejecución. Las características fundamentales de una aplicación contextual son: capacidad, reconfiguración dinámica, modularidad, extensibilidad y portabilidad(Rodríguez & Holgado, 2008). Es por ello la presente investigación busca que la arquitectura propuesta permita implementar aplicaciones móviles sensibles al contexto.

Se espera que los resultados de la presente investigación sean de gran importancia para la industria del software como parte de los activos de software en los ambientes de desarrollo de aplicaciones móviles, y asimismo que sirva como referencia para el desarrollo de nuevos proyectos.

Finalmente, es el usuario quien experimentará los cambios de optimización de recursos en los dispositivos móviles.

5. Alcance y Limitaciones

El alcance de la presente investigación se ajusta a la presentación de una arquitectura de Software para el desarrollo de Aplicaciones Sensibles al Contexto, que garantice la adaptabilidad y la reconfiguración dinámica de los componentes en las aplicaciones. En el proceso se examinará los diferentes tipos de arquitectura que aporten solución al problema para luego crear el modelo arquitectural, finalmente será evaluado para validar el modelo propuesto.

Con respecto a las limitaciones, se observan brechas en las actividades que guían la obtención de la arquitectura propuesta “Arquitectura de Software para el desarrollo de Aplicaciones Sensibles al Contexto”, careciendo de componentes necesarios en muchos casos la utilización de tecnología emergente asociado a la línea de desarrollo

de aplicaciones. Estas limitaciones son propias del desarrollo de aplicaciones contextuales.

6. Definición de Variables

Tabla 2: Definición de Variables

VARIABLE	DIMENSION	INDICADOR
Arquitectura de Software	Arquitectura	Modelo de arquitecturas. Dominio omnipresente. Ontologías. Cambios de contexto. Estructura. Componentes Manejo de sensores
	Soporte y Diseño	Soporte para generación de información. Motor de reglas. Implementación de componentes Proveedores de diseño. Procesador del contexto.
Aplicaciones sensibles al contexto	Interacción.	Vocabulario común. Software heterogéneo. Medulares y escalables. Fuentes de información de manera flexible. Herramientas accesibles. Localización. Inexactitudes. Generación de librerías específicas. Productividad. Comunicación. Puntos de interacción. Disponibilidad del controlador. Colaboración de componentes. Flexibilidad y extensibilidad. Tiempo de ejecución

CAPITULO II

MARCO TEORICO

Teorías generales

Arquitectura

La arquitectura es el arte y la técnica de proyectar, diseñar, construir y modificar el hábitat humano, incluyendo todo tipo de estructuras arquitectónicas y urbanas.

La arquitectura de sistemas basados en software se resume como: la organización fundamental de un sistema representada por sus componentes, sus relaciones entre ellos y con su entorno, y los principios que gobiernan su diseño y evolución. Un esquema de arquitectura es un conjunto de herramientas que puede ser utilizado para desarrollar un amplio espectro de diversas arquitecturas. Estos esquemas deben describir una metodología para la definición de un sistema de información en términos de un conjunto de bloques constituidos que encajen entre si adecuadamente, contener un conjunto de herramientas, proveer un vocabulario común e incluir una lista de estándares (The Open Group Architecture Framework, 2015).

Arquitectura de Software

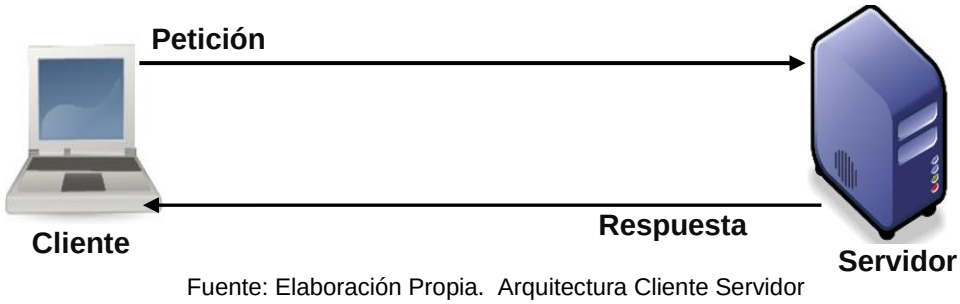
La arquitectura es la organización fundamental de un sistema que incorpora en sus componentes, sus relaciones con el entorno y los principios que conducen su diseño y evolución (Tahuiton, 2011). La arquitectura de un sistema debe definir los elementos que formaran el software. Tales elementos definen como el software será fragmentado en pedazos más pequeños y se define como el software será interpretado.

(Ramos & Lozano, 2000). La arquitectura de software es la representación de alto nivel de la estructura de un sistema o aplicación, que describe las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición y las restricciones a la hora de aplicar esos patrones. Dentro de las arquitecturas más comunes tenemos: Cliente Servidor, Arquitectura de tres niveles, Modelo Vista Controlador, Orientada a Servicios.

Arquitectura Cliente Servidor

(Tahuiton, 2011). La arquitectura cliente servidor está compuesto por 2 niveles, cliente y servidor, y cada nivel puede tener un número diferente de capas, las aplicaciones basadas en la arquitectura cliente servidor, siempre serán aplicaciones con 2 niveles como mínimo, aunque no está limitada a solo 2 niveles.

Figura 1: Arquitectura Cliente Servidor



Fuente: Elaboración Propia. Arquitectura Cliente Servidor

(Marini, 2012). En la arquitectura cliente/servidor. Los clientes y servidores son objetos separados desde un punto de vista lógico y que se comunican a través de una red de comunicaciones para realizar una o varias actividades de forma conjunta. Un cliente hace una petición de un servicio y recibe la respuesta a dicha petición. Un servidor recibe y procesa la petición, y devuelve la respuesta solicitada.

La arquitectura Cliente/Servidor: trata de una arquitectura de tres niveles que introduce un agente en el lado del servidor. El agente se convierte en un intermediario para las interacciones entre el cliente y el servidor. De esta manera, el servidor puede comunicar con el agente servidor incluso si el dispositivo móvil es inalcanzable en ese momento, y el agente del servidor comunicarse adecuadamente con el cliente cuando se recupera la conectividad inalámbrica (Molina, Corchado, & Bajo, 2008).

Arquitectura Modelo Vista Controlador

El patrón Modelo Vista y Controlador (MVC) es el más extendido para el desarrollo de aplicaciones donde se deben manejar interfaces de usuarios, éste se centra en la separación de los datos o modelo, y la vista, mientras que el controlador es el encargado de relacionar a estos dos. Las ventajas de usar un patrón MVC son: a) Permitir la distribución de las interfaces de usuario. B) Generar componentes de las interfaces. C) Diseñar vistas simultaneas del mismo modelo. D) Aplicar fácilmente cambios de las interfaces. Considerando el acoplamiento como el grado de interdependencia entre las unidades de software (módulos, funciones, subrutinas) de un sistema informático. En este sentido en el patrón MVC, el acceso a datos depende directamente del mismo modelo que se mapea por medio de la consulta SQL (vista), por tanto, los DTO y DAO corresponden a una estructura muy acoplada a la vista, ya que los objetos de intercambio (DTO) dependen directamente de los DAO y la generación de los mismos dependen directamente del modelo (Camarena, Trueba, Martínez, & López, 2012).

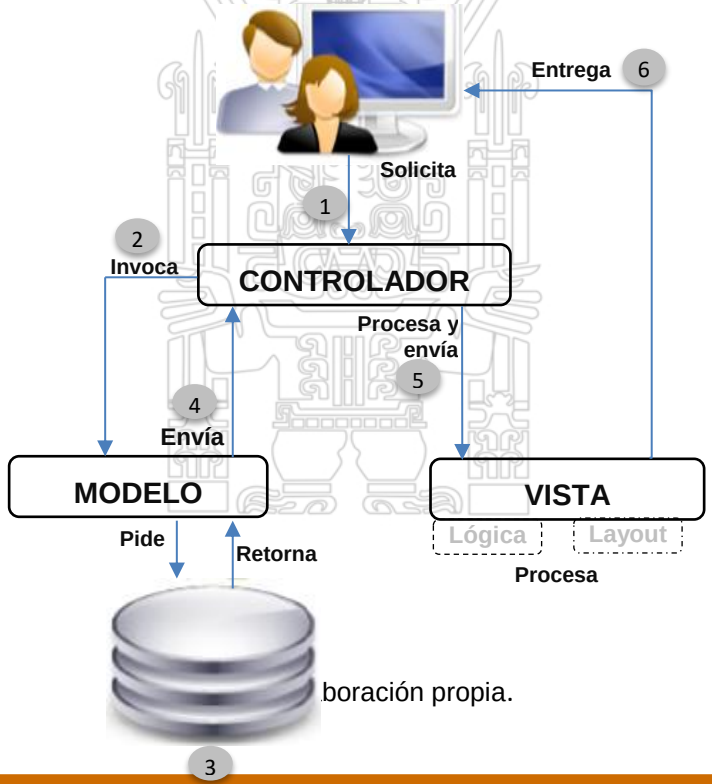
(Lacerda, Ribeiro, & Silveira, 2013). El Modelo Vista Controlador (MVC) es uno de los patrones más utilizados. El MVC comenzó como un framework desarrollado por Trygve Reenskaug para la plataforma Smalltalk en los finales de los años 70. Desde ese

entonces ha ejercido un papel muy importante en el desarrollo de los de más frameworks para interfaces de usuario.

Según (Gutiérrez, 2006), el patrón Modelo-Vista-Controlador es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información, el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema.

El modelo de arquitectura Model-View-Controller (MVC) divide una aplicación interactiva en tres componentes. El modelo contiene la funcionalidad principal y los datos. Las vistas muestran información al usuario. Los controladores manejan la entrada del usuario. Las vistas y los controladores juntos forman la interfaz de usuario. Un mecanismo de cambio-propagación asegura la coherencia entre la interfaz de usuario y el modelo(Red & Green, 2011).

Figura 2: Funcionamiento del patrón modelo-vista-controlador



boración propia.

Arquitectura orientada a Servicios – SOA

Arquitecturas Orientadas a Servicios (SOA) utilizan una arquitectura centrada en el concepto de servicio que se puede aplicar en el diseño de aplicaciones distribuidas (Wegdam, 2005).

(Gamarra, 2008). Estrictamente, SOA significa Arquitectura Orientada a Servicios, y se aplica a la arquitectura de un sistema de información que se puede describir como compuesta principalmente de servicios. Esto es, las funciones que dicho sistema realiza se distribuyen a lo largo de los diferentes servicios de los que se compone. (Georgakopoulos & Papazoglou, 2009). Dentro del mundo de la arquitectura SOA, los servicios son los bloques de construcción y en el nivel más bajo de la pila. Sin embargo, la mayoría de las empresas están tan centrados en la arquitectura de los servicios subyacentes, y ponen muy poca atención en la base de la arquitectura, que es el nivel más esencial.

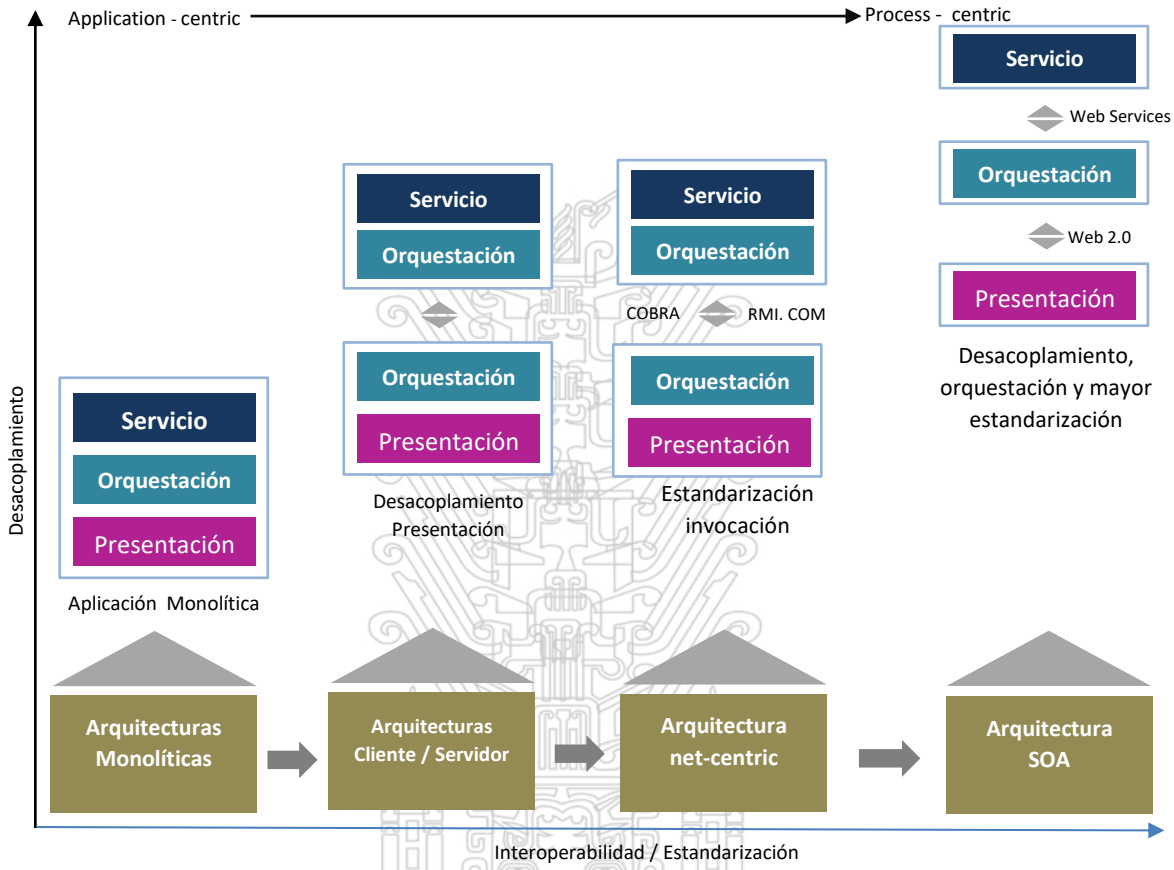
(Gonzales, 2011). En su estudio de arquitecturas de redes orientadas a servicios, describe desde el punto de vista tecnológico como es el resultado de la constante evolución hacia un mayor desacoplamiento de las capas de una aplicación (presentación, orquestación de procesos y servicios de negocio) y a un mayor nivel de estandarización/interoperabilidad de cada una de estas capas.

Aumenta el grado de reutilización al desacoplar las capas de una aplicación, permite reutilizar las aplicaciones existentes mediante la encapsulación en servicios, permite la utilización de servicios de terceros, permite reaprovechar las plataformas existentes, simplifica la adaptación de los sistemas existentes, evita el desarrollo de interfaces punto a punto entre los sistemas, aumenta la interoperabilidad entre sistemas, permitiendo tanto la externalización como la prestación de servicios, aumenta el nivel de automatización de los procesos, reduciendo el número de actividades manuales, permite monitorizar la actividad del negocio, permite realizar un análisis estadístico de los flujos de negocio reales en base a indicadores clave de negocio, permitiendo la identificación de puntos de mejora a optimizar.

Asimismo, permite evaluar el impacto y beneficio de variantes en los procesos mediante simulación. Favorece la industrialización, mejora la especificación de los requerimientos de negocio, proporciona una filosofía de desarrollo común a todos los negocios y canales, mejora la calidad, desacopla el desarrollo de servicios y de procesos, mejora el mantenimiento, permite presentar al usuario la información dispersa en distintos sistemas y de forma integrada, permite alcanzar un mayor nivel de automatismo en las

aplicaciones en procesos complejos de workflow, permite utilizar tecnologías de presentación avanzadas como Web 2.0.

Figura 3: Impacto SOA en la evaluación de las TI

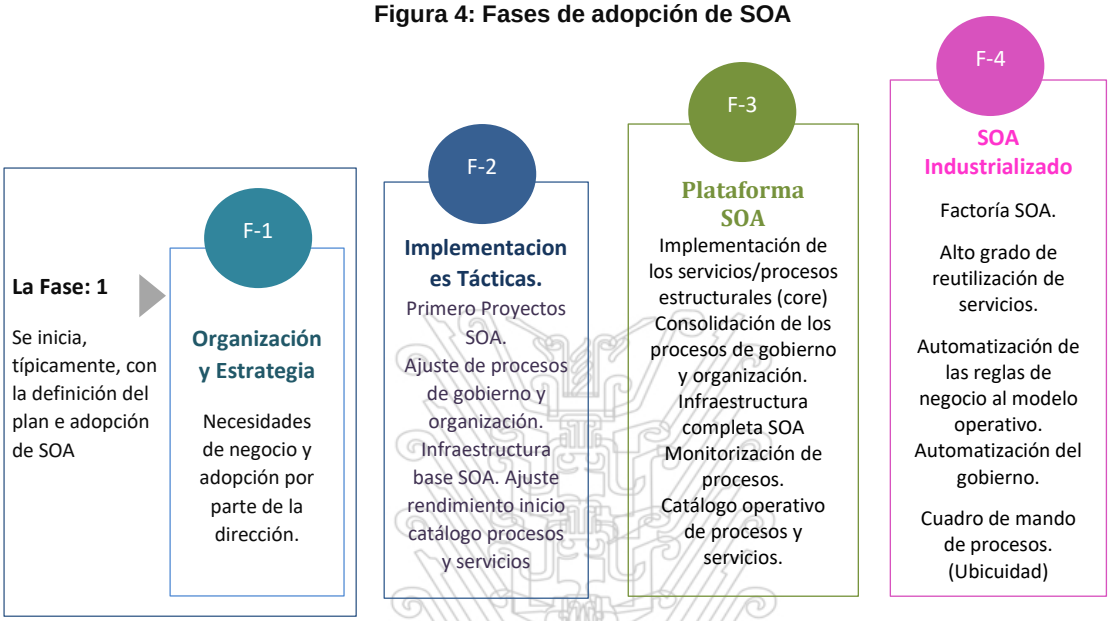


Fuente: González (2011). Estudio de Arquitecturas de Redes Orientadas a Servicio

(Fernández(Accenture), 2011). Propone una estrategia de adopción de SOA. Al insistir en que la tecnología de la información sea parte de una arquitectura corporativa más amplia, es evidente que la arquitectura SOA cuenta con un enorme potencial para maximizar el valor de la tecnología como habilitadora de ventajas competitivas. Los estudios realizados por Accenture demuestran que esta capacidad de utilizar la tecnología para impulsar la agilidad y la innovación en el negocio, constituye un elemento fundamental para el alto rendimiento y tener éxito.

Como siempre, la cuestión principal es encontrar la forma de alcanzar las ventajas deseadas. La experiencia acumulada por Accenture, trabajando con clientes de todo el mundo en busca del alto rendimiento, indica que es necesario abordar la implantación de SOA en cuatro fases, realizando varias iteraciones en cada una de ellas.

Figura 4: Fases de adopción de SOA



Fuente: Fernández Accenture (2011)

Fase 1. Organización y estrategia

Esta es la fase de toma de contacto con SOA, donde la compañía se centrará en la evaluación de la situación actual y en el plan para definir el alcance de la transformación hacia SOA, asegurando una base sólida de servicios y una hoja de ruta para obtener todos los beneficios de SOA. Tradicionalmente, esta fase se compone de cuatro tareas secuenciales: Comprensión de la estrategia de negocio y procesos, Análisis de la situación actual de los sistemas, definición del modelo objetivo de referencia SOA y creación de la hoja de ruta SOA.

Adicionalmente, en esta fase (o en la siguiente) se pueden realizar algunos pilotos con los proveedores de infraestructura y software.

Fase 2. Implantaciones tácticas

En esta fase se realizarán las primeras implantaciones tácticas de SOA, con el objetivo de que sirva también para familiarizarse tanto con la tecnología usada como con los procedimientos de gobierno y organización. Además, durante la fase se creará la infraestructura base de SOA y se iniciará el catálogo de procesos y servicios. Es recomendable que en la fase 2 se elijan las aplicaciones con un alto componente de workflow para obtener el máximo beneficio de la tecnología SOA y permitir probar dicha tecnología en su máxima extensión. También en esta fase se suele iniciar el proceso de

identificación y reutilización de los servicios existentes, así como su publicación en el catálogo.

Fase 3. Plataforma SOA

En la fase 3 se consolidará la implantación de SOA, tanto desde el punto de vista tecnológico como desde el punto de vista organizativo y de gobierno. En esta fase, además de consolidar la infraestructura base de SOA, se profundizará en la monitorización de procesos y se dispondrá de un catálogo operativo de procesos y servicios. Desde el punto de vista de negocio se realizará la implantación de los servicios y procesos estructurales.

Fase 4. SOA industrializado

Durante la última fase se obtendrán todos los beneficios de la filosofía SOA. Se alcanzará un alto grado de reutilización de servicios y se impondrá el modelo de factoría SOA, donde la organización se centrará en diseñar los procesos, y tanto la construcción de los mismos como los servicios requeridos (que no existan en el catálogo) se externalizarán en factorías.

Por otra parte, se alcanzará un mayor grado de sofisticación en la gestión de SOA, como en la automatización de las reglas de negocio al modelo operativo, en la automatización del gobierno, en la implantación de un cuadro de mando de procesos y en la federación (ubicuidad) de los servicios.

En esta tesis, se adopta este estilo de arquitectura para el diseño de aplicaciones sensibles al contexto. En este sentido, podemos decir que una aplicación sensible al contexto proporciona un servicio sensible al contexto a sus usuarios.

Computación Ubicua

(Weiser, 1991a). La computación ubicua es un mecanismo por el cual se facilita el uso de computadores, haciendo que en nuestro entorno existan múltiples sistemas de computación, pero siendo éstos «imperceptibles» para el usuario. De esta forma, el entorno se transforma en un entorno «inteligente», capaz de responder nuestros requerimientos. Así, el usuario puede concentrarse en la tarea que desea realizar y no en la interacción con los sistemas informáticos.

(Sloman et al., 2006). El objetivo de la computación ubicua no es apoyar la ubicuidad global, para interconectar todos los sistemas para formar un dominio de servicio omnipresente, sino más bien para apoyar la ubicuidad basada en el contexto.

(Brumitt, Meyers, Krumm, Kern, & Shafer, 2000). «La computación ubicua es el resultado de la ecuación computación móvil + entorno inteligente». Según los autores, en los próximos cinco a diez años veremos algunas soluciones operando en salas de reuniones o en hogares, así como en aplicaciones de enfermería doméstica.

(Poslad, 2009). La autonomía de los sistemas Ubicuos, se refiere a la propiedad de un sistema que permite controlar sus propias acciones de forma independiente. Un sistema autónomo puede todavía ser conectado con otro sistema y el medio ambiente. Los sistemas ubicuos se definen como sistemas que son autónomos y son capaces de tomar sus propias decisiones y acciones independientes. Hay varios tipos, y diferentes sistemas ubicuos. En Internet, un sistema ubicuo es un sistema que se rige por una política enrolada para una o más redes, controlada por un administrador de red común a favor de una sola entidad administrativa. Un agente de software se caracteriza a menudo como un sistema ubicuo autónomo.

Los sistemas ubicuos pueden ser diseñados de manera que, estos objetivos puedan ser asignados de forma dinámica entre ellos, tal vez por los usuarios. Por lo tanto, en lugar de los usuarios pueda interactuar y controlar cada interacción de tareas de bajo nivel, los usuarios sólo tienen que interactuar para especificar las tareas de alto nivel necesarias y programar automáticamente el conjunto de tareas de bajo nivel necesarios y el calendario de forma automática, lo que reduce la complejidad para el usuario. El sistema también puede planificar de nuevo en caso de que un plan o cronograma no logre los objetivos. Los problemas de planificación a menudo se resuelven utilizando inteligencia artificial.

El propósito principal de la computación ubicua, es la reducción de la interacción humana con la tecnología. Gran parte de la interacción del sistema omnipresente no puede ser totalmente centrado en el hombre, considerando que la interacción humana puede convertirse rápidamente en un cuello de botella para operar un sistema complejo. Los sistemas pueden ser diseñados para apoyarse en los seres humanos que están en el bucle de control. El cuello de botella puede suceder en cada paso, si el usuario está obligado a validar o entender cada una de las tareas.

(Yang, Syukur, & Loke, 2013). La computación móvil y ubicua es una parte esencial de la informática de hoy y será así en los años venideros. El visionario Mark Weiser, da a conocer al mundo alrededor de 1990 la computación ubicua, y desde ese entonces se ha apoderado de la imaginación de muchos. La tendencia es que la informática se volverá más móvil y más omnipresente en el futuro. Quizás en 30 años a partir de ahora, la

computación será sinónimo de computación ubicua, es decir, la palabra "omnipresente" ya no necesitará una mención especial.

La computación móvil y ubicua, hace referencia a una convergencia de dispositivos de computación avanzada como sensores, software conectado a través de tecnología inalámbrica. El internet ambiental como medio abierto para una vía de investigación conocido como "móvil y la computación ubicua". Un sistema informático que reconoce la información de ubicación, se considera un contexto esencial para ofrecer información correcta para el usuario. Sin embargo, la información más precisa puede ser entregada mediante la identificación de la semántica de una ubicación.

(Molina et al., 2008). Los entornos inteligentes, también conocidos como inteligencia ambiental, se han convertido en creciente cada vez más importante en los últimos años. Estos ambientes se caracterizan por ciertas capacidades como la ubicuidad, la transparencia y la inteligencia.

Los sistemas multi-agente (MAS) se han vuelto cada vez más relevante para el desarrollo de entornos distribuidos, dinámicos e inteligentes. Una de las ventajas de estos agentes es su adaptabilidad para trabajar en dispositivos móviles, por lo que el soporte inalámbrico, la comunicación (Wi-Fi, Bluetooth, WiMAX, UMTS, etc.) lo que facilita su portabilidad a una amplia gama de dispositivos móviles. Esta ventaja hace que los agentes y sistemas multi-agente sean muy apropiadas para ser aplicadas al desarrollo de entornos ubicuos y móviles.

Los agentes pueden ser caracterizados a través de sus capacidades en áreas tales como la autonomía, reactividad, pro actividad, habilidades sociales, el razonamiento, el aprendizaje y la movilidad.

Estas capacidades hacen que los sistemas multi-agente sean muy apropiados para la construcción de entornos inteligentes. Un agente puede actuar como una interfaz entre el usuario y el resto de los elementos del entorno inteligente. Además, dada la adaptabilidad de los agentes a los dispositivos móviles, es posible proporcionar una interacción ubicua y transparente, incluso la personalización de acceso al usuario. Un agente inteligente puede adaptarse a los cambios ambientales o hacer predicciones basadas en el conocimiento o experiencia previa. En este sentido, un agente es sensible al contexto y puede tomar decisiones que le permite adaptarse automáticamente a los cambios en su entorno. Un agente generalmente se integra dentro de un multi-sistema de agente o agente de la sociedad, el intercambio de información y la resolución de problemas en forma distribuida.

De acuerdo a (Biegel & Cahill, 2004) La conciencia del contexto y la movilidad son conceptos centrales en la visión de la computación ubicua donde las redes de pequeños dispositivos informáticos se dispersan en el entorno físico, operando de forma autónoma e independiente del control centralizado. Las aplicaciones conscientes al contexto son un subconjunto importante del conjunto de aplicaciones informáticas omnipresentes, y ya han demostrado las ventajas obtenidas de la capacidad de percibir el entorno. Estas aplicaciones, sin embargo, siguen siendo difíciles de desarrollar y desplegar, sin un modelo de programación ampliamente aceptado disponible. A menudo se requiere que los programadores escriban grandes cantidades de código e interactúen con sensores y actuadores para desarrollar aplicaciones relativamente simples. La movilidad de los dispositivos en el entorno informático omnipresente también plantea retos en las áreas de comunicación e interacción debido a factores tales como direcciones de red dinámicamente cambiantes y configuraciones del sistema, susceptibilidad a la desconexión y bajo ancho de banda.

Los componentes principales de una aplicación contextual son un conjunto de sensores para la captura de datos de contexto, un conjunto de reglas que gobiernan el comportamiento de acuerdo con el contexto y un conjunto de actuadores para generar respuestas. Hemos desarrollado el modelo de objeto sensible para el desarrollo de aplicaciones sensibles al contexto en un entorno móvil ad-hoc, que define abstracciones de software para sensores y actuadores, y proporciona un marco para la especificación del comportamiento impulsado por reglas de producción. Los objetos sensibles tienen una serie de características que son importantes en entornos de computación omnipresentes. Sentencia (Capacidad para percibir el estado del medio ambiente mediante sensores), Autonomía (Capacidad de operar independientemente de la interacción humana) y pro actividad (Capacidad para actuar en previsión de la objetivos o problemas.)

Contexto

Las aplicaciones sensibles al contexto adaptan automáticamente su comportamiento y configuración, dependiendo de las condiciones del entorno y de las preferencias del usuario(Loayza, Proaño, & Camacho, 2013).

El contexto es un concepto que aparece en diversas disciplinas (Bradley & Dunlop, 2005). Puesto de relieve la importancia de la comprensión y el estudio del contexto, organismos, objetos y eventos son partes integrales del medio ambiente y no pueden entenderse de manera aislada de ese entorno (Fetzer, 2007). En la psicología (Steven, 2013) y (Tom, 2007), los investigadores están interesados en cómo los cambios en el

contexto deben controlar diversos procesos cognitivos, tales como la percepción, razonamiento, toma de decisiones, y aprendizaje.

En informática la noción de contexto ha sido mencionada por primera vez en la inteligencia artificial (McCarthy, 1993). En este campo, el contexto aparece como un medio de dividir una base de conocimientos en conjuntos manejables o construcciones lógicas que facilitan las actividades de razonamiento. Más recientemente, con los avances en la informática móvil, el contexto se ha convertido en un tema de interés a otras áreas de la informática, como la telemática y la computación ubicua. En estas áreas, el contexto es usualmente considerado como el conjunto de condiciones ambientales que pueden ser utilizados para adaptar las aplicaciones móviles a los usuarios y a las necesidades actuales. Desde el contexto del usuario que cambia con frecuencia, las aplicaciones que se ejecutan en los usuarios de dispositivos móviles pueden adaptar su comportamiento en función de estos cambios.

En este ámbito por ejemplo (Kotz & Chen, 2000), definen contexto desde la perspectiva de la aplicación. Ellos consideran contexto como el conjunto de estados ambientales y los entornos que determinan un comportamiento en las aplicaciones o en los que se produce un evento que es interesante para el usuario. Por el contrario, (Abowd et al., 1999) define el contexto como el estado físico, social, emocional, o informativo del usuario, centrándose así en el usuario, en lugar de centrarse en la aplicación. Del mismo modo, (Bradley & Dunlop, 2002) describe contexto como el conocimiento acerca de los objetivos, las tareas, las intenciones, la historia y preferencias del usuario que una aplicación de software aplica para optimizar la eficacia de la aplicación. La definición más mencionada del contexto dado por (Anind, Abowd, & Salber, 2001)equilibra el enfoque en: usuarios y aplicaciones: contexto es cualquier información que se puede utilizar para caracterizar la situación de entidades (es decir, si una persona, lugar, u objeto) que se consideran relevantes para la interacción entre un usuario y una aplicación.

(Du & Wang, 2008) apuntan a la programación automática de aplicaciones, y para este efecto proponen un framework específicamente orientado a la generación de aplicaciones sensibles al contexto. Este entorno permite usar un conjunto de especificaciones con las cuales se puede definir la conducta de las aplicaciones mediante la generación de reglas que pueden atarse a un conjunto de acciones dependientes del contexto. A partir de estas especificaciones el modelo del framework genera automáticamente la aplicación.

(Khalil & Connelly, 2005) estudian las reacciones de los usuarios de dispositivos móviles frente a dos posibilidades: configuración puramente automática para adaptarse al contexto, o configuración manual. El estudio arroja como resultado que la configuración

automática es preferida sobre la manual, sin embargo, también sugiere que un enfoque híbrido, automático-manual, es preferido sobre aquel puramente automático, lo cual podría explicarse por el gusto que el usuario experimenta al manipular su teléfono.

Según (Floréen, Lindén, Niklander, & Raatikainen Floréen, 2005), define el contexto como estado emocional del usuario, foco de atención, ubicación y orientación, fecha y hora, objetos y personas en el entorno del usuario.

Computación Móvil

De acuerdo a (Kalle & Yoo, 2002) la computación móvil tiene que ver fundamentalmente con incrementar nuestra capacidad de mover físicamente los servicios computacionales junto con nosotros. Como resultado la computadora se convierte en un dispositivo que siempre está presente (porque nos acompaña), el cual expande nuestras capacidades para recordar, comunicarnos y razonar, independientemente de la ubicación física del dispositivo. Esto puede ocurrir ya sea por la reducción de tamaño de las computadoras y/o por la provisión de acceso a la capacidad computacional a través de una red, mediante dispositivos menos poderosos.

Esta evolución tiene que ver con la migración de los dispositivos desde los escritorios hasta nuestros bolsos (notebooks), bolsillos (handhelds y smart-phones) e inclusive el cuerpo (wereable computers). La parte interesante de esto es que entonces la computación se convierte en algo que literalmente nos puede acompañar a cualquier lugar, soportando una gran variedad de actividades humanas, Sin embargo, en Mobile computing existe una limitación importante: el modelo computacional no cambia considerablemente con la posición (ni el resto del contexto). Esto ocurre porque en general los dispositivos no aprovechan la información del contexto en el que están ejecutando, por lo tanto, no pueden adaptarse a éste. La información del contexto puede ingresar al sistema mediante el monitoreo de sensores; también es posible obtener esa información a través del ingreso manual por parte del usuario, pero este tipo de molestias deben ser minimizadas para evitar su distracción.

Según (Poslad, 2009), los teléfonos móviles se clasifican de la siguiente manera

Dispositivos que no están implantados. Pueden ser portátiles o de mano, que se lleva en la ropa o accesorios de moda. Portátil: tales como ordenadores portátiles que están orientados a la operación a dos manos mientras están sentados. Estos son generalmente los dispositivos más altos de recursos.

Dispositivos manuales: dispositivos funcionan normalmente con una sola mano y en ocasiones en manos libres, la combinación de múltiples aplicaciones como la comunicación, la grabación de audio-vídeo. Estos son dispositivos de bajos recursos.

Wearable: dispositivos tales como accesorios y joyas generalmente se operan con manos libres y funcionar de forma autónoma, por ejemplo, los relojes que actúan como gestores de información personal, auriculares que actúan como transmisores-receptores de audio, vidrios que actúan como transmisores-receptores visuales y lentes de contacto. Estos son dispositivos de bajos recursos.

Implantado o incrustados: estos son utilizados a menudo por razones médicas para aumentar las funciones humanas, por ejemplo, un marcapasos cardíaco. También pueden ser utilizados para mejorar las capacidades físicas y mentalmente. Los implantes pueden ser circuitos integrados macro o micro a base de silicio o pueden ser a base de carbono, por ejemplo, la nanotecnología.

Estática puede ser considerado como un antónimo de móvil. Los dispositivos estáticos tienden a ser movido antes de la instalación en una ubicación fija y luego residen allí durante todo su ciclo de vida completo. Ellos tienden a utilizar una conexión continua a la red (cableada o inalámbrica) y la fuente de energía fija. Pueden incorporar altos niveles de recursos locales de computación, por ejemplo, un ordenador personal, grabadores y reproductores, diversos aparatos domésticos y de oficina, etc.

Inteligencia Ambiental

La inteligencia ambiental propone una nueva manera de interactuar entre las personas y tecnología, donde la tecnología se adapta a las personas y su contexto en el que viven. Esta nueva forma tiene los siguientes objetivos:

Promover una visión de las personas rodeadas de interfaces inteligentes que se funden en la vida diaria.

Fomentar un ambiente con conocimientos informáticos de procesamiento inteligente mediante la creación de una interfaz hombre-máquina, natural y sin esfuerzo.

Desarrollar un conjunto de sistemas e interfaces inteligentes e intuitivas.

Desarrollar la capacidad de reconocer y responder a requerimientos de usuarios individuales de una manera integral.

Crear entornos tecnológicamente complejos en numerosos contextos, tales como la medicina, la academia, las estructuras sociales, etc.

Desde esta perspectiva, los agentes deben ser capaces de responder a los eventos, tomar la iniciativa de acuerdo con sus objetivos, comunicarse con otros agentes, interactuar con los usuarios, representar y gestionar la información de contexto y hacer uso del razonamiento y mecanismos para encontrar la mejor solución a los objetivos. Los nuevos enfoques para los sistemas basados en agentes de Inteligencia Ambiental proponen el uso del contexto para manejar un conjunto de tecnologías e Incorporar mecanismos de representación y administración de la información del contexto que proporcionan los agentes con la flexibilidad, y adaptación para sobrevivir en entornos dinámicos y llevar a cabo la visión de inteligencia ambiental.

El creciente uso de dispositivos inalámbricos en los últimos años ha dado lugar a nuevas necesidades, así como a una gran oportunidad para extender las técnicas tradicionales de comunicación inalámbrica (Punie, 2005).

Computación Pervasiva

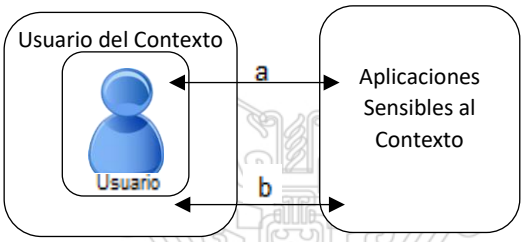
De acuerdo (Federico, 2006) El concepto pervasive computing implica que la computadora tiene la capacidad de obtener información del ambiente donde opera y utilizarla para construir modelos computacionales de manera dinámica. Este proceso es recíproco, pues el ambiente también debería ser capaz de detectar a aquellos dispositivos que están entrando en el. De esta manera ambos, dispositivo y ambiente, son conscientes uno del otro y pueden interactuar de manera inteligente para brindar una mejor calidad en los servicios. Esto es el núcleo de la idea de pervasive computing: Un área poblada con sensores, pads, badges y un conjunto de modelos del ambiente (físico, social, cognitivo, etc.) Los servicios de computación pervasiva pueden ser construidos ya sea embebiendo modelos de ambientes específicos en computadoras dedicadas, o bien construyendo capacidades genéricas en las computadoras para consultar, detectar, explorar y construir dinámicamente modelos de los ambientes.

Bases teóricas especializadas

Aplicaciones Sensibles al Contexto

La información del contexto es el conjunto de datos útiles que, en un instante concreto de tiempo, describe los elementos que rodean al usuario y algunos aspectos interesantes del propio usuario.

Figura 5: Aplicaciones sensibles al contexto

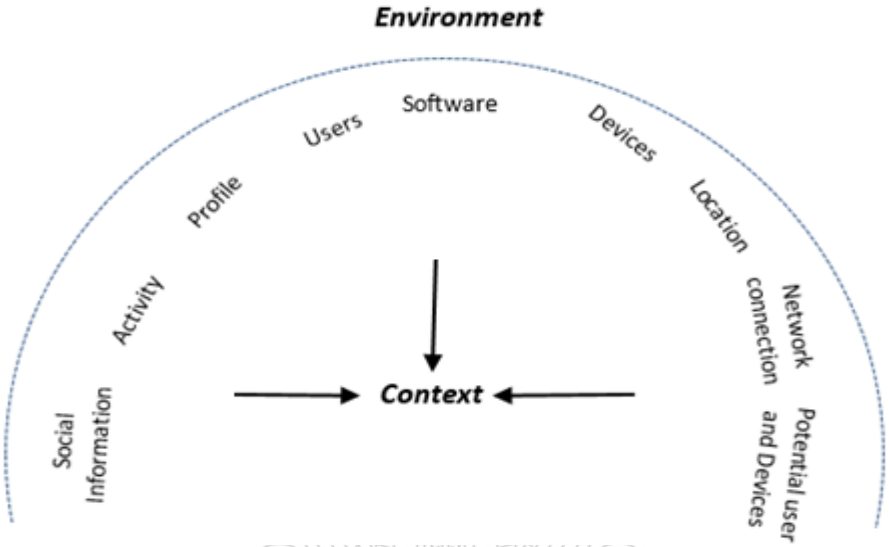


En la Figura 05, la flecha "a" muestra que el usuario y la aplicación sensible al contexto interactúan. Del mismo modo, la flecha "b" muestra que los usuarios del contexto interactúan con las aplicaciones sensibles al contexto.

Las interacciones representadas por la flecha "a" permitir, por ejemplo, la entrada de comandos y preferencias del usuario, y el uso del servicio proporcionado por la aplicación sensible al contexto. Las interacciones representadas por la flecha "b" permiten que las aplicaciones sensibles al contexto capturen determinadas condiciones del contexto de los usuarios (Ferreira, 1992).

La figura 06 representa todos los elementos típicos que podrían ser parte del contexto del usuario en un sistema de computación ubicua convencional (que incluye los agentes basado en software). Esta información tiene componentes estáticos y dinámicos. La información estática abarca todos los detalles relacionados con el usuario que no cambian a través del tiempo, o por lo menos no varían con frecuencia. El perfil de un usuario es un ejemplo de dicha información. En el perfil, la fecha de nacimiento no cambia, pero las preferencias en la música, por ejemplo, pueden cambiar a través del tiempo. La información dinámica cambia constantemente y proporciona una descripción instantánea y actualizada del usuario y su entorno. Por ejemplo, la ubicación del usuario en un edificio es un buen ejemplo de este tipo de información.

Figura 6: Contexto de un usuario en un sistema de computación ubicua convencional



Pero, ¿qué es en realidad un servicio sensible al contexto? se define un servicio sensible al contexto como el tipo de servicio que modifica su comportamiento de acuerdo a la información que obtiene, en relación con el contexto del usuario. Por lo tanto, un servicio de búsqueda de música que toma en cuenta implícitamente un usuario podría ser un servicio sensible al contexto. Otro ejemplo es: un servicio que busca restaurantes para un usuario, teniendo en cuenta su proximidad a las posibles opciones. Un concepto interesante relacionado con este tipo de servicio es la prestación de servicios sensibles al contexto. Se compone de los mecanismos necesarios para proporcionar a los usuarios de forma dinámica la información de los servicios. El conjunto de servicios ofrecidos dependerá de tiempo y de la información del contexto para su selección.

¿Por qué el contexto debe incorporarse en los sistemas de información de computación ubicua? La información del contexto es crucial para garantizar que los servicios se pueden personalizar en función al perfil del usuario, la ubicación del usuario, estado del usuario y el dispositivo del usuario. Estos elementos componen un conjunto de información contextual, cuya disponibilidad está en cualquier momento y lugar que se requiera (Chen, Finin, & Joshi, 2003). Ventajas típicas obtenidas mediante de la incorporación del contexto en un sistema son los siguientes:

El aumento de satisfacción del usuario: los servicios están más adaptados a sus preferencias y perfil.

Automatizar algunas funciones: por medio de reglas de comportamiento definidas por el usuario, algunas actividades pueden ser automatizados.

Información correcta en el momento adecuado y en el lugar adecuado: un modelo semántico de la información del contexto hace que sea posible filtrar información entrante, dependiendo de la situación del usuario.

Software autómata: ya que es capaz de decidir cuándo es más apropiado, y cómo, para interactuar con el usuario.

Uno de los retos de la telefonía móvil en computación distribuida es explotar los cambios del entorno con una nueva clase de aplicaciones que son conscientes del contexto en el que se ejecutan. Este tipo de software sensible al contexto se adapta de acuerdo con el lugar de utilización, el conjunto de personas cercanas, hosts y dispositivos accesibles, así como a los cambios a tales cosas en el tiempo. Un sistema con estas capacidades puede examinar el entorno informático y reaccionar a los cambios en el medio ambiente. Tres aspectos importantes del contexto son: donde estás, con quién estás y qué recursos están cerca. Contexto abarca más que la ubicación del usuario, debido a que otras cosas de interés también son móviles y cambiantes. El contexto incluye iluminación, nivel de ruido, conectividad de red, los costos de comunicación, el ancho de banda de comunicación, e incluso la situación social (Schilit & York, 1995).

(Loayza et al., 2013) Desarrollar aplicaciones sensibles al contexto requiere combinar activamente el conjunto de sensores del dispositivo con las preferencias del usuario respecto a sus posibles interacciones con dicho dispositivo y las actividades que se encuentra realizando, las cuales muy posiblemente estarán relacionadas a su localización. Hacer trabajar todos estos elementos de manera armoniosa requiere el uso de técnicas y metodologías específicas de software para facilitar el desarrollo de estas aplicaciones.

Herramientas para el desarrollo de aplicaciones contextuales en dispositivos móviles.

Existen distintas librerías y/o frameworks que facilitan el desarrollo de aplicaciones contextuales en dispositivos móviles. Podemos encontrar una Plataforma para el desarrollo de aplicaciones contextuales en dispositivos móviles (Raento, Oulasvirta, Petit, & Toivonen, 2005) . Este trabajo se desarrolló en 2005 y está enfocado a teléfonos de la plataforma Symbian y los dispositivos Nokia Serie60, lo cual supone una limitación para su uso. La motivación de ContextPhone es similar a la de este trabajo: investigar y desarrollar un patrón de arquitectura que permita el futuro desarrollo de aplicaciones contextuales de manera más sencilla y con más posibilidades explotando todas las características de los dispositivos móviles. En concreto en este trabajo se desarrollan

distintos módulos para controlar el dispositivo y obtener información contextual, podemos encontrar cuatro módulos:

- *Sensors*: El módulo de sensores obtiene información como la localización por GPS o identificación por células, así como los registros de llamadas, mensajes SMS/MMS, nivel de batería o actividad del usuario. Este módulo es quizás el más enfocado a la búsqueda de información contextual.
- *Communications*: Módulo que permite la comunicación exterior a través de distintos medios, como GPRS, SMS, MMS o Bluetooth. Nótese que este módulo no se centra en la obtención de información contextual, pero para el desarrollo de aplicaciones contextuales ofrece la posibilidad de conexión a internet a través de GPRS o de comunicación a través de medios como mensajes cortos, multimedia o bluetooth.
- *Customizable applications*: El módulo de personalización de aplicaciones permite sustituir o aumentar las características de las aplicaciones integradas en el dispositivo.
- *System services:* Módulo de servicios del sistema que permite a las aplicaciones acceder a la recuperación de errores y el log, además de poder lanzar servicios en segundo plano.

En resumen, ContextPhone permite la obtención de información contextual desde un teléfono móvil para el desarrollo de aplicaciones contextuales. Dentro de esta información contextual se puede encontrar la geolocalización por GPS o Cell ID, los contactos del usuario, nivel de batería, actividad del usuario en el dispositivo y acceso a contenidos como mensajes cortos y multimedia, así como al registro de llamadas. Esta plataforma permite el desarrollo de aplicaciones contextuales pudiendo utilizar estas características de manera más sencilla y rápida sin la necesidad de una implementación nativa en el dispositivo. El principal inconveniente es que su uso está restringido a dispositivos de la plataforma Symbian y Nokia Series60, lo cual en la fecha de desarrollo del trabajo pudo ser una decisión acertada ya que por aquel entonces Symbian y Nokia copaban el mercado de dispositivos móviles. Sin embargo, en la actualidad y en pocos años (debido a la aparición de los Smartphones) este mercado ha sido dominado por las plataformas Android e iOS.

En a la dualidad Android e iOS tienen el monopolio de los smartphones han surgido iniciativas para estandarizar el desarrollo de aplicaciones. (Wargo, 2015), en su libro presenta una plataforma de desarrollo de aplicaciones móviles mediante tecnologías web (HTML). De esta forma se posibilita desarrollar una misma aplicación para distintas plataformas sin la necesidad de hacer grandes cambios al cambiar de una a otra.

PhoneGap se creó en 2008 por Nitobi Software con la intención de profundizar en el uso de lenguajes web (JavaScript) para el desarrollo móvil. Su propuesta se fundamentó en dos principales creencias, la primera es que el formato web puede dar solución a la portabilidad entre plataformas, la segunda, que toda tecnología acaba volviéndose obsoleta. La primera creencia se basa en que los lenguajes web HTML, CSS y JavaScript, aún con sus limitaciones y debilidades han conseguido llegar a una masa crítica, gracias a que cualquier persona puede publicar cualquier cosa desde cualquier lugar. La segunda creencia, la obsolescencia de la tecnología, es algo que no necesita demostración, es algo observable a lo largo de la historia, aunque haya tecnologías que perduran décadas lo que hoy es tendencia mañana puede caer en el olvido.

De esta manera resulta conveniente no cerrar los desarrollos de software para móviles a determinadas plataformas. Esto es lo que pretende PhoneGap con su propuesta de utilizar el formato web para el desarrollo de aplicaciones. Para ello ofrecen un framework gratuito y de código libre que permite hacer de puente entre las aplicaciones web y los dispositivos móviles. PhoneGap contribuye en la Apache Software Foundation (AFS) bajo el nombre de Apache Córdova.

Tabla 3: Frameworks para el desarrollo de aplicaciones sensibles al contexto

Framework	Marco	Descripción
GDK	Comercial	Solamente válido para el desarrollo para hardware Google Glass
Android Wear	Comercial	SDK público, valido para el desarrollo sobre hardware corriendo sobre android Wer.
Apple iBeacon	Comercial	Define un standard basado en BLE cerrado para dispositivos iOS
Qualcom Gimbal	Comercial	Basado en iBeacon, requiere de la integración de los Beacons Gimbal para proveer información de contexto.
Estimote	Comercial	Basado en iBeacon, requiere de la integración de los Estimote para proveer información de contexto.
Smart Things	Comercial	Limitado al contexto provisto por la domótica.
Hermes	Académico	Propuesta de un framework para el desarrollo de aplicaciones mobiles context-aware, pero sin aplicación real.
Socam	Académico	Propuesta de un middleware orientado a servicios para el desarrollo de aplicaciones context-aware

Patrones - Arquitecturas y Manipulación del Contexto

En esta sección se presenta tres patrones para apoyar el desarrollo de la Arquitectura de software para aplicaciones contextuales en dispositivos móviles. Estos patrones ayudan a solucionar los problemas recurrentes asociados a la gestión de la información del contexto y reaccionar de forma proactiva a los cambios del contexto. Además, se presenta los componentes genéricos que integran la plataforma. También los patrones como medios para documentar soluciones de diseño para los problemas recurrentes conocidos. Se profundiza en los actores involucrados el desarrollo de la arquitectura

Patrones Sensibles al Contexto

Los patrones de arquitectura de software han sido propuestos como formas de solucionar problemas recurrentes en situaciones de diseños específicos. Estos documentan experiencias de diseño existentes y probadas, permitiendo la reutilización de los conocimientos adquiridos por los profesionales con experiencia. Por ejemplo, un patrón de arquitectura de software describe un problema de diseño recurrente y presenta un esquema genérico para su solución. El esquema contiene componentes, responsabilidades y relaciones entre sí.

Los modelos de arquitecturas de software también presentan otras propiedades deseables, tales como:

- Los Patrones proporcionan un vocabulario común y la comprensión de los principios de diseño.
- Son un medio para documentar arquitecturas de software.
- Apoyan la construcción de software con propiedades definidas.
- Apoyan la construcción de arquitecturas de software complejo y heterogéneo
- Ayudan a la gestión de la complejidad del software.

Presentamos tres patrones de arquitectura que se pueden aplicar en el desarrollo de plataformas sensibles al contexto, es decir, el patrón de Event-Control-Action, el patrón de jerarquía de las fuentes de contexto y el patrón de acciones. Estos patrones solucionan los problemas recurrentes asociados con la gestión de la información de contexto, y reaccionan de forma proactiva a los cambios de contexto.

Patrón Event-Control–Actión (ECA)

El patrón Event-Control-Action proporciona una estructura de alto nivel para las aplicaciones sensibles al contexto que reaccionan de forma proactiva a los cambios de contexto. Un modelo de aplicación define el comportamiento de la aplicación, que puede

describirse por medio de, por ejemplo, las reglas de condición. En este modelo, los problemas de gestión de información del contexto, tales como la detección y el procesamiento del mismo, están desconectados de las cuestiones relativas a reaccionar a los cambios del contexto (Costa, 2007).

Por ejemplo, supongamos que nuestra plataforma debe proporcionar soporte para aplicaciones en el ámbito médico. Un ejemplo de ello sería una aplicación de monitoreo que monitoriza a los pacientes epilépticos y proporciona momentos de asistencia médica antes y durante una crisis epiléptica. La aplicación debe medir la variabilidad de la frecuencia cardíaca y la actividad física, y a la vez predecir futuros ataques de forma automática. Además, el paciente debe ser informado con anticipación, ser capaz de detener las actividades en curso, tales como conducir un coche, etc. El objetivo de la utilización de esta aplicación es proporcionar al paciente con los niveles más altos de seguridad e independencia que le permite funcionar más normalmente en la sociedad a pesar de su trastorno.

En este escenario la aplicación lleva un sistema de monitoreo del corazón del paciente que recoge las señales del corazón a lo largo del día. Estas señales son procesadas por algoritmos inteligentes que son capaces de detectar anomalías en cuestión de segundos, tales como la probabilidad de tener un ataque epiléptico. En otro escenario, el paciente tendría una enfermera capaz de proporcionar los primeros auxilios ante un posible ataque epiléptico. Sin embargo, en este caso de la aplicación es quien se encarga de enviar las señales derivadas del sistema de supervisión a los médicos en tiempo real, y los médicos deciden si debe ser llevado o no al hospital más cercano.

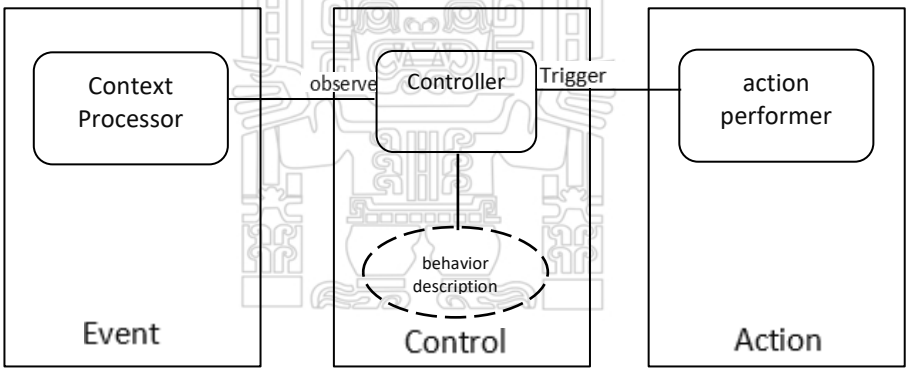
En el ejemplo presentado anteriormente impone una serie de requisitos a la plataforma.

- La plataforma debe ofrecer soporte para la obtención de información del contexto, tales como la frecuencia cardíaca y la presión arterial del paciente con el fin de predecir posibles ataques epilépticos.
- Ubicaciones de los médicos, enfermeras y del paciente deben ser conocidos por la aplicación.
- Conectividad debe ser permanente con el paciente.
- Dispositivos (por ejemplo, teléfonos móviles) de las enfermeras y médicos tienen que tener la aplicación con la alarma instalada en caso de convulsiones.
- Conexiones de transmisión en tiempo real necesitan estar establecidos con los teléfonos móviles de los médicos.
- En el caso de una situación crítica, una ambulancia tiene que llevar al paciente al hospital más cercano.

La solución al problema anterior, según el patrón arquitectónico evento-control-acción tiene como objetivo proporcionar un esquema estructural para permitir la coordinación, configuración y la cooperación de la funcionalidad distribuida dentro de la plataforma. Divide las tareas de recolección y procesamiento de la información del contexto, ejecuta las tareas de activación en respuesta a los cambios del contexto y el control de comportamiento de la aplicación.

Dada la naturaleza reactiva de las aplicaciones sensibles al contexto, se pueden describir en términos de reglas, por ejemplo: If <*condición*> then <*acciones*>. La parte condicional especifica la situación en la cual se habilitan las acciones, las condiciones están representadas por combinaciones lógicas de eventos. Los eventos se modelan y se observan por uno o más componentes del procesador del contexto. En caso que la condición se convierta en verdad, un componente de acción más destacado desencadena las acciones especificadas en las normas de condición. Las acciones son operaciones que afectan el comportamiento de la aplicación en respuesta a la situación definida en la parte condicional de la regla. Una acción puede ser una simple llamada de servicios web o una entrega de SMS, o puede ser una composición compleja de los servicios. El esquema arquitectónico propuesto por el patrón de la ECA consta de tres componentes: procesador de contexto, controlador y los componentes de acción. La figura 8: muestra un diagrama de componentes del patrón de ECA, y se aplica a plataformas de servicios sensibles a contexto (Costa, 2007).

Figura 7: Diagrama de componentes del patrón ECA



La información del contexto es gestionada por el componente del procesador de contexto, que genera y observa los sucesos. Este componente depende de la modelización de la información del contexto. El componente del controlador está inicialmente provisto de una especificación del comportamiento de la aplicación, que describe un fragmento particular de la lógica de la aplicación. Esta especificación determina el comportamiento de la

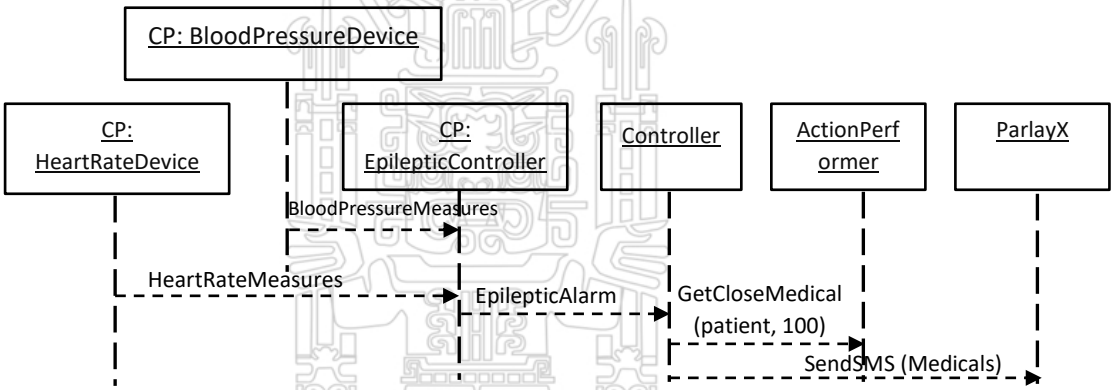
información, y la ejecución de las acciones pertinentes en respuesta a los cambios observados en el contexto. Dada la naturaleza reactiva de aplicaciones sensibles al contexto, se sugiere un enfoque basado en reglas de comportamiento de las aplicaciones sensibles al contexto. La descomposición y la implementación de enlaces, son abordadas por el componente de acción. El patrón ECA sugiere que una aplicación sensible al contexto debe ser lógicamente estructurado en el más alto nivel de abstracción(Costa, 2007).

Considerando el ejemplo presentado anteriormente en el que se detecta un posible ataque epiléptico y se ponen en contacto con los médicos del paciente a través de SMS. Aquí se supone que, cuando se detecta un posible ataque epiléptico, los médicos más cercanos se ponen en contacto a través de SMS. La figura 08 ilustra el flujo de información entre los componentes del patrón de Evento-control-acción. La regla de la condición es (aquí se aplica a un paciente llamado Carlos) definido dentro del controlador tiene la forma:

Figura 8: Dinamismo del Patrón Evento-Control-Acción

If <event: EpilepticAlarm>

Then <SendSMS (carlos, closeby (Medical, 100))>



El controlador observa la ocurrencia del evento EpilepticAlarm. Este evento es capturado por el controlador, que es una instancia del procesador de contexto. Las medidas de la presión arterial y la frecuencia cardíaca se obtienen de otras instancias ofrecidas por el procesador de contexto. Sobre la base de estas medidas y un complejo algoritmo, el componente controlador es capaz de predecir en cuestión de segundos que un ataque epiléptico está a punto de suceder, y el evento EpilepticAlarm se genera como consecuencia.

Ante la ocurrencia del evento EpilepticAlarm, el controlador activa la acción especificada en la regla. Él envió de mensajes “SendSMS” (más cercanos (médicos, 100)) es una acción compuesta que puede ser resuelto y ejecutado parcialmente por la plataforma. La acción interior (médicos, 100) puede ser ejecutado por completo dentro de la plataforma. La ejecución de esta acción requiere otro ciclo de recopilación de información del contexto, a fin de proporcionar la ubicación actual del paciente y sus médicos, y calcular la proximidad de estas personas. Al invocar la operación getCloseMedical (paciente, 100) el controlador es capaz de obtener los médicos que se encuentran dentro de un radio de 100 metros del paciente. Por último, invoca al controlador de forma remota para proporcionar un proveedor de negocio (por ejemplo, un proveedor de Parlay X) para enviar mensajes de alarma a los médicos (Costa, 2007).

Aplicando el principio de diseño clásico de la separación por capas, el patrón event-control-acción, ha permitido efectivamente la distribución de responsabilidades en plataformas sensibles al contexto. Permitiendo que los componentes del procesador de contexto encapsulen las acciones relacionadas con el contexto, lo que les permite ser implementadas y mantenidas por diferentes partes del negocio. Las acciones se desacoplan del contexto, permitiendo ser desarrollado y operado ya sea dentro o fuera de la plataforma. La aplicación de tales principios de diseño mejora la extensibilidad y flexibilidad de la plataforma, considerando que los procesadores de contexto y componentes de las actividades pueden ser desarrollados y desplegados bajo demanda. Además, la definición de comportamiento de la aplicación por medio de reglas de condición permite el despliegue dinámico de aplicaciones sensibles al contexto y permite la configuración de la plataforma en tiempo de ejecución (Costa, 2007).

Siempre que algunas circunstancias específicas cambian en el contexto del usuario, las aplicaciones deben ser capaces de ajustar su comportamiento. Para este propósito se puede utilizar el patrón Event-Control-Action-ECA. El patrón ECA es un patrón de arquitectura que puede facilitar el desarrollo de aplicaciones sensibles al contexto, ya que presenta soluciones para los problemas recurrentes asociados con la gestión de la información de contexto y reaccionar a los cambios del mismo. La figura 08 muestra que el patrón ECA, divide las tareas de recolección y procesamiento de información del contexto (*módulo de eventos*), a partir de las tareas se desencadena acciones en respuesta a los cambios de contexto (*Módulo de la acción*). Estas tareas separadas se realizan bajo el control de una descripción de comportamiento de la aplicación (*módulo de control*), en el que se describen comportamientos de las aplicaciones sensibles al contexto reactivos en términos de reglas ECA, también llamadas reglas de condición. Estas reglas tienen la forma if <*condición*> then <*acción*>.

La parte condicional de una regla de ECA especifica la situación en la cual se habilitan las acciones, la cual consiste en combinaciones lógicas de eventos. La parte de acción de la regla se compone de una o más acciones que se activan cada vez que la parte condicional se satisface (Costa, 2007).

Patrón de Fuentes de Contexto y Gestores de Jerarquía

El patrón de arquitectura de fuentes de contexto y gestores de jerarquía proporciona una estructura jerárquica de componentes del procesador de contexto. Este patrón se ha ideado con el fin de aplicar de forma recursiva las operaciones del procesamiento de la información del contexto en una jerarquía de componentes. En esta cadena de procesamiento de información del contexto, el resultado de una unidad de procesamiento se convierte de entrada para una unidad de nivel superior en la jerarquía hasta que se alcanza la información del contexto de nivel superior requerido.

Supongamos que extendemos el escenario del sistema presentado anteriormente, en el que se prevé un posible ataque epiléptico. Además de contactar a los médicos más cercanos, nos gustaría saber si el paciente está impulsando, con el fin de enviarle una alarma personalizada, con el siguiente mensaje; por favor traer el carro lo más pronto, es posible que tenga un ataque epiléptico. El procesamiento de la información del contexto es un reto. Deduciendo que a partir de los sensores básicos (por ejemplo, para mediar la frecuencia cardíaca, la presión arterial) puede requerir un cálculo complejo, en el cual puede haber varias fases de procesamiento de información necesarios antes de ceder semánticamente la información del contexto. Las actividades de procesamiento de información del contexto incluyen:

- Detectar: La recopilación de información del contexto a través de dispositivos y sensores. Por ejemplo, la recopilación de información de localización (latitud y longitud) a través de un dispositivo GPS.
- Agregación: observar, recoger y componer la información del contexto a partir de unidades de procesamiento de información. Por ejemplo, la recogida de información sobre la ubicación de varios dispositivos GPS.
- Inferir: interpretación de la información del contexto a fin de obtener otro tipo de información. La interpretación puede ser realizada sobre la base de, por ejemplo, las reglas de la lógica, las bases de conocimientos y técnicas basadas en modelos. La inferencia se produce, por ejemplo, al derivar información de proximidad de las localizaciones de los objetos de interés.

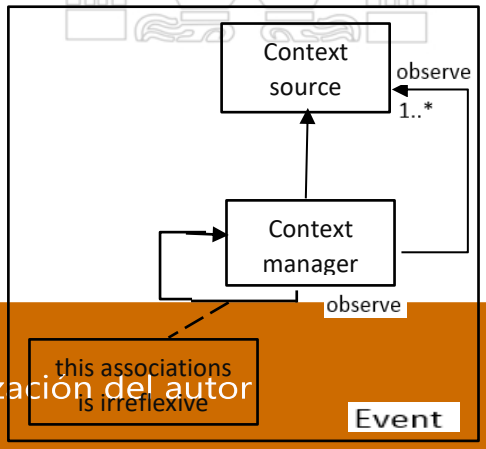
- Predicción: La proyección de información del contexto, produce información contextual con un cierto grado de incertidumbre. Podemos ser capaces de predecir el tiempo en la ubicación del usuario mediante la observación de los movimientos anteriores, la trayectoria, ubicación actual, velocidad y dirección de los movimientos.

La plataforma debe proporcionar mecanismos para distribuir las actividades de procesamiento de información del contexto entre múltiples componentes. Además, debe ser capaz de crear información del contexto a base de diversas fuentes de información de manera flexible (Costa, 2007).

La *solución* al problema anterior, el Patrón de Fuentes de Contexto y Gestores de Jerarquía tiene como objetivo proporcionar un esquema estructural para permitir la distribución y composición de los componentes de procesamiento de la información del contexto. Se definen dos tipos de componentes de procesamiento de información del contexto: *fuentes de contexto y gestor de contexto*. Los componentes de fuentes del contexto encapsulan sensores de dominio único, como el dispositivo de medición de la presión arterial o un GPS. *Los componentes de gestores del contexto cubren múltiples fuentes de contexto de dominio*, tales como la integración de la presión arterial y la medición de la frecuencia cardíaca. Ambos realizan actividades de procesamiento de la información del contexto.

La *estructura* propuesta por este patrón consiste en cadenas jerárquicas de fuentes de contexto y gestores, en el que, el resultado de una unidad de procesamiento de información del contexto puede llegar a ser de entrada para la unidad de nivel superior en la jerarquía. El resultado de la estructura es un gráfico a cíclico dirigido, en el que los vértices iniciales (nodos) del gráfico son siempre componentes de la fuente de contexto y el vértice final puede ser cualquiera de las fuentes de contexto o gestores de contexto. Los bordes dirigidos del gráfico representan el flujo de información del contexto entre los componentes (Costa, 2007).

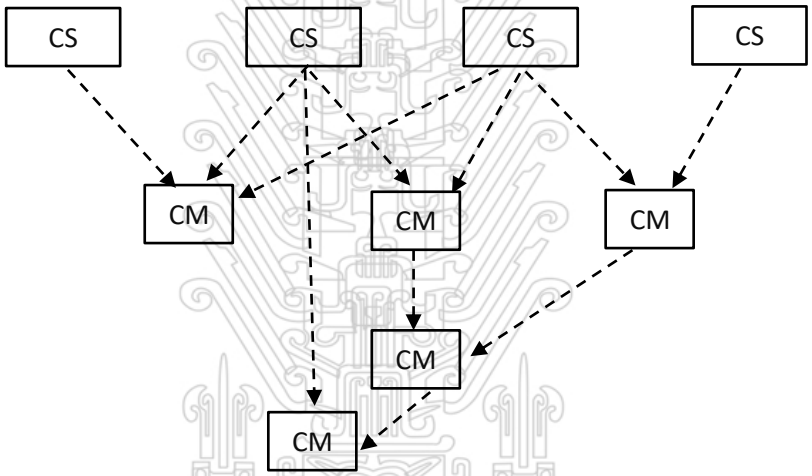
Figura 9: Patrón de Fuentes de Contexto y Gestores de Jerarquía



En el patrón, los gestores de contexto también desempeñan el papel de fuentes de contexto, ya que un gestor de contexto es una fuente de información de contexto derivada. Por lo tanto, los gestores de contexto heredan las características de las fuentes de contexto, e implementan funciones adicionales para manejar la información del contexto, recopilan de varias fuentes de contexto y gestores. Un gestor de contexto observa una o más fuentes de contexto y, posiblemente, otros gestores de contexto. La asociación entre la clase de gestor de contexto en sí es irreflexiva (Costa, 2007).

Figura 10: representa una estructura cíclica, que es una instancia del diagrama representado en la Figura 9. Las “CS” representan cajas de fuentes de contexto y “CM” representan instancias de gestores de contexto.

Figura 10: Instancia del Patrón de Fuentes de Contexto y Gestores de Jerarquía

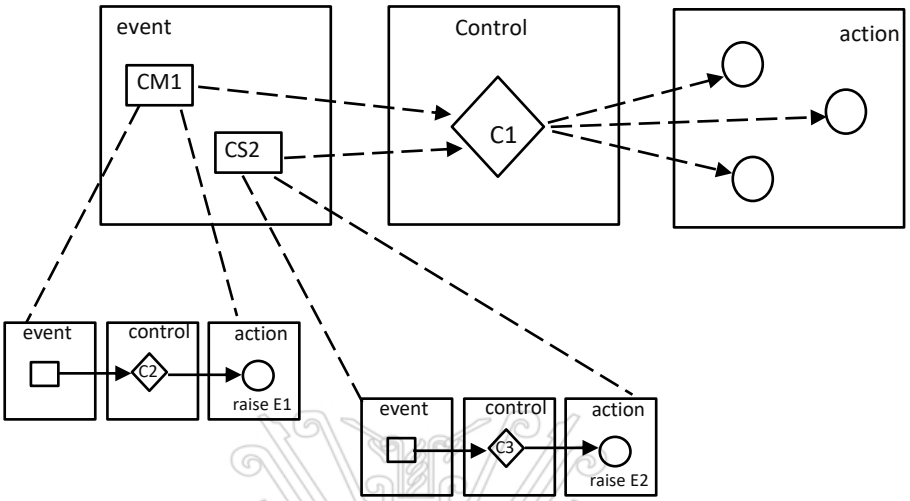


Dentro de una unidad de procesamiento de información de contexto (fuente de contexto o gestor), verificamos aplicaciones recursivas del patrón de evento-control-acción. Veamos en la figura 11: nos muestra la regla de estado del dispositivo manipulado por el controlador C1:

If <event: (EpilepticAlarm ^ driving)>

then <SendSMS (Por favor detenga el auto, es posible que tenga un ataque epiléptico)>

Figura 11: Aplicación recursiva del Patrón evento-control-acción



El evento (EpilepticAlarm ^ driving) es un evento compuesto observado en los siguientes componentes:

Un componente del gestor de contexto (CM1 figura 11) que detecta un evento de alarma epiléptico (E1). Un componente de fuentes de contexto (CS2 figura 11) que detecta cuando un paciente empieza un ataque epiléptico (E2). Dentro del gestor de contexto se detecta el evento epiléptico (CM1), la siguiente condición se describe en el controlador C2, la recursividad es la naturaleza del patrón de evento-control-acción:

If <event:(HeartRate > threshold)>
then <RaiseEvent (EpilepticAlarm)>

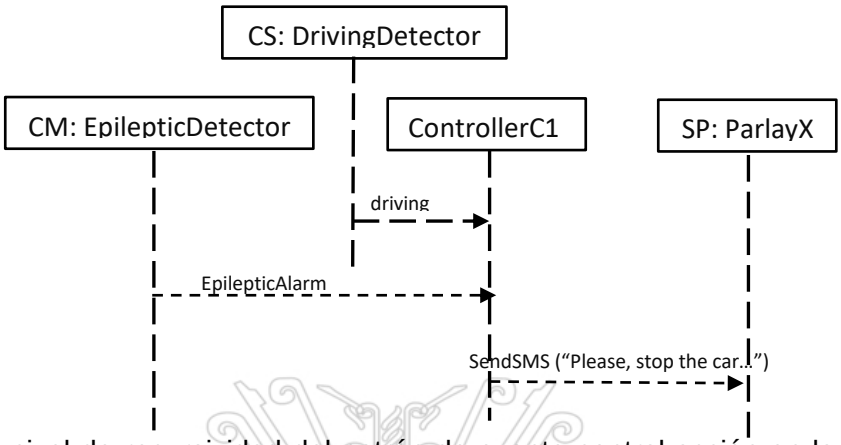
Controlador C2 observará las medidas de frecuencia cardíaca en un componente del contexto. La acción provoca un evento de alarma epiléptico. Dentro de la fuente de contexto se detecta la conducción (CS2), la regla siguiente describe en el controlador C3:

If <event:(userSignalOn)>
then <RaiseEvent (driving)>

El evento *userSignalOn* puede establecerse directamente al paciente o automáticamente detectar través de un dispositivo integrado en el auto que es capaz de detectar su presencia (Costa, 2007)

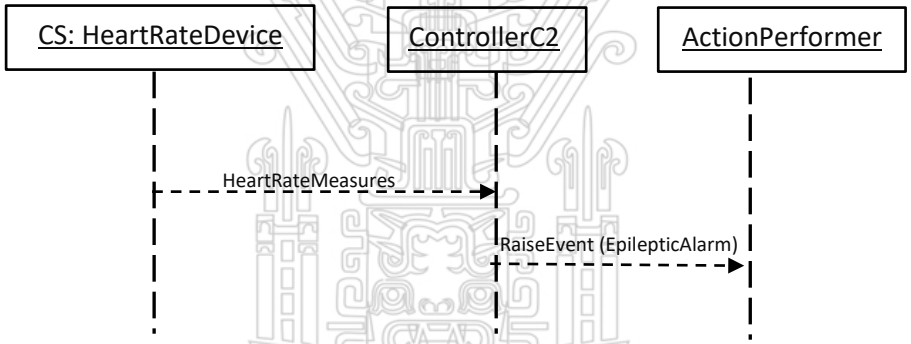
Consideremos de nuevo el ejemplo del ataque epiléptico se tocamos en las secciones anteriores. Figura 13 representa el flujo de información entre los componentes de la estructura de las fuentes de contexto y los gestores, en la parte superior la mayoría usa el patrón evento-control-acción. En este nivel, el Controlador C1 observará la ocurrencia del evento (EpilepticAlarm ^ driving), que se genera a partir de CM: EpilepticDetector y CS: DrivingDetector, respectivamente. Cuando la condición se vuelve verdadera (la alarma se ha puesto en marcha), el mensaje SMS personalizado se envía al paciente.

Figura 12: Dinámica del Patrón de fuentes de Contexto y gestores en el nivel más alto del patrón de recursividad evento-control-acción



En el segundo nivel de recursividad del patrón de evento-control-acción en la Figura 13, el controlador C2 observará las medidas de frecuencia cardíaca a partir de una fuente de contexto. Facultado con algoritmos capaces de detectar anormalidad del ritmo cardíaco, el controlador provoca el evento EpilepticAlarm cuando detecta la posibilidad de un ataque epiléptico (Costa, 2007).

Figura 13: Patrón de fuentes de Contexto y gestores en el segundo Nivel de recursividad



El Patrón de Fuentes de Contexto y Gestores de Jerarquía define una referencia estructural y jerárquica de componentes de código contextual. Este enfoque ha permitido la encapsulación y una distribución más eficaz, flexible y desacoplada de las actividades de procesamiento de información del contexto (detección, agregación, inferir y predecir). Este intento de mejorar la colaboración entre los propietarios de la información del contexto, se ha vuelto muy atractiva para nuevas las partes a unirse a esta red de colaboración, ya que la colaboración entre los socios permite la disponibilidad de información contextual.

Otro beneficio importante de la aplicación de este modelo es que permite el filtrado de información innecesaria a través de la jerarquía de las unidades de procesamiento de información del contexto. En el nivel más bajo de la recopilación de información contextual, una gran sobrecarga de flujo de información puede ser detectado, pero sólo la

información relevante para la lógica de la aplicación se mantiene y se envía al siguiente nivel de la jerarquía (Costa, 2007).

Patrón de la Arquitectura de acciones

El patrón de arquitectura de acciones proporciona una estructura de componentes para apoyar el diseño e implementación de la plataforma. Este patrón se ha diseñado con el fin de desacoplar los propósitos de las acciones, implementaciones y coordinar la composición de las mismas. El propósito de la acción define una intención abstracta, mientras que su aplicación representa la realización de este propósito con la utilización de las tecnologías específicas de implementación.

Algunas de las acciones que se presentan en el ejemplo se puede realizar de forma independiente y en paralelo, tales como: el envío de un mensaje de advertencia al paciente, el llamando a los parientes y notificar a los médicos más cénanos. Sin embargo, la acción para llamar a los enfermeros sólo se activa en caso de notificar a los médicos más cercanos, en el caso no se tenga éxito (no haya médicos disponibles). Esta situación define una dependencia entre las acciones. Además, algunas acciones pueden desencadenar una secuencia de otras acciones. Por ejemplo, para enviar la ayuda a los profesionales de la salud, puede ser necesario solicitar un expediente médico del paciente, para seleccionar la medicación pertinente, para comprobar la disponibilidad de transporte, y así sucesivamente. La plataforma debe proporcionar mecanismos para gestionar la coordinación de las acciones, especialmente cuando existen dependencias. Además, la plataforma debe apoyar el desacoplamiento de un objetivo de acción de sus implementaciones.

Aunque la acción "enviar profesionales de la salud" presenta un propósito común, sus implementaciones pueden variar, desde la logística puede diferir de un hospital a otro. La distribución y coordinación de las acciones se realiza de una manera flexible y desacoplada.

El patrón de arquitectura de acciones tiene como objetivo proporcionar un esquema estructural para permitir la coordinación de las acciones y la disociación entre las implementaciones. Se trata de: a) un componente de resolución de la acción que lleva a cabo la coordinación de las acciones dependientes, b) un componente proveedor de acción que define los propósitos de acción y un componente implementador de acciones que define las implementaciones.

El propósito de la acción describe una intención de realizar un cálculo sin ninguna indicación de cómo y por quién se implementan estos cálculos. Los ejemplos de los

propósitos de acción son "son llamadas" o "envió de mensaje". El componente acción define varias formas de implementar una determinada acción (Costa, 2007).

Por ejemplo, la acción "envió de mensaje " pueden tener diversas implementaciones, cada uno definido por un proveedor de telecomunicaciones específico. Por último, en el componente de resolución de acción se aplica técnicas para resolver las acciones, que se descomponen en unidades de los propósitos de acción que son indivisibles desde el punto de vista de la plataforma.

Figura 14 Se muestra un diagrama de clases del patrón de las acciones, ya que se puede aplicar para plataformas sensibles al contexto.

Por lo tanto, la resolución de la acción y componentes de proveedor de acción heredan las características del componente de acción. El componente de acción resuelve y realiza acciones compuestas. Los proveedores de actuación pueden ser los proveedores de servicios de comunicación, los proveedores de servicios de comunicación realizan servicios de comunicación, como una solicitud de red, mientras que los proveedores de servicios realizan servicios generales orientados a la aplicación, ejecutadas ya sea interno o externo a la plataforma, tales como la generación de alarmas o una simple entrega de mensajes, respectivamente.

Un proveedor de acción puede agregar varios componentes acción realizando implementaciones concretas con una determinada finalidad.

Figura 14: Estructura del Patrón Acción

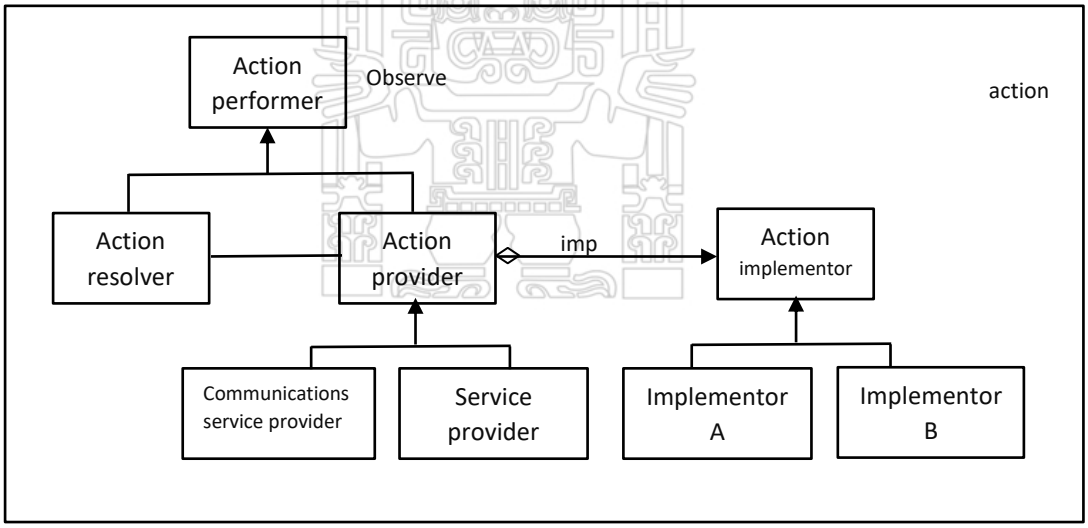
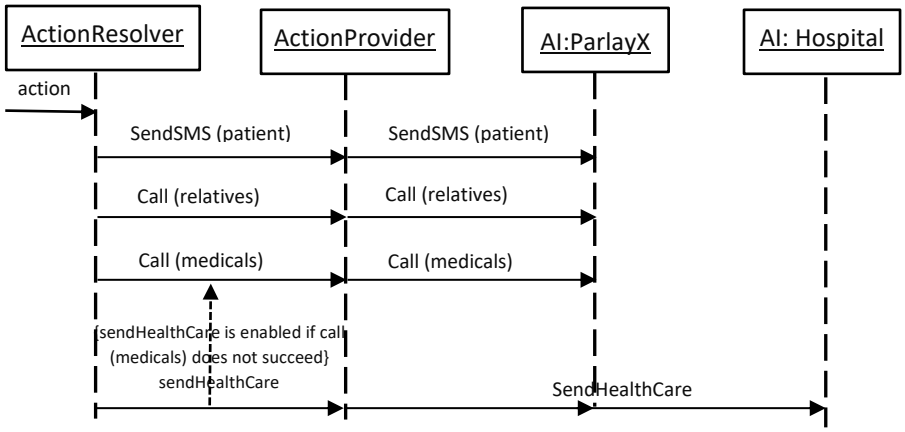


Figura 15: Ilustra el flujo de información entre los componentes del patrón de acciones para el escenario que se ha presentado anteriormente

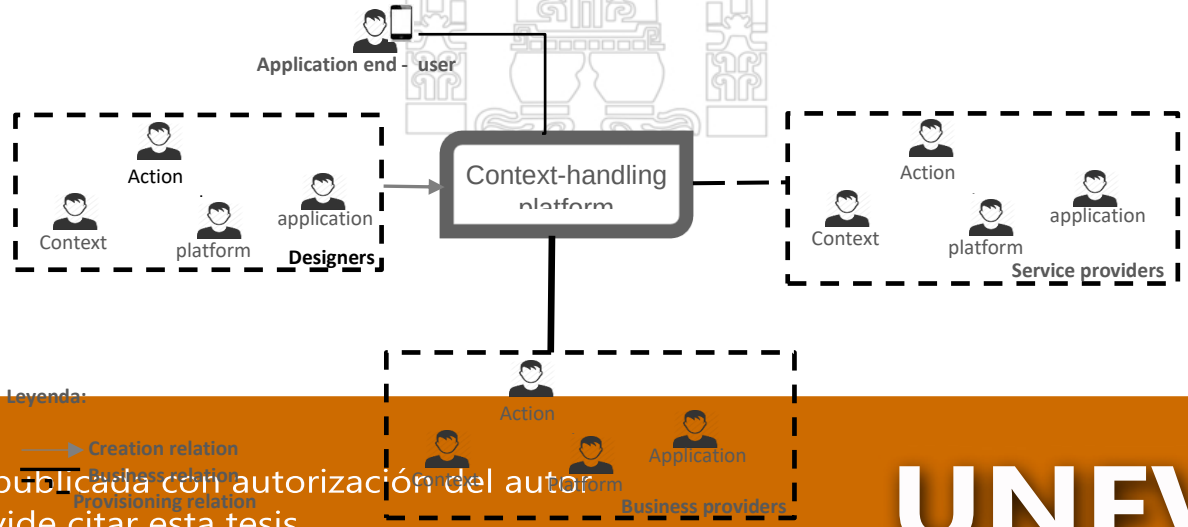


Mediante la definición de una estructura de resolución de acciones, el patrón ha permitido la coordinación de acciones compuestas y la separación del objeto abstracto y de la implementación. Este intento permite el enlace entre una acción y su implementación, lo que permite la selección de diferentes implementaciones en tiempo de ejecución de la plataforma. Además, los propósitos de acciones abstractas e implementaciones concretas de acción pueden ser cambiados y extenderse de manera independiente, con beneficios para la configuración dinámica y extensibilidad de la plataforma.

Los Stakeholders de la Plataforma

La arquitectura de la plataforma que gestiona el contexto proporciona medios para diversas partes de empresas, o grupos de interés, a fin de colaborar. Las plataformas de gestión de contexto solo se realizan con la colaboración de diversas partes interesadas, como los proveedores de ubicación, los proveedores de algoritmos de razonamiento de información de contexto, proveedores de acción, y así sucesivamente. La figura 16 representa los tipos de actores involucrados en el desarrollo de la plataforma, el servicio y el aprovisionamiento de negocios(Costa, 2007).

Figura 16: Stakeholders de la Plataforma



Se ha identificado tres distintas relaciones entre estos tipos de grupos de interés y la plataforma: la creación de acceso, lo comercial y el aprovisionamiento. La relación se refiere a la creación del proceso de desarrollo, que incluye el diseño y la implementación de componentes y servicios. La relación comercial se refiere a la gestión de la prestación de servicios y utilización del servicio desde una perspectiva empresarial. En el ámbito de los negocios, la prestación de servicios por lo general incluye la comercialización, los problemas financieros y legales, y el uso del servicio por lo general se refiere a los acuerdos comerciales entre el usuario del servicio y el proveedor, como el contrato de suscripción. La relación de aprovisionamiento se refiere a la realización de la prestación de servicios desde un punto de vista técnico, que incluye la configuración, instalación y mantenimiento de software y hardware.

En la figura 16, las relaciones están representados por líneas con flechas, las relaciones comerciales están representados por líneas más gruesas no con flechas, y las relaciones de aprovisionamiento están representados por líneas de trazos no con flechas. Para cada tipo de componente de la plataforma, hay partes interesadas: a) crear un componente (relación de creación), b) proporcionar el servicio ofrecido por ese componente desde la perspectiva de los asuntos de negocios (relación de negocios), y (c) proveer el servicio desde un punto de vista técnico (Aprovisionamiento de relación). Estos actores pueden coincidir, es decir, la misma organización pueden jugar simultáneamente las funciones de promotor, mantenedor o proveedor de negocio; Sin embargo, ya que están jugando diferentes papeles, los consideramos por separado (Costa, 2007).

Diseño de Plataforma y Proveedores Comerciales de Servicios

Hay tres tipos de actores que son responsables del desarrollo, mantenimiento y comercialización de la plataforma, a saber, el diseñador de la plataforma, proveedor de plataforma de negocios, y el proveedor de servicio, respectivamente. El diseñador de la plataforma es responsable de: creación de la plataforma, definición de las interfaces de los componentes, definición de los modelos de contexto y las directrices para ampliar estos modelos, desarrollo del componente controlador y el desarrollo de los componentes de acondicionamiento de contexto y de acciones internas. Además, el diseñador plataforma proporciona directrices a los desarrolladores de aplicaciones en componentes de cómo se pueden utilizar, configurar para satisfacer las necesidades particulares de la aplicación.

El proveedor de plataforma de negocios tiene como objetivo proporcionar los servicios de la plataforma desde un punto de vista comercial. Se trata de maximizar el

número de aplicaciones interesadas en utilizar la plataforma, ofreciendo oportunidades de negocio para los proveedores de aplicaciones. La plataforma enriquece sus servicios al permitir que los proveedores de acción y procesador de contexto se incorporen a la plataforma. De esta manera, una amplia variedad de servicios se ofrece a través de la plataforma, que es una invitación atractiva para un mayor número de usuarios finales y, por lo tanto, a las aplicaciones. Estas relaciones de negocios también son beneficiosas para el procesador de la acción y el proveedor de contexto, ya que la plataforma sirve como un puente para llegar a los usuarios finales.

El proveedor de servicios de plataforma tiene como objetivo la realización de los servicios que ofrece la plataforma desde un punto de vista técnico. El proveedor de servicios tiene como objetivo mantener una alta disponibilidad del controlador, procesador de contexto interno y componentes de la acción. Además, el proveedor de servicios es responsable de configuración e instalación de tales componentes(Costa, 2007).

Aplicación end-user

Las aplicaciones sensibles al contexto proporcionan valor a los usuarios finales en un sin número de maneras. Por ejemplo, los usuarios finales pueden mejorar su productividad en el trabajo o mejorar sus vidas personales mediante el uso de ciertos tipos de aplicaciones sensibles al contexto. Con el fin de utilizar los servicios de aplicaciones, los usuarios finales establecen relaciones de negocios con las aplicaciones, es decir, que se suscriban a aplicaciones en diversos campos, como las aplicaciones de turismo, salud, y de oficina.

No se espera que los usuarios finales de la aplicación interactúen directamente con la plataforma, la comunicación se produce a través de las aplicaciones sin la interacción humana. Sin embargo, dependiendo del modelo de negocio elegido, puede haber relaciones de negocios entre los usuarios finales y el proveedor de la plataforma de negocios, además de las relaciones comerciales entre los proveedores de aplicaciones empresariales y usuarios finales(Costa, 2007).

Diseño de la aplicación y proveedores de servicios

Los proveedores de diseño de aplicaciones y servicios de oficina son los actores responsables del desarrollo, comercialización y mantenimiento de las aplicaciones sensibles al contexto, respectivamente. Los diseñadores de aplicaciones tienen por objeto desarrollar aplicaciones sensibles al contexto desde una perspectiva de ingeniería. Con el fin de crear aplicaciones con el apoyo de la plataforma, los diseñadores de aplicaciones deben entender los servicios ofrecidos por la plataforma, ser capaces de

extender los modelos de contexto, ser capaz de combinar y configurar servicios de la plataforma para satisfacer las necesidades particulares de la aplicación.

Proveedores comerciales de aplicación tienen por objeto proporcionar servicios de aplicaciones desde un punto de vista comercial. Este grupo de interés prevé oportunidades de negocio al utilizar la plataforma, ya que las aplicaciones se enriquecen con una variedad de servicios de aprovisionamiento del contexto, que se ofrecen a través de la plataforma. Además, la plataforma proporciona la adaptación y la atención con el servicio de control. El proveedor de negocio de la aplicación también controla las relaciones comerciales hacia el proveedor de la plataforma de negocios, y los usuarios finales.

El proveedor de servicios de aplicaciones tiene por objeto la realización de los servicios ofrecidos por las aplicaciones. Esto puede incluir la instalación, configuración y mantenimiento de servidores de aplicaciones y el uso de mecanismos tales como servidores espejo para permitir alta disponibilidad de aplicaciones(Costa, 2007).

Diseño del procesador de contexto y proveedores comerciales y de servicios

Los diseñadores de procesador de contexto, comerciales y de servicios son los actores involucrados en el desarrollo, comercialización y mantenimiento de los componentes del procesador contexto. El diseñador de procesador de contexto desarrolla los componentes del procesador de contexto desde una perspectiva de ingeniería. Con el fin de construir componentes del procesador contexto que esté disponible a través de la plataforma, el diseñador de procesador de contexto debe entender y cumplir con la plataforma de aprovisionamiento de servicios y modelos de contexto.

Los modelos de contexto de plataforma deben ser utilizados por el diseñador del procesador de contexto como un modelo para guiar las actividades de procesamiento, como la interpretación o la agregación. El proveedor de negocio de procesadores de contexto tiene como objetivo proporcionar servicios de aprovisionamiento de contexto desde un punto de vista comercial. La prestación de servicios de aprovisionamiento de contexto a través de la plataforma es una oportunidad para llegar a más usuarios finales y para enriquecer los servicios del contexto mediante la colaboración con otros servicios de aprovisionamiento de contexto. La colaboración entre los componentes y procesadores de contexto permite la disponibilidad de información potencialmente más rica. El proveedor de procesadores contexto también controla las relaciones comerciales hacia el proveedor de la plataforma de negocios, y los usuarios finales, en caso de que exista tal relación.

El proveedor de servicios de procesador de contexto se hace cargo de los componentes del procesador contexto, mediante el mantenimiento de los artefactos de software y hardware necesarios para ofrecer este tipo de servicios de aprovisionamiento de contexto(Costa, 2007).

Diseñador de componentes y proveedores comerciales de servicios

Proveedores de Acción de diseño de componentes, comerciales y de servicios son los actores involucrados en el desarrollo, la comercialización y el mantenimiento de los componentes de la acción. El diseñador de componente de acción, desarrolla componentes de la acción desde una perspectiva de ingeniería. Con el fin de construir componentes de acción que esté disponible a través de la plataforma.

El proveedor de negocio de componente de acción tiene como objetivo proporcionar servicios de acción desde un punto de vista comercial. Proporcionar servicios de acción a través de la plataforma es una oportunidad para llegar a más usuarios finales y enriquecer los servicios de acción mediante la combinación de diversos servicios. El proveedor de negocio componente de acción también controla las relaciones comerciales hacia el proveedor de la plataforma de negocios, y los usuarios finales, en caso de que exista tal relación (Costa, 2007).

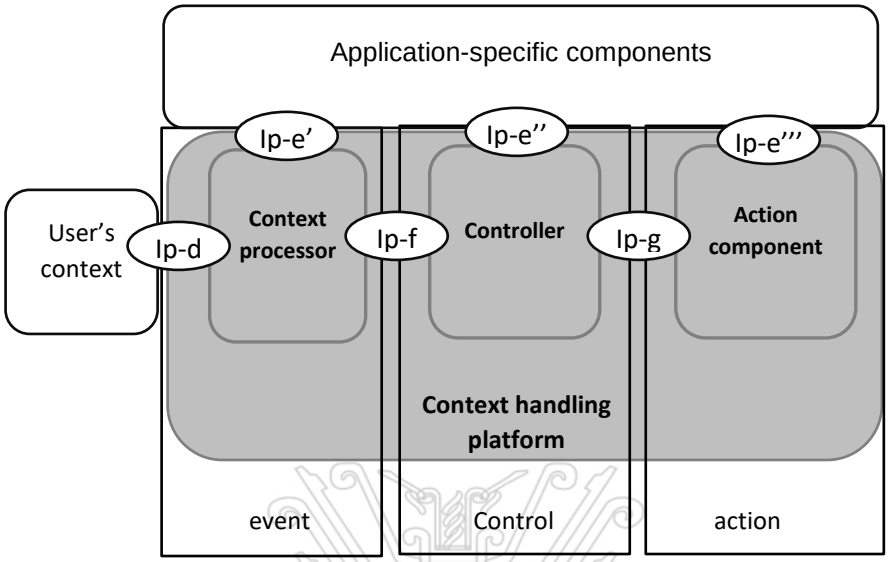
Controlador de servicios

El Controlador

El componente controlador [controller] tiene por objeto controlar la flexibilidad, extensibilidad y capacidad de adaptación de la plataforma de manejo de contexto. El componente controlador recibe las especificaciones de comportamiento de la aplicación como entrada, y ejecuta estos comportamientos en la plataforma, en nombre de las aplicaciones. Con el fin de ejecutar los comportamientos específicos de la aplicación, el componente controlador observa los eventos, reglas de condición, y desencadena acciones cuando se producen eventos particulares y las condiciones son satisfechas.

La Figura 17 representa el manejo de la plataforma de contexto, la plataforma se redefine en subcomponentes como el procesador de contexto, el controlador y los componentes de acción. El uso del patrón de ECA también está representado en esta figura por medio de rectángulos de trazos alrededor de los componentes. Esta cifra representa una configuración lógica de los componentes del procesador de contexto, controlador y la acción. Otras configuraciones también son posibles, en el que varias instancias de estos componentes interactúan entre sí.

Figura 17: Plataforma de Gestión de Contexto



La interacción ip-e ', ip-e'' e ip-e''' permiten a los componentes específicos de la aplicación interactuar directamente con los diferentes componentes de la plataforma. Los comportamientos específicos de la aplicación se delegan a la plataforma a través de los puntos de interacción de tipo ip-e''. El componente de control adopta estas conductas como entrada y configura el resto de la plataforma para operar correctamente de acuerdo a los requisitos específicos de la aplicación. Para ello, el controlador debe anunciar a los componentes del procesador del contexto que se necesitan ciertos tipos de información de contexto.

Los componentes del contexto como el procesador, es responsable de capturar el contexto del usuario a través de los puntos de interacción de tipo ip-d. Sobre la base de la información del usuario, los procesadores de contexto generan información del contexto y eventos, que se observan por los componentes del controlador a través de los puntos de interacción de tipo ip-f. Los componentes específicos de la aplicación pueden interactuar directamente con los componentes del procesador del contexto, a través de los puntos de interacción de tipo ip-e'. Esto permite que los componentes de la aplicación pueden acceder a la información del contexto, independientemente de la disponibilidad del componente controlador.

Cuando se cumple la combinación de condiciones del contexto definido por los comportamientos específicos de la aplicación, el componente controlador activa las acciones requeridas a través de los puntos de interacción de tipo ip-g. *Applicationspecific* componentes también pueden interactuar directamente con los componentes de la acción, a través de los puntos de interacción de tipo ip-e '''. Esto permite que las aplicaciones disparen las acciones, con independencia de la utilización del componente controlador (Costa, 2007).

Controlando el Servicio

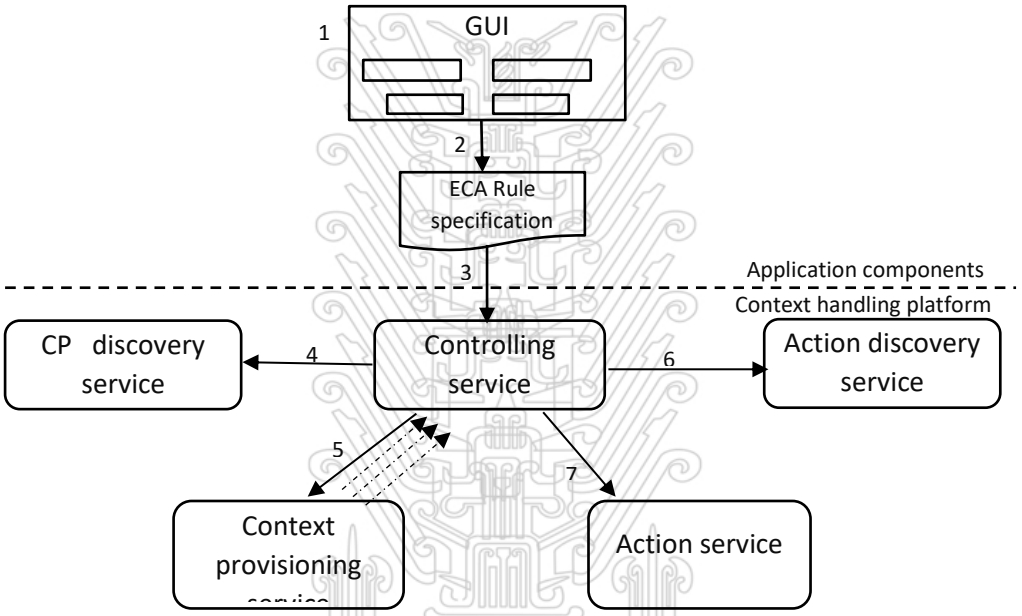
El componente controller ofrece servicio de control a los solicitantes. Este servicio permite a los usuarios llevar a cabo las reglas de Event-Condition-Action (ECA), y la consulta para casos específicos de información de contexto. El servicio es compatible con los siguientes tipos de control de operaciones: suscribirse, darse de baja, consultar y *notifyApplication*. Al suscribirse se activa una regla de ECA dentro de la plataforma, y al darse de baja se desactiva la regla ECA, cuando se consulta la información específica del contexto, la acción notifyApplication Notifica a la aplicación de la ocurrencia de eventos ECA.

La activación de la regla ECA se produce en tiempo de ejecución de la plataforma, lo que require el descubrimiento de los servicios de provision de contexto en tiempo de ejecucion. La Figura 18 representa un uso típico del servicio de control de flujo (Costa, 2007).

- *Fase 1:* La suscripción se inicia con un usuario final a algún servicio de la aplicación sensible al contexto por medio de una interfaz gráfica. Este tipo de servicio por lo general requiere que la aplicación debe comportarse de forma reactiva a los cambios del contexto y la situación.
- *Fase 2*: consiste en realizar el mapeo de una petición del usuario final a una especificación de la regla que debe proporcionarse a la plataforma, en un formato de secuencias de comandos (xml). La traducción de las peticiones del usuario para descartar especificaciones en alguna notación, es responsabilidad de los componentes de la aplicación. También es posible que los desarrolladores de aplicaciones especifican las reglas de aplicación en tiempo de diseño.
- *Fase 3*: consiste en la invocación real del servicio de control, en el que una aplicación suscribe reglas específicas. El servicio de control verifica si la especificación está bien formada y lo separa en eventos, condiciones y acciones.
- *Fase 4*: corresponde a la tentativa del servicio de control de buscar los servicios de aprovisionamiento del contexto capaces de proporcionar las notificaciones de eventos. El servicio de control decide si o no para suscribirse a uno o más de estos servicios de aprovisionamiento de contexto.
- *Fase 5:* consiste del en el Intercambio de mensajes de solicitud y suscribirse posibles notificaciones de eventos y solicitudes de consultas y respuestas. El servicio de control de determina si se cumplen las condiciones de la Información facilitada por los servicios de aprovisionamiento de contexto.

- *Fase 6:* comienza típicamente cuando una determinada condición se cumple. En este momento, una acción debe ser activado, y, por lo tanto, su aplicación real necesita ser encontrado. A tal efecto, el servicio de control utiliza el servicio de acción de descubrimiento.
- *Fase 7:* Los componentes del servicio de acción son invocados por el componente controlador.

Figura 18: Plataforma de Gestión de Contexto



Servicios de Localización

Los gestores y fuentes de contexto ofrecen servicios como en una arquitectura orientada a servicios y, por lo tanto, están registrados en un repositorio de servicios. Una fuente de contexto o un gestor de registros de servicios y ubicación, siempre debe estar disponible. Los consumidores de información de contexto (por ejemplo, los componentes del controlador y de la aplicación) hacen una solicitud de consulta para un servicio que tiene ciertas características, por ejemplo, el tipo de servicio, costos, ubicación y calidad de los parámetros de servicio. El registro de servicios comprueba la solicitud de consulta en el servicio que posea y responde al consumidor con la ubicación de la interfaz del servicio seleccionado. Los registros de servicio se comportan de forma proactiva mediante notificación a los consumidores de servicio exportado ofertas que mejor coinciden con la descripción de su consulta de servicio.

Tras la suscripción de la regla ECA, el componente controlador verifica si los eventos del contexto y la situación necesarios ya están a disposición del controlador. Si esta

información no está disponible, el controlador utiliza los servicios de localización para encontrar fuentes de contexto relevantes (para las notificaciones de información de contexto) y los gestores de contexto (para notificaciones de eventos situación), que son capaces de ofrecer esta información.

La solicitud de consulta puede devolver un conjunto de fuentes de contexto o gestores. Se asume que todos los proveedores de contexto en esta respuesta se comprometen a un cierto nivel de calidad. El proceso de suscripción con fuentes de contexto y gestores pueden utilizar una de las siguientes estrategias:

- *Suscribirse a todas las fuentes*: el controlador se adhiere a todas las fuentes de contexto o gestores emparejados en la solicitud de consulta. En este enfoque, las distintas notificaciones se reciben de la misma información. Para decidir cuál de ellos a tener en cuenta, el solicitante podrá utilizar las propiedades de calidad, tales como la actualización de eventos y precisión. Aunque este enfoque puede ser beneficioso para mejorar la fiabilidad, es potencialmente más caro debido al espejo suscripciones y tráfico de comunicación intensiva.
- *Suscribirse a una fuente*: el controlador selecciona una fuente contexto de la reserva de las fuentes dado lugar a la solicitud de consulta. El proceso de selección puede ser al azar o de los parámetros de calidad de uso.

Dado que las fuentes de contexto son vulnerables a fallos o errores, puede ser necesario realizar de nuevo una suscripción para el mismo evento con una fuente contexto diferente en caso de fallos (Costa, 2007). Las re-suscripciones pueden ser necesarios en las siguientes situaciones:

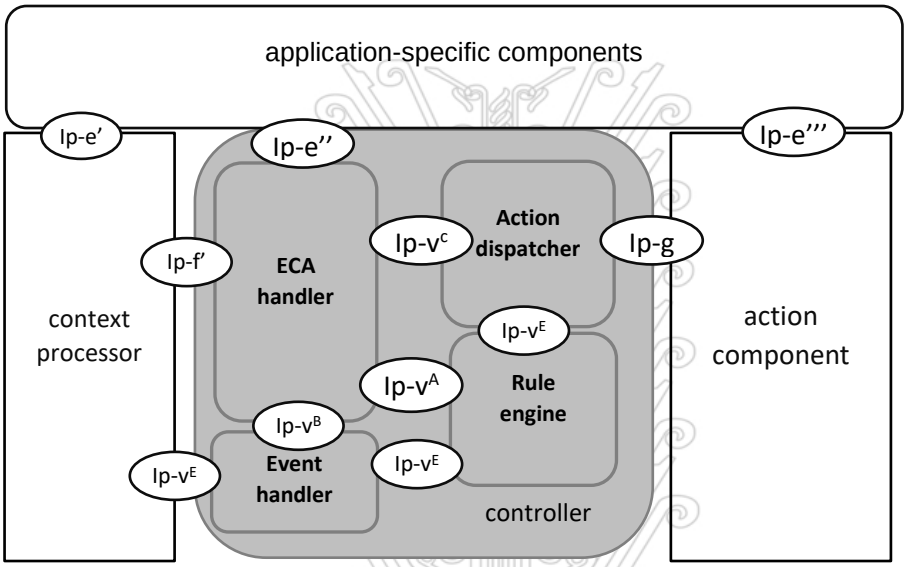
- Calidad del evento disminuye y ya no como se requiere en el contrato de servicios es.
- La fuente contexto o gerente es momentáneamente fuera de servicio debido a un fallo inesperado.
- La fuente contexto o gerente Se despega de la plataforma o queda sin conexión a propósito

Detalle del Diseño del Controlador

Como ya se ha mencionado en el apartado anterior, el componente controlador debe interpretar las normas ECA-DL, que se proporcionan como entrada al controlador. El procesador del contexto recopila información de diversos componentes del contexto, monitorea la ocurrencia de eventos y condiciones, e invocar acciones en respuesta a ocurrencias de eventos y evaluaciones. Para la realización de estas funciones, hemos refinado el componente controlador en subcomponentes: manejador de ECA, el

controlador de eventos, el motor de reglas y el constructor de acciones. Cada uno de estos subcomponentes tiene por objeto la realización de las funciones antes mencionadas, respectivamente. Figura 19: refina y se centra en el diseño interno del componente controlador.

Figura 19: Diseño Detallado del Controlador



El controlador ECA acepta las especificaciones de regla ECA a través de los puntos de interacción de tipo ip-e'' y comprueba si están bien formados con respecto a los modelos de contexto y situación compartida por los componentes de la plataforma (fase 3 en la figura 19). Tras el éxito de la validación de la especificación de la regla ECA, el controlador de ECA envía al controlador de contexto la lista de valores necesarios para evaluar la situación general, a través de los puntos de interacción de tipo ip-v^B. Una vez que la memoria de trabajo del motor de reglas puede ser adecuadamente llenada con eventos del contexto, la especificación de la regla ECA puede ser cargado en el componente de motor de reglas, a través de los puntos de interacción de tipo ip-v^A.

El controlador también verifica si una acción especificada en la regla ECA es (estado de emergencia) crítico. Si este es el caso, una solicitud para la obtención previa de la acción se envía al despachador de acciones a través de los puntos de interacción de tipo ip-v^C, evitando de esta manera, retrasar una acción de consultada.

Al recibir información del controlador ECA, el componente controlador de eventos verifica si los eventos del contexto y la situación ya están a disposición del controlador, y si no, se prepara una solicitud de consulta para cada evento de contexto que se necesita.

Esta consulta se realiza en un componente de descubrimiento de contexto, que no se representa en la Figura 19 en aras de la simplicidad. El componente de descubrimiento de contexto devuelve un conjunto de concurrencias de servicios que cumplan con los requisitos de la consulta.

El controlador de eventos elige una o más ofertas de servicios y se adhiere a ella a través de los puntos de interacción de tipo ip-f'', a fin de recibir las notificaciones de eventos. También realiza peticiones basados en consultas, en las que se espera una respuesta inmediata con valores del contexto y la situación. El controlador de eventos llena la memoria de trabajo por medio de puntos de interacción de tipo ip-v^{D}. Estos eventos se recolectan de los componentes del procesador de contexto distribuidos a través de los puntos de interacción de tipo ip-f''. El controlador de eventos realiza el filtrado de eventos y pre-procesamiento. Además, se puede detectar que un evento basado en el tiempo no ha sido notificado en el tiempo esperado, lo que provoca una suscripción adicional a ese evento.

El componente de motor de reglas es el núcleo de la arquitectura de controlador. Puede ser implementado en la parte superior de un motor de reglas de Jess, que es capaz de determinar cuando las condiciones son satisfechas por el conjunto actual de los datos disponibles de la memoria de trabajo. En respuesta a una condición adaptada, una acción se desencadena a través de los puntos de interacción de tipo ip-v$^{E.}$ (Costa, 2007)

La acción despachador comprueba si contiene o no un servicio de acción precargadas para la acción requerida. De lo contrario, de manera similar al controlador de la ECA, el despachador de acción realiza una solicitud de consulta con un componente de descubrimiento de acción para encontrar ofertas de servicios de acción adecuados. Por último, el despachador de acción invoca el servicio del componente de acción a través de los puntos de interacción de tipo ip-g.

ECA-DL

Dentro del campo de las aplicaciones informáticas sensibles al contexto podemos hacer uso de patrones arquitectónicos, que proporcionan una estructura general para este tipo de aplicaciones, y a la vez definir cómo debe comportarse. En el campo de la sensibilidad del contexto, uno de este patrón de arquitectura es el patrón de Event-Control-Action (ECA), que especifica cómo una aplicación sensible al contexto debe reaccionar a los cambios en su contexto (Maatjes, Pires, Costa, & Sinderen, 2007).

ECA-DL es un lenguaje de dominio específico dirigido a aplicaciones sensibles al contexto. *Las reglas en ECA-DL* consisten de parte de un Event que simula la ocurrencia de interés en contexto, una parte que especifica una condición que debe mantener antes

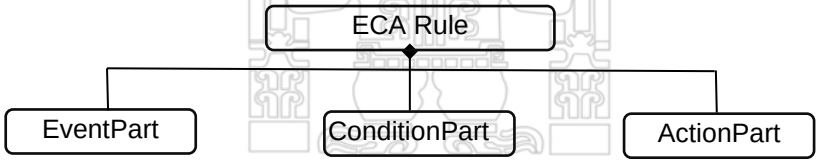
de la ejecución de la acción, y una parte de *Action* que se ejecuta cuando se cumplen las condiciones. A menudo la parte de acción de una regla consiste en la invocación de un servicio de notificación, pero también podría ser cualquier operación necesaria por la aplicación. Reglas ECA-DL siguen el patrón ECA, y, por lo tanto, pueden ser utilizados para especificar reglas de ECA. ECA-DL se ha desarrollado con los siguientes requisitos:

- *Poder Expresivo*: con el fin de permitir la especificación de las relaciones de eventos complejos. ECA-DL permite el uso de predicados operadores relacionales (por ej. <,>, =) y el uso de conectores lógicos (AND, OR, NOT) en eventos, lo que permite construir condiciones compuestas.
- *Uso conveniente*: con el fin de facilitar su utilización por parte de los desarrolladores de aplicaciones sensibles al contexto. ECA-DL proporciona construcciones de alto nivel que facilitan la definición de composiciones de eventos.
- *Extensibilidad*: con el fin de permitir la extensión de los predicados para dar cabida a eventos que se definen en la demanda, así como propiedades de evento.

En ECA-DL, los cambios de contexto se describen como cambios en los estados de situación. Las situaciones representan casos específicos de información de contexto, típicamente información de contexto de alto nivel. Las situaciones pueden ser definidas en otras situaciones o hechos. Lo datos definen el actual "estado de cosas" en el entorno del usuario.

Hechos y situaciones se definen como parte de los modelos de información, que hemos definido el uso de diagramas de clase UML. Nuestros modelos definen entidades, el contexto y las relaciones mutuas entre cada entidad y su contexto (Costa, 2007).

Figura 20: Elementos básicos de una Regla - ECA



Sintaxis de ECA-DL

Las reglas ECA-DL constan de tres partes: Evento, Condición y Acción. Cada una de estas partes está representada en la sintaxis en una línea separada por los llamados tokens (un "bloque primitivo de texto estructurado", por lo general una sola palabra) y sus argumentos. La sintaxis de las tres partes es:

- **Event**: Upon <event>
- **Condition**: When <condition>
- **Action**: Do <action>

Aquí cada uno tiene un argumento diferente

<Event>:

Este argumento contiene un evento, que consiste en una o más transiciones de estado. Hay tres estados posibles: verdadero, falso y desconocidos. Una transición se denota el uso de funciones como *EnterTrue (inOffice) o FalseToUnknown (LightOn),* es decir, que indica el cambio. Múltiples transiciones pueden ser atadas por los operadores lógicos como *AND, OR y NOT* para crear una transición compuesta. Sólo cuando toda la expresión compuesta ha evaluado a la verdadera, es decir, cuando se han sucedido las transiciones necesarias, es cuando se genera el evento y las acciones se dispararán. Mientras que las transiciones pueden contener expresiones, no pueden contener otras transiciones. Además, es importante destacar que una transición simplemente indica el cambio de un estado, mientras que un evento en realidad provoca una acción.

<Condition>:

> Este argumento contiene una condición que ha de cumplirse para que las acciones se activen. Considerando que las transiciones se producen en un determinado punto en el tiempo, las condiciones se mantienen para ciertos períodos.

<Action>:

> Este argumento contiene una o más acciones, que se activará cuando un evento se lleva a cabo y se cumple la condición. En el contexto, una acción es a menudo una notificación.

Jess

Jess es una librería escrita en Java, por lo tanto, para usar jess, se requiere una máquina virtual java (JVM). Se puede obtener JVM para Windows, Linux, Solaris, considerando que jess 7.1 es compatible con todas las versiones publicadas de Java desde el JDK 1.6. Para utilizar el entorno de desarrollo integrado de JessDE, se necesita la versión 3.1 o posterior del SDK de Eclipse desde http://www.eclipse.org. Asegúrese de que Eclipse está instalado y funciona correctamente antes de instalar el JessDE. El lenguaje Jess es una forma altamente especializada de Lisp (Friedman, 2008).

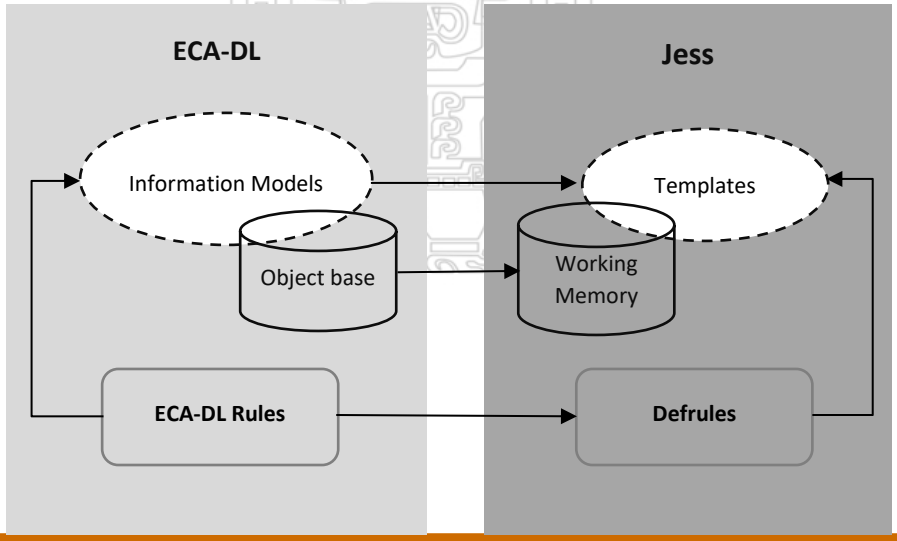
Jess es un motor de reglas para Java, que apoya el idioma Jess. Un motor de reglas (también llamado motor de inferencia) es un programa que intenta hacer coincidir reglas en contra de la información y luego desencadena en una o más acciones. A diferencia de los programas escritos en un lenguaje como C o Java, el lenguaje Jess no especifica cómo debe comportarse el ordenador (por ejemplo, leer la entrada, la salida de impresión, etc.), sólo especifican lo que el equipo debe hacer. Es entonces responsabilidad del motor de Jess para encontrar la manera de hacer mejor lo que se declara. Por lo tanto, a Jess se conoce como un lenguaje declarativo (Maatjes et al., 2007).

Dentro de este contexto para ejecutar reglas de ECA-DL, existen diferentes tecnologías como: CLIPS, Jess, Jdrew y Mandarax, del cual, Jess es el más adecuado. Considerando que, Jess (Java Experto Shell System) es un motor de reglas rápido y de gran alcance que apoya el desarrollo de sistemas basados en reglas y se ejecuta en la plataforma Java (Costa, 2007).

Mapeo de ECA a Jess

La librería Jess sólo puede expresar reglas de proceso en el lenguaje de Jess. Por lo tanto, si deseamos utilizar Jess para ejecutar reglas ECA-DL, necesitamos definir asignaciones de ECA-DL en el lenguaje de Jess. Hemos realizado algunos estudios de casos con el fin de identificar a estas asignaciones sobre la base de nuestra experiencia, definimos las orientaciones de dichas asignaciones, que se pueden utilizar como entrada para la traducción automática (Costa, 2007).

Figura 21: Muestra el enfoque general de diseño de asignaciones



Un modelo de información en ECA-DL consiste en un diagrama de clases UML que representa entidades y contextos, lo que refleja el conocimiento de que el objetivo

manipula aplicaciones sensibles al contexto. Entidades y contexto se representan como clases, y las relaciones entre ellos se definen como asociaciones entre estas clases. Una plantilla en Jess es la estructura estática para definir la estructura de los hechos. Tenemos que definir plantillas antes de afirmar los hechos en la memoria de trabajo del motor de reglas. El primer paso en nuestro enfoque ha sido proporcionar un mapeo a partir de los modelos de información ECA a las plantillas Jess.

Podemos crear instancias de las clases representadas en un modelo de información ECA. Estos casos son los objetos contenidos en la *Object Base* que se muestra en la Fig. 22. De manera análoga a la definición de los objetos de ECA, también podemos afirmar hechos con valores específicos de Jess. Estos hechos reflejan la estructura de las plantillas que están contenidos en la memoria de trabajo del motor de Jess. Por lo tanto, la operación para crear objetos en la ECA-DL corresponde a la operación de afirmar hechos de Jess.(Costa, 2007).

Realización de Reglas ECA-DL en Jess

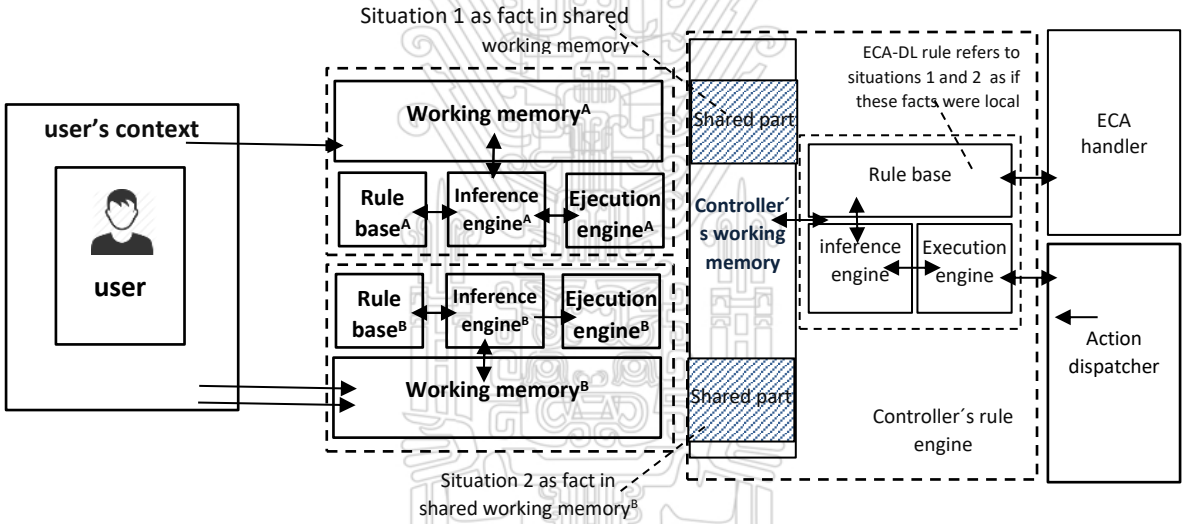
Existen varias alternativas para ejecutar las reglas ECA-DL en una aplicación. Para este propósito se ha decidido utilizar una tecnología madura basada en reglas disponibles *off-the-shelf* para poder ejecutar las reglas ECA-DL. Finalmente se optó por Jess. Para realización de nuestro enfoque basada en reglas. Del mismo modo utilizamos Jess para ejecutar las reglas ECA-DL. *ECA-DL es un lenguaje específico de dominio*, es decir ha sido diseñado con el propósito de definir comportamientos reactivos sensibles al contexto. El lenguaje Jess, es un lenguaje de propósito general adecuado para la diversidad de dominios. Por esta razón, la escritura y comportamientos sensibles al contexto en ECA-DL requiere menos esfuerzo de programación que en el lenguaje Jess.

La aplicación de una única regla ECA en Jess requiere varias reglas Jess. Puesto que las reglas ECA no están soportados directamente por Jess. Además, suponemos que para el mapeo de la memoria de trabajo en el motor del controlador contiene la información necesaria para ejecutar las reglas en cualquier momento en tiempo de ejecución. El contenido de la memoria de trabajo se adquiere a partir de fuentes de contexto, gestores de contexto, y fuentes locales. La información de fuentes de contexto y gestores es típicamente adquirida de forma remota. La información local es proporcionada al controlador por el desarrollador de la plataforma en tiempo de diseño o por el proveedor de la plataforma en tiempo de ejecución (Costa, 2007)

Una solución alternativa para la comunicación con componentes de gestión de contexto para la obtención de información se usa el middleware DJess basado en reglas para permitir la distribución. DJess permite a los motores Jess que se ejecuten en diferentes nodos de una red para comunicarse. Proporciona transparencias para la distribución de contexto y la situación, es decir, cada motor funciona en los hechos distribuidos como si fueran locales. Usando el software DJess como puente, el componente controlador puede convertirse en un miembro de DJess de la inferencia, y en el entorno detecta nodos remotos que se reflejan automáticamente en el controlador de memoria de trabajo.

La figura 22 ilustra un escenario de distribución, en el que dos motores de detección del contexto y un motor de reglas de un controlador participan en una red de inferencia. Las reglas ECA se ejecutan en el contenido de la memoria de trabajo del controlador, que es (parcialmente) compartida con los otros motores.

Figura 22: Usando Djess para el inicio del controlador del motor de reglas



Supongamos que una regla de ECA se refiere a hechos situación 1 y 2, que son detectados por los motores de remotos. Puesto que el controlador y los motores A y B participan en la misma red de inferencia, hechos de la situación 1 y 2 también son percibidos por la memoria de trabajo del controlador. La ventaja de utilizar DJess es que alivia la responsabilidad de gestionar los mensajes de eventos de suscripción y mensajes basados en consultas. Por lo tanto, el diseño y la implementación del controlador componente se simplifican sustancialmente.

Dado que las notificaciones de eventos de la situación ya no se esperan por el componente controlador, la forma en que las cláusulas realizan las consultas en DJess

difiere de la forma en que se realizan de Jess. En el enfoque orientado a servicios, las notificaciones de eventos se identifican mediante un único identificador, que es asignado por el componente controlador de eventos.

Cuando el componente controlador de eventos recibe notificaciones de estos eventos, se crea las instancias correspondientes de *SituationContainedEvents*, que son compartidas en la memoria de trabajo. Usando *DJess*, este mecanismo funciona de manera diferente, ya que no hay ningún componente controlador de eventos de intermediación entre las memorias de trabajo. Para que el componente controlador pueda detectar la ocurrencia de sucesos de una situación (hechos de eventos), incluimos reglas DJess adicionales en el motor de reglas del controlador que detectan estos eventos de forma explícita (Costa, 2007).

Modelado del Contexto

De acuerdo con esta definición" Es el conjunto de posibles condiciones interrelacionadas en las que existe una *entidad*", el contexto sólo es significativo con respecto a una cosa que existe, se llama *entidad*. El concepto de entidad es fundamentalmente diferente del concepto de contexto: el contexto es lo que se puede decir de una entidad en su entorno, es decir, el contexto no existe por sí mismo. El contexto de una entidad puede tener muchos componentes, llamados condiciones de contexto. Ejemplos de condiciones de contexto de una persona son la ubicación de la persona, el estado mental, y la actividad. En conjunto, estas condiciones de contexto forman el contexto de la entidad.

El proceso de identificar el contexto consiste en determinar las condiciones del "contexto" de las entidades en el universo de la aplicación, que son relevantes para una aplicación sensible al contexto o una familia de este tipo de aplicaciones. La representación de estas condiciones o circunstancias se le llama un modelo de contexto.

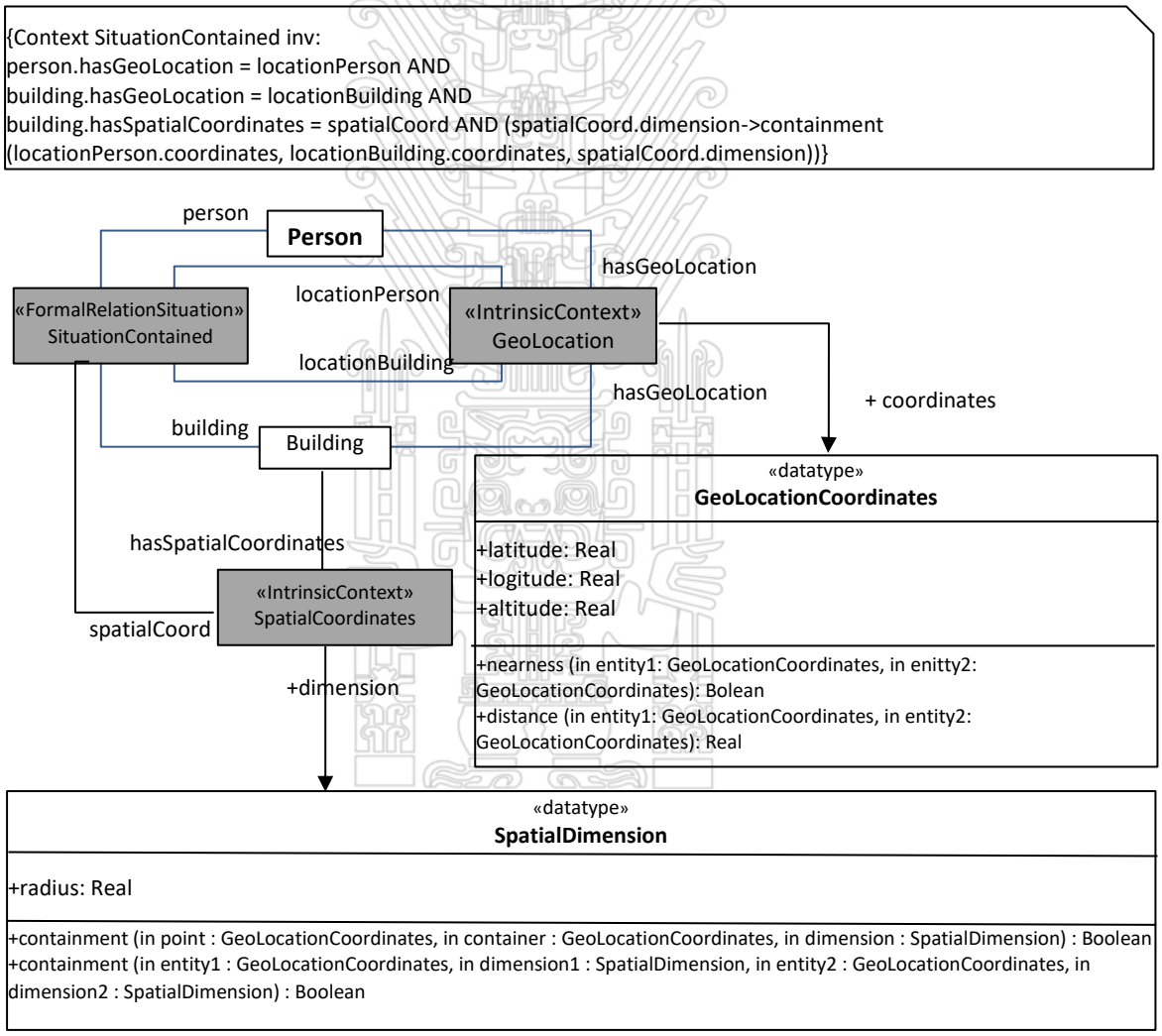
Los modelos conceptuales son, representaciones abstractas de un "*dominio determinado, independiente de diseño específico o las opciones tecnológicas*" (Guizzardi, 2005). Por lo tanto, en un modelo conceptual de contexto, hacemos abstracción de cómo se detecta el contexto. La fase de modelado conceptual es fundamental para el desarrollo de aplicaciones sensibles al contexto, ya que esta fase produce modelos que promueven "*la comprensión, resolución de problemas, y la comunicación, entre las partes interesadas acerca de un tema determinado*". Estos modelos se utilizan como modelo para las fases posteriores del proceso de desarrollo de un sistema. Por lo tanto, la calidad de aplicaciones sensibles al contexto depende de la calidad de los modelos conceptuales de contexto en que se basa su desarrollo. En nuestro enfoque de desarrollo de

aplicaciones sensibles al contexto, el modelado conceptual de contexto precede al diseño detallado de aplicaciones sensibles al contexto (Costa, 2007).

Especificación de la Situación a la Realización de la Situación

La especificación de situaciones se basa en diagramas de clases UML enriquecidos con restricciones OCL 2.0. La figura 23 muestra un ejemplo de una especificación de una situación. *SituationContained* especifica un tipo de situación en la que una persona está dentro de un edificio. Esta especificación se basa en un modelo de contexto que se ha definido anteriormente, En este modelo se han definido los tipos de entidad (*Person and Building*), y los tipos de contexto (*GeoLocation and SpatialCoordinates*), y las relaciones formales (*Containment*), que son los bloques de construcción para la construcción de la especificación *SituationContained*.

Figura 23: Especificación de la Situación



La entidad contenedora, vincula los valores del contexto como: localización de dos entidades especiales, es decir, se comprueba si una entidad está físicamente contenida

en otra entidad, que es una entidad contenedora. En este ejemplo particular, la entidad contenida es una persona, y la entidad recipiente es un edificio.

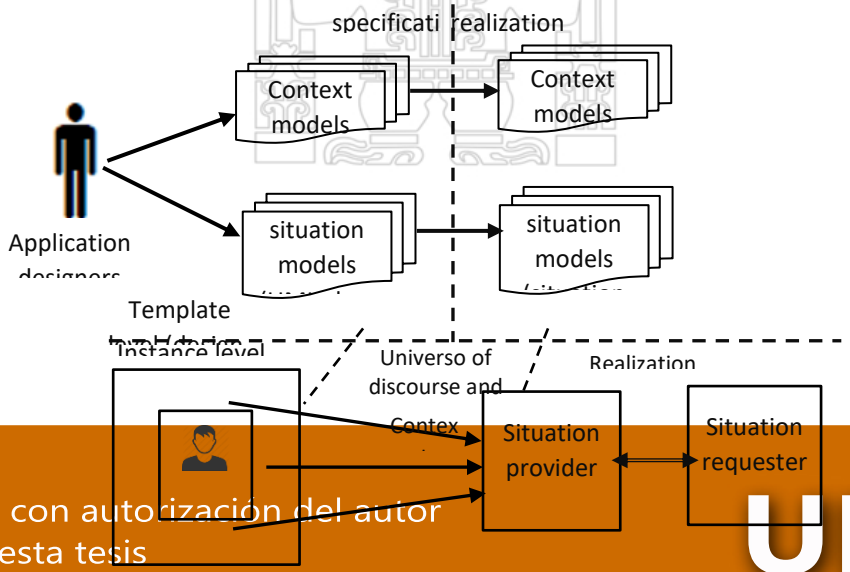
La invariante OCL limita instancias de la clase *SituationContained* a estar asociadas con una persona que se encuentra en un edificio. Por lo tanto, según esta invariante, para todas las personas situadas dentro de un edificio, se crea una instancia de *SituationContained*. También sería posible restringir aún más esta limitación mediante la inclusión, por ejemplo, la identificación de la persona y el edificio para una situación que se creará.

En este ejemplo hemos utilizado los artefactos de especificación para la especificación de la situación ofrecida por nuestro enfoque, los modelos de contexto (clases UML), y los modelos de situación (clases UML y OCL). Estos artefactos son utilizados por los desarrolladores de aplicaciones al inicio del proceso de diseño para restringir su universo. Tras el proceso de diseño, los desarrolladores de aplicaciones necesitan elegir los artefactos de realización para la realización de la situación.

Figura 24 explícitamente separa los elementos de especificación y realización, y se destacan las correspondencias entre ellos. Los modelos de contexto en la fase de especificación corresponden a los elementos del modelo de la fase de realización. Del mismo modo, modelos de situación en la fase de especificación corresponden a elementos de la situación en fase de realización.

La figura 24 también muestra las relaciones entre el contexto del usuario y la aplicación basada en componentes a nivel de instancia (tiempo de ejecución). Los componentes de fuente de contexto proporcionan información que se introduce a los componentes del proveedor de situación (Costa, 2007).

Figura 24: Correspondencias entre las especificaciones UML y elementos de realización



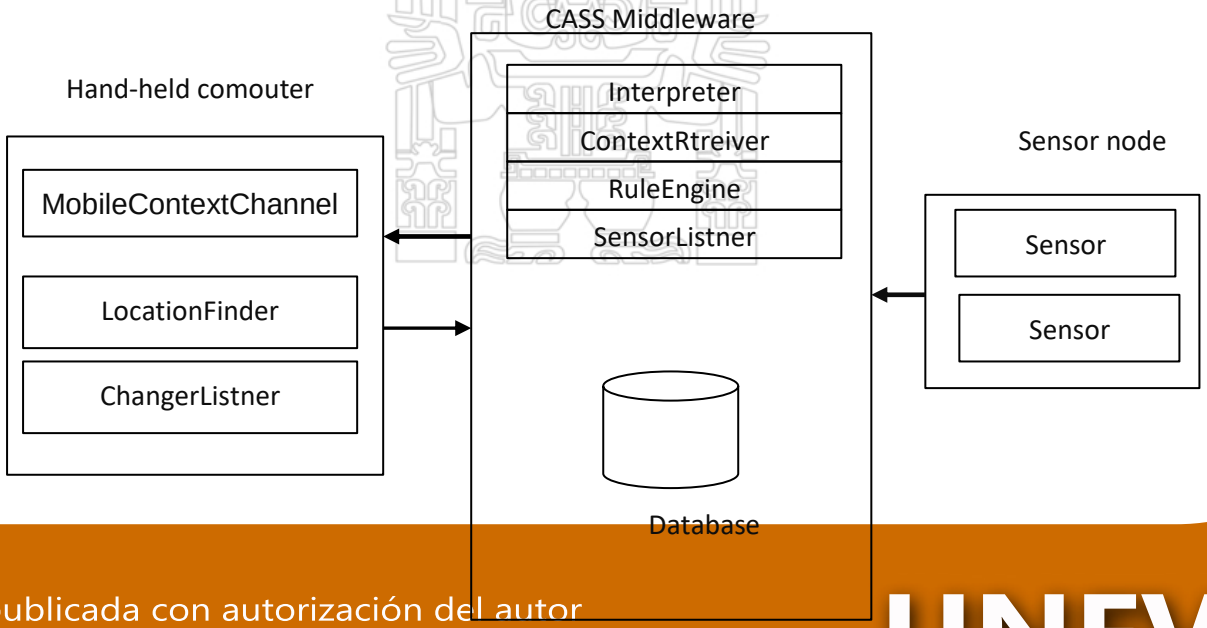
Arquitectura CASS

De acuerdo a (Fahy & Clarke, 2000) CASS (subestructura de conciencia de contexto) es un middleware basado en servidor destinado a soportar aplicaciones contextuales en computadoras portátiles y otros pequeños ordenadores portátiles. CASS permite a los desarrolladores superar las restricciones de memoria y procesador de las plataformas de ordenadores portátiles pequeñas, al tiempo que soporta un gran número de sensores de bajo nivel y otras entradas de contexto. Una característica clave de CASS es su apoyo a la abstracción de datos del contexto de alto nivel y la separación de inferencias y comportamientos basados en el contexto del código de la aplicación. Esta separación abre el camino para que las aplicaciones sensibles al contexto sean configurables por los usuarios.

Consciente de lo que es actualmente el caso. Proporcionar apoyo a los diseñadores de aplicaciones en la integración de una gama más amplia del contexto en sus aplicaciones fomentará el desarrollo de aplicaciones útiles y convincentes para las computadoras de mano.

CASS aborda la cuestión de la separación del código de aplicación contextual desde el razonamiento del contexto de alto nivel y los comportamientos. Más precisamente, esto permitiría que el razonamiento del contexto de una aplicación y los comportamientos resultantes sean cambiados sin re-compilación. Esto abre el camino a los cambios y, de hecho, las extensiones que están realizando los propietarios de una aplicación en lugar de los programadores.

Figura 25: Arquitectura CASS



Arquitectura JCAF

Según (Bardram, 2004) El objetivo del JCAF es crear una infraestructura de servicios orientada a eventos generales, sólida, basada en eventos y un marco genérico y expresivo de programación Java para el despliegue y desarrollo de aplicaciones sensibles al contexto. JCAF incorpora muchas de estas preocupaciones y aquí nos limitaremos a destacar los principios de diseño de la JCAF.

Básicamente, el JCAF se divide en dos partes: una Infraestructura de Tiempo de Ejecución de Contexto y un Marco de Programación de Conciencia de Contexto (API). Los principios básicos de diseño de la infraestructura de tiempo de ejecución son:

- *Servicios Distribuidos y Cooperantes*: Un servicio de contexto puede estar dedicado a un propósito específico, como manejar información de contexto en una casa privada. La mayoría de la administración del contexto es específica para esta casa, pero ocasionalmente puede llegar a ser relevante para contactar a los servicios que se ejecutan en otros hogares. Por lo tanto, una infraestructura de concientización del contexto debería ser distribuida y ligeramente acoplada, manteniendo al mismo tiempo maneras de cooperar de manera peer-to-peer o jerárquica.
- *Infraestructura basada en eventos*: las aplicaciones deben ser capaces de suscribirse a eventos de contexto relevantes y ser notificadas cuando ocurren tales eventos.
- *Seguridad y privacidad*: Datos de contexto, utilizados, por ejemplo. En un entorno médico, debe ser protegido, sujeto al control de acceso, y no revelado a los clientes no autorizados. Además, la credibilidad y el origen de la información de contexto es clave para algún tipo de aplicaciones sensibles al contexto. Estos casos pueden requerir un mecanismo de autenticación para los clientes, e incluso un vínculo de comunicación seguro entre clientes y servicios. Sin embargo, la información de los sensores como la temperatura y la ubicación no suele ser un problema importante de seguridad.
- *Extensible*: La infraestructura debe ser extensible de varias maneras, sin la necesidad de reiniciarla. En primer lugar, debería ser posible implementar, modificar y eliminar servicios de contexto. En segundo lugar, la infraestructura debería apoyar la evolución de los tipos de contexto soportados mediante la carga dinámica de definiciones de contexto, funcionalidad y mecanismos de adquisición, como los nuevos sensores de contexto

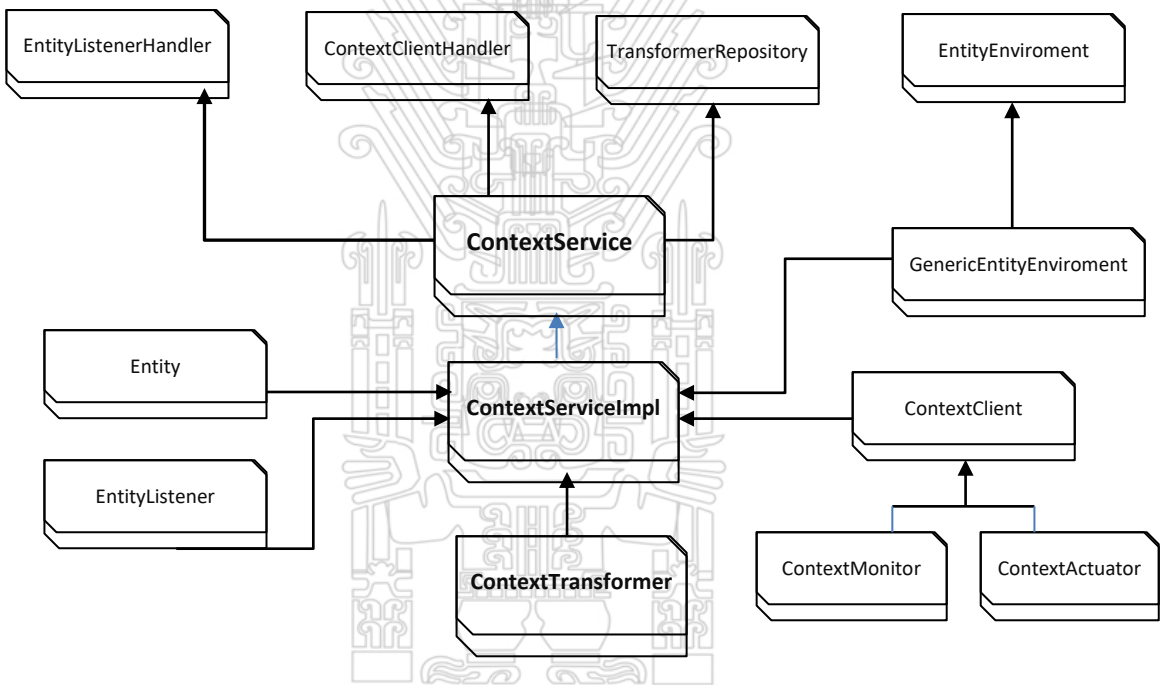
De acuerdo con (Rizzo, 2011) JCAF incorpora conceptos de contribuciones previas sobre context-aware para obtener un framework distintivo en al menos 3 formas.

- La infraestructura JCAFs orientada a servicios es un sistema distribuido basado en la idea de dividir la adquisición, manejo, y distribución en una red cooperativa de servicios de contexto.
- JCAF es una infraestructura robusta, modificable, basada en eventos, que presenta una arquitectura segura.
- JCAF es genérica, extensible, y un modelo expresivo de programación Java para el despliegue y desarrollo de aplicaciones context-aware y modelos de contexto.

Siendo JCAF una infraestructura genérica, extensible, robusta y modificable, es posible establecer una correcta integración con Sakai. Además, existe homogeneidad desde el punto de vista tecnológico.

La incorporación de las APIs JCAF al entorno Sakai permite a los programadores, crear aplicaciones sensibles al contexto integradas en Sakai. La siguiente figura, muestra el diagrama de clases e interfaces de las APIs pertenecientes a JCAF.

Figura 26: Diagrama de clases del Framework JCAF



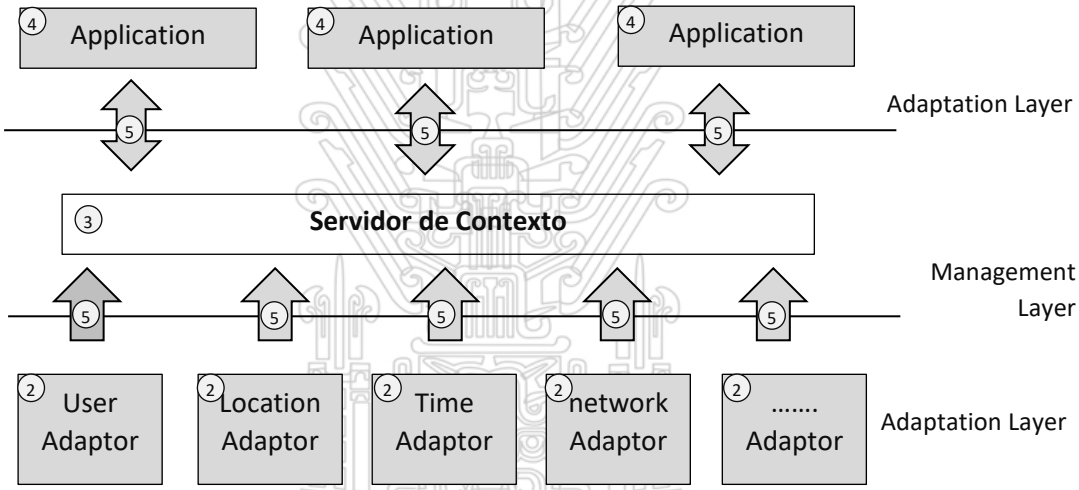
Arquitectura HYDROGEN

Según (Miraoui, Tadj, & Amar, 2011) El Hydrogen es una arquitectura y un marco para los sistemas sensibles al contexto. Se trata de una arquitectura de tres capas que responde a los requisitos particulares de los dispositivos móviles. La arquitectura tiene las siguientes capas: *adaptación, gestión y aplicación*. El servidor de contexto (capa de gestión) contiene toda la información detectada percibida por los sensores de la capa del

adaptador y proporciona información del contexto a la capa de aplicación del dispositivo adjunto otros dispositivos que utilizan un modelo de comunicación peer-to-peer. El enfoque de Hydrogen considera el contexto como cualquier información pertinente sobre un entorno de aplicación y lo describe utilizando un modelo orientado a objetos.

La arquitectura se puede implementar fácilmente, es simple y tiene en cuenta los limitados recursos de los dispositivos móviles (batería, memoria, procesamiento, etc.) y utiliza un modelo de comunicación peer-to-peer (distribuido). La capa del adaptador realiza tanto la tarea de detección como la de interpretación del contexto que no ofrece una buena abstracción del contexto y limita la reutilización de tal componente. Además, lo hace muy dependiente de los sensores. La arquitectura no contiene un módulo de razonamiento en contexto para facilitar la tarea de adaptación.

Figura 27: Arquitectura de Hydrogen



De acuerdo a (Hofer et al., 2003) Las aplicaciones sensibles que conocen el contexto a menudo se dan cuenta de la detección de información del contexto de una manera ad hoc. Los programadores de aplicaciones deben tratar el suministro de la información del contexto, incluyendo la detección del ambiente, su interpretación y su disposición para el procesamiento, además del propósito principal de la aplicación. El estrecho entrelazamiento del manejo de contexto específico del dispositivo con la aplicación obstruye su reutilización con otras configuraciones de hardware (por ejemplo, otros sensores, otros dispositivos).

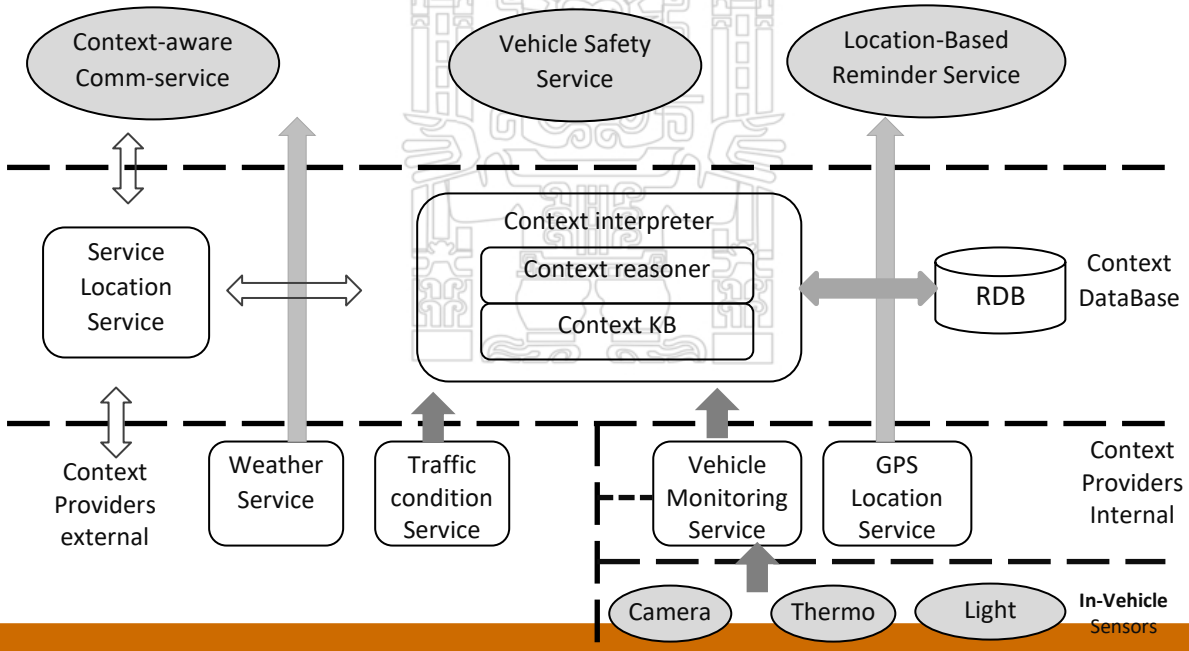
Arquitectura SOCAM

De acuerdo a (Miraoui et al., 2011) SOCAM es una arquitectura de un middleware orientado al servicio del contexto para la construcción y el prototipado rápido de los

servicios móviles sensibles al contexto en un coche inteligente. La arquitectura está compuesta de los siguientes componentes: *proveedor de contexto*, *intérprete de contexto*, (conocimiento de contexto y razonador de contexto), *servicio de localización*, *servicio móvil contextua*l y *base de datos de contexto*. La arquitectura utiliza el modelo cliente / servidor en el que el intérprete de contexto recoge información contextual de los proveedores de contexto (internos o externos) y la base de datos del contexto y los proporciona a los servicios móviles contextuales y al servicio de localización. La principal fuerza de la arquitectura SOCAM es su razonador de contexto que utiliza la ontología para la descripción del contexto y permite un razonamiento sólido sobre el contexto. Utiliza dos clases de ontologías: *ontologías específicas del dominio* y *generalizadas*. Varios sistemas de razonamiento pueden ser incorporados en el intérprete de contexto para soportar una variedad de tareas de razonamiento.

La arquitectura se propuso para apoyar el desarrollo de una pequeña aplicación no distribuida (coche inteligente) que limita su uso en una amplia gama de aplicaciones informáticas. El intérprete de contexto está sobrecargado con una importante calidad de información (ontologías de diferentes dominios) que afecta el rendimiento global del sistema, pero aumenta su reutilización, además del problema principal de una arquitectura centralizada que contradice la naturaleza de un sistema omnipresente que es distribuida con dispositivos autónomos.

Figura 28: Arquitectura SOCAM



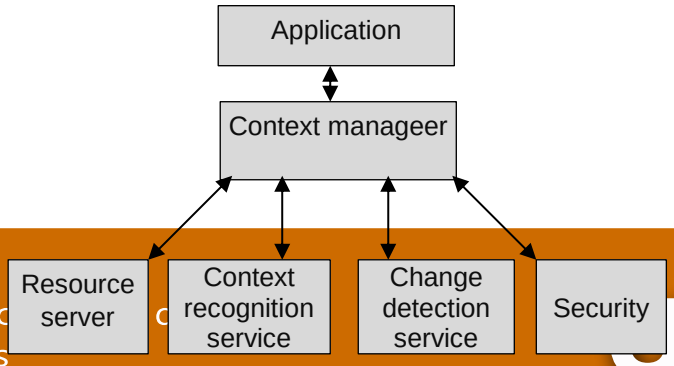
Arquitectura CMF

Según (Miraoui et al., 2011) El CMF (context management framework) permite el razonamiento semántico sobre el contexto en tiempo real e incluso en presencia de ruido, la incertidumbre y la rápida variación del contexto. Proporciona información contextual a las aplicaciones utilizando un modelo de comunicación basado en eventos. El marco propone una arquitectura cliente / servidor compuesto de los siguientes componentes básicos:

- *Context manager*: responsable del almacenamiento de la información contextual en el servidor y la entrega del contexto a los clientes utilizando diferentes tipos de mecanismos (solicitud / respuesta, suscripción / notificación, etc.)
- *Resources server*: responsable de la adquisición de información contextual desde sensores físicos y su interpretación según un formato específico antes de enviarlos al gestor de contexto.
- *Context recognition service:* responsable de la conversión del flujo de datos a una presentación definida en el contexto ontológico.
- *Change detection service:* responsable de la detección del cambio de servicio y, por tanto, del cambio de contexto.
- Security: responsable de la verificación y control de la información contextual.

El CMF utiliza la ontología para la representación del contexto, pero no ofrece un módulo de razonamiento de contexto. Contiene un buen mecanismo para la interpretación del contexto que proporciona una buena abstracción del contexto y mejora la reutilización además de un módulo para la seguridad del contexto. Utiliza un servidor para la gestión de contexto (sistema centralizado), que es el principal problema ya que, cuando el servidor está inactivo, todo el sistema se verá afectado y hará que los dispositivos sean menos autónomos, algo que no es deseable en un sistema informático omnipresente.

Figura 29: Arquitectura CMF



Marco Conceptual

Contexto

Contexto se define como cualquier información relativa al estado de las personas, lugares y objetos que es relevante para la interacción con los usuarios. La información del contexto cambia dinámicamente durante la ejecución de las aplicaciones, referida a la localización, humedad, nivel de ruido, etc.

El conjunto de posibles condiciones interrelacionadas en las que existe una entidad.

Sensibilidad

Refleja la interacción de los dispositivos con el entorno, permitiendo a estos adaptarse al mismo. Las aplicaciones móviles sensibles al contexto son aquellas que pueden extraer, interpretar y usar información del contexto en uso para adaptar sus funcionalidades. El servicio sensible al contexto debe utilizar toda la información contextual del mundo físico que pueda obtener a partir de los sensores.

Computación Ubicua

Computación ubicua, conocida también como inteligencia ambiental. Interacción persona-ordenador se expande a todo el espacio, dando lugar a entornos capaces de adquirir información de forma autónoma y de emplearla para adaptarse a las necesidades de sus ocupantes. Esta interacción entre el entorno y el usuario se ve beneficiada al considerar información contextual como puede ser la localización, la tarea que está realizando el usuario, entre otros recursos que se encuentren cerca y las condiciones ambientales del entorno.

Modelado - UML

El lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

UML se puede usar para modelar distintos sistemas como: sistemas de software, sistemas de hardware y organizaciones del mundo real (Popkin, 2007).

Jess

Jess es un motor de reglas para la plataforma Java. Para utilizarlo, se debe específicar la lógica en forma de reglas utilizando uno de los dos formatos: el lenguaje de reglas Jess (preferido) o en XML. Aunque Jess puede funcionar como un programa autónomo, usualmente integrará la biblioteca Jess en su código Java y es manipulada usando su propia API Java o las facilidades básicas ofrecidas por la API javax.rules.

Se puede desarrollar código de lenguaje Jess en cualquier editor de texto, pero Jess viene incorporado en el entorno de desarrollo Eclipse. (Friedman, 2008).

Mapeo

Es una representación gráfica que define y refleja la estructura y relación de los diferentes objetos o entidades de un sistema.

Sintaxis

La sintaxis se entiende como el conjunto de normas que marcan las secuencias correctas de los elementos propios de un lenguaje de programación

Controlador

Un controlador en un lenguaje de programación es una clase que puede contener acciones, variables, constructores, etc.

Servicios

En informática, un servicio es un conjunto de actividades que buscan responder a las necesidades de una solicitud (cliente) por medio de una red.

Localización

Localización es la ubicación de un objeto o persona que tienen un determinado espacio. El mismo requiere de coordenadas que otorguen puntos de referencia para que esta sea trazable y comunicable. A nivel geográfico, la localización se realiza a partir de la latitud y longitud. En la presente investigación la localización es de enorme importancia para gestionar el contexto.

Plataforma

En informática, una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software (entornos de aplicaciones). Al definir plataformas se establecen los tipos de arquitectura, sistema operativo, lenguaje de programación todos compatibles.

Patrón

En la presente tesis, patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Los patrones por lo general no son fáciles de entender porque tienen un grado de complejidad, pero una vez comprendido su

funcionamiento, los diseños son mucho más flexibles, modulares y reutilizables. Por lo general existe más de un patrón para cada lenguaje de programación.

Componentes

Un componente se define como paquete de software o modulo, que realiza un proceso en específico obteniendo resultados acordes con el proceso realizado. Un componente por lo general está compuesto por uno o varios archivos, los cuales pueden estar compilados o sin compilar (componentes ejecutables o basados en código fuente) (Quality&Programming, 2012).

Aplicación

Una aplicación, también conocida como App, es una aplicación de software que se instala en dispositivos móviles o tablets para realizar una labor concreta, ya sea de carácter profesional o de ocio y entretenimiento.

Stakeholders

El término stakeholder, significa “interesado” o “parte interesada”, y que se refiere a todas aquellas personas u organizaciones afectadas por las actividades y decisiones de una empresa. Este término fue acuñado por primera vez por (Freeman, 1984) en su libro “Gestión estratégica”, en el cual su autor sostenía que estos grupos de interés son un elemento esencial que debe ser tomado en cuenta en la planificación estratégica de los negocios.

Red

Red también llamada red de comunicaciones de datos o red informática, es un conjunto de equipos informáticos y software conectados entres si por medio de dispositivos físico o lógicos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios.

Software

Se conoce como software al soporte lógico de un sistema informático, que comprende un conjunto de componentes lógicos necesarios que hacen posible la realización de tareas específicas. Los componentes lógicos incluyen, entre muchos otros, las aplicaciones informáticas, tales como procesadores de texto, que permite al usuario realizar todas las tareas concernientes a la edición de textos.

Sensores

Un sensor es un dispositivo empleado para convertir una magnitud física o química en una señal generalmente eléctrico que puede de esta forma ser fácilmente procesada, almacenada o transmitida. Es por tanto un elemento transductor que puede transformar una magnitud en otra diferente, en este caso eléctrico, como los dispositivos actuadores, encargados de transformar una magnitud electica en una de otro tipo, generalmente provocando una acción (Fernandez & Estrada, 2005).

FrameWorks

El concepto framework se emplea mucho en el ámbito del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Podemos encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier ámbito que pueda ocurrírsenos(Gutiérrez, 2006).

En general, con el término framework, nos estamos refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta(Gutiérrez, 2006).

Hipótesis

Hipótesis Principal

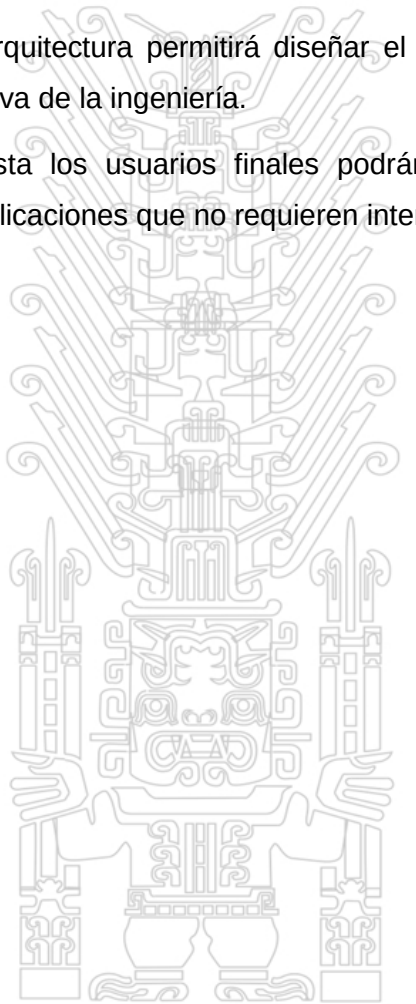
La arquitectura de software propuesta permitirá el desarrollo estándar de aplicaciones sensibles al contexto.

Hipótesis Secundarias

La arquitectura de software, permitirá un desarrollo ordenado, eficaz y estándar para la implementación de sistemas sensibles al contexto.

El soporte y diseño de la arquitectura permitirá diseñar el proceso de información del contexto desde una perspectiva de la ingeniería.

Con la arquitectura propuesta los usuarios finales podrán mejorar su productividad mediante el uso de ciertas aplicaciones que no requieren interacción humana.



CAPITULO III

MÉTODO

Tipo de Investigación

La presente tesis utiliza el tipo de investigación aplicada de carácter tecnológica, pues está orientada a lograr un nuevo conocimiento proponiendo soluciones para el desarrollo exitoso de aplicaciones sensibles al contexto.

Diseño de Investigación

Se utiliza el diseño experimental, pues la presente tesis trata de un estudio donde tiene como objetivo “Diseñar una arquitectura de Software para el desarrollo de aplicaciones sensibles al contexto”. Esto con respecto al problema que presenta los desarrolladores de aplicaciones móviles, para estructurar y comunicar aplicaciones sensibles al contexto. Con el fin de demostrar la viabilidad, se ha construido un prototipo de aplicación para el escenario de la asistencia médica que implementa los productos de diseño obtenidos de las actividades. Con este prototipo somos capaces de medir los problemas de escalabilidad y rendimiento en una aplicación. Finalmente proporcionamos una discusión que analiza nuestro enfoque de desarrollo a la luz de los requisitos presentados.

En el primer punto se presenta es el diseño de la *aplicación* tras el proceso de la arquitectura propuesta. En el segundo punto se presenta la aplicación de gestión de políticas con el contexto, y finalmente se desarrolla el prototipo de la aplicación.

Diseño de la Aplicación

Para el desarrollo del presente caso, se presenta un escenario de epilepsia en un puesto de salud, este escenario se ha mencionado en varias ocasiones en los patrones presentados en el marco teórico. Por ejemplo, se presentó un escenario de un paciente epiléptico, con el fin de detectar o predecir las crisis epilépticas. Tras un ataque epiléptico, una serie de medidas pueden tomarse como advertencia de un próximo ataque, por ejemplo, el envío de mensajes a los familiares que se encuentran cerca de él. Este escenario se utiliza en el proyecto sensibilización y su relevancia en la mejora de la calidad de vida de los pacientes, esto ha sido confirmada por el instituto de investigación y desarrollo Roessingh (Roessingh, 2000). Este instituto de investigación es reconocido internacionalmente, que contribuye a la mejora de la medicina de rehabilitación en con énfasis en la tecnología.

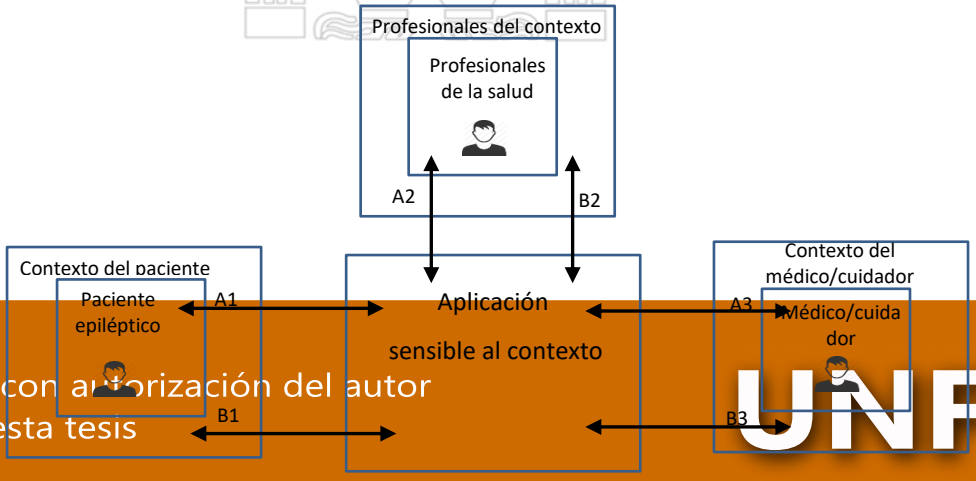
El escenario es el siguiente: "*El Sr. Carlos es un paciente epiléptico y a pesar de sus medicamentos, todavía sufre convulsiones. Debido a su condición médica, el Sr. Carlos requiere una vigilancia constante para que los médicos estén alertados de un posible ataque severo. Recientemente, el Sr. Carlos se ha dotado de una aplicación sensible al contexto de tele-monitoreo capaz de monitorear pacientes epilépticos y proporcionar asistencia médica momentos antes y durante una crisis epiléptica. Esta aplicación es*

capaz de medir la variabilidad de la frecuencia cardíaca, la actividad física, predecir convulsiones y contactar de forma automática a familiares o médicos que se encuentra cerca de él. Además, el Sr. Carlos puede ser informado con anticipación sobre un posible ataque epiléptico y detener las actividades en curso, tales como detener un auto, etc. El objetivo es proporcionar al Sr. Carlos los dos niveles más altos de seguridad y la independencia que le permite funcionar con mayor libertad en la sociedad a pesar de su trastorno".

Con el fin de poner en práctica este escenario, se propone una aplicación móvil sensible al contexto, es decir, la aplicación consciente del contexto de la salud. El objetivo de esta aplicación es detectar las convulsiones, y reaccionar de la siguiente manera: (a) notificar al paciente epiléptico de posibles convulsiones, y (b) notificar a su / sus cuidadores más cercanos de próximas convulsiones del paciente al mostrar un mapa con la ubicación del paciente. Los cuidadores que reciben la notificación de ayuda deberían ser asignados como uno de los cuidadores de ese paciente en particular, físicamente es el que está cerca del paciente. Con respecto a la notificación en busca de ayuda, los médicos pueden aceptar o rechazar la solicitud de ayudar al paciente epiléptico. Cuando un médico particular, acepta ayudar, los otros médicos que habían recibido la notificación de ayuda son informados de que un determinado medico ya ha aceptado para ayudar a ese paciente.

La Figura 30 representa una visión intuitiva de los tipos de usuarios admitidos en este escenario, su contexto, y la aplicación consciente del contexto de la salud. Nos centramos en esta figura en donde existe una instancia de usuario para cada tipo de usuario compatible. Se muestran tres tipos de usuarios, es decir, el paciente epiléptico, el profesional de la salud y el cuidador. Las flechas de tipo A1, A2 y A3 muestran la interacción de la aplicación de la salud y los usuarios. Del mismo modo, las flechas de tipo B1, B2 y B3 muestran que los usuarios del contexto y la aplicación interactúan.

Figura 30: Vista informal de la aplicación



Para los tres tipos de usuarios, las interacciones del tipo "a" permite las entradas del usuario a la aplicación sensible al contexto, y definir las preferencias de configuración. Además, las interacciones representadas por las flechas de tipo A1 permiten que un paciente reciba una notificación de la próxima crisis epiléptica que se entrega por la aplicación. Las interacciones representadas por las flechas de tipo B1 permite que la aplicación capture determinadas condiciones del contexto a partir del contexto del paciente epiléptico, tales como sus señales biológicas (por ejemplo, medidas de la frecuencia cardíaca y la presión arterial), e información de ubicación. Un paciente epiléptico puede estar realizando una actividad potencialmente peligrosa, tales como conducir un auto, manipulando un cuchillo. Las interacciones representadas por las flechas de tipo B1 permiten que la aplicación capture información de si el paciente está realizando una actividad peligrosa o no.

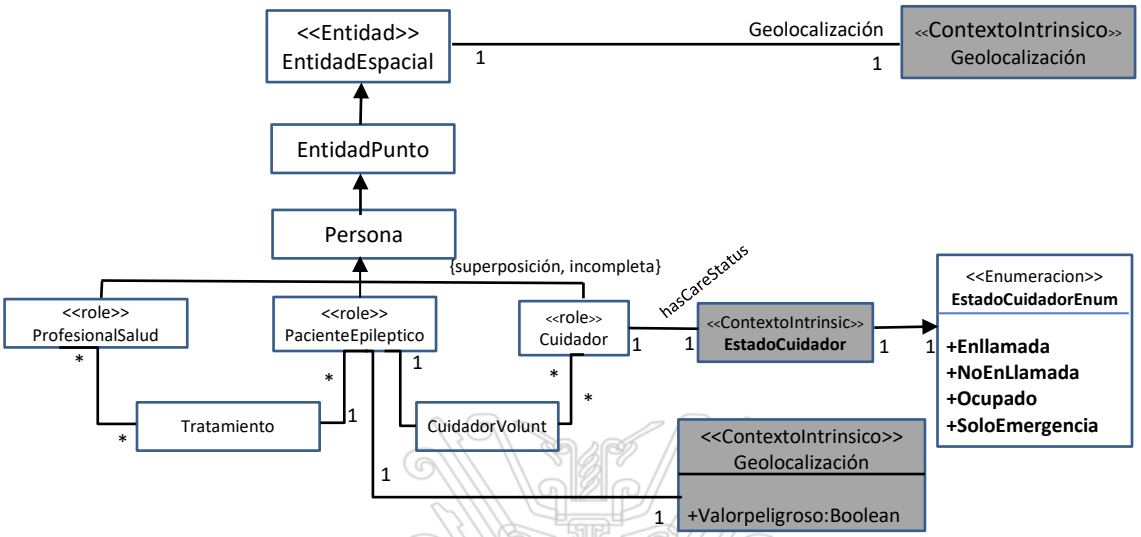
Del mismo modo, las interacciones representadas por las flechas de tipo A2 permiten a los profesionales de la salud recibir notificaciones de alarmas de ataque epiléptico para monitorear las señales biológicas del paciente con el fin de detectar anomalías peligrosas. Las interacciones representadas por las flechas de tipo B2 permiten que la aplicación capture determinadas condiciones del contexto de los profesionales de la salud, tales como su información de ubicación.

Las interacciones representadas por las flechas de tipo A3 permiten a un cuidador recibir notificaciones para ayudar a determinados pacientes epilépticos con convulsiones próximas. El cuidador puede aceptar o rechazar las solicitudes. Las interacciones representadas por las flechas de tipo B3 permiten que la aplicación capture determinadas condiciones del contexto del cuidador, tales como su ubicación e información de su disponibilidad. El estado de disponibilidad del cuidador podría ser "de guardia", "en la llamada", "ocupado", o "solamente de emergencia", que se explican en detalle en el modelado del contexto.

Modelado del Contexto

De acuerdo con nuestro enfoque de desarrollo, el primer paso hacia la realización de aplicaciones, es el modelado del universo de la aplicación. Nuestra metodología de modelado del contexto y la situación ha sido ampliamente discutida en el marco teórico. En el análisis del escenario de la aplicación se identifican los tipos de entidad, contexto y la situación, necesarios para modelar la aplicación. La Figura 18 muestra el modelo de contexto que representa el universo de esta aplicación con respecto a los tipos de entidad y el contexto pertinente.

Figura 31: Modelo de contexto de la aplicación



Identificamos los roles que una persona puede jugar en este escenario, que son HealthProfessional, EpilepticPatient, y el cuidador. Un HealthProfessional puede ser un doctor, una enfermera o una persona entrenada que es capaz de proporcionar un tratamiento profesional a un paciente epiléptico. Un EpilepticPatient representa las personas que sufren de una condición médica de epilepsia. Por último, el cuidador representa las personas que se han ofrecido para ayudar a los pacientes epilépticos que tienen un ataque epiléptico. La relación entre un profesional de la salud y un paciente epiléptico se caracteriza por un tratamiento, que está representado por la clase de tratamiento. Del mismo modo, la relación entre un paciente epiléptico y sus cuidadores se caracteriza por una relación de cuidado voluntario, que está representado por la clase VoluntaryCare.

Se han identificado tres tipos de contexto intrínsecos: la ubicación geográfica de la persona (GeoLocation), el estado de un cuidador (CareStatus), y la información acerca de si un paciente está haciendo actualmente una actividad peligrosa (HazardousActivity). Los cuidadores pueden configurar su estado *OnCall*, que especifique que están disponibles actualmente para recibir solicitudes para ayudar a los pacientes; n*otOnCall*, especifica que no están disponibles para la recepción de solicitudes de ayuda, *busy*, especifica que actualmente está recepcionando solicitudes, pero están ocupados en este momento; emergencyOnly, especifica que están actualmente disponibles para la recepción de solicitudes sólo en situaciones de emergencia. La ubicación geográfica de una persona está representada por la *GeoLocationCoordinates* (Figura 31), que especifica la latitud, longitud y la altitud de la ubicación actual de las personas. Además, en este tipo de datos se especifica la relación formal de cercanía y distancia de las operaciones. Un paciente epiléptico también puede estar haciendo una actividad

potencialmente peligrosa, que es capturado por un atributo booleano de la clase HazardousActivity.

Con el fin de saber si una persona es un cuidador de un paciente, definimos una operación estática en la clase Caregiver *(isCaregiver (Caregiver c, EpilepticPatient p))*, que recibe un cuidador C y paciente P como argumentos, y devuelve verdadero si el paciente P puede ser ayudado por el cuidador C, y falso en caso contrario. El cuerpo de este método se define en OCL de la siguiente.

context Caregiver::isCaregiverOf (caregiver:Caregiver, patient: EpilepticPatient): Boolean body: caregiver:VoluntaryCare->exists(care | care.EpilepticPatient=patient)

Del mismo modo, con el fin de saber si una persona es uno de los profesionales de la salud que tiene relación con el tratamiento de un paciente determinado, definimos una operación estática en la clase *HealthProfessional*, *isHealthProfessionalOf (HealthProfessional hp, EpilepticPatient p)*. Este método recibe dos argumentos: a los profesionales de la salud (hp) y al paciente (p), y devuelve verdadero si *h*p es uno de los profesionales de la salud del paciente *p*, y falso en caso contrario. El cuerpo de este método se define en OCL de la siguiente.

context HealthProfessional:: isHealthProfessionalOf (professional:HealthProfessional, patient: EpilepticPatient): Boolean body: profesional.Treatment ->exists(prof | prof.EpilepticPatient=patient)

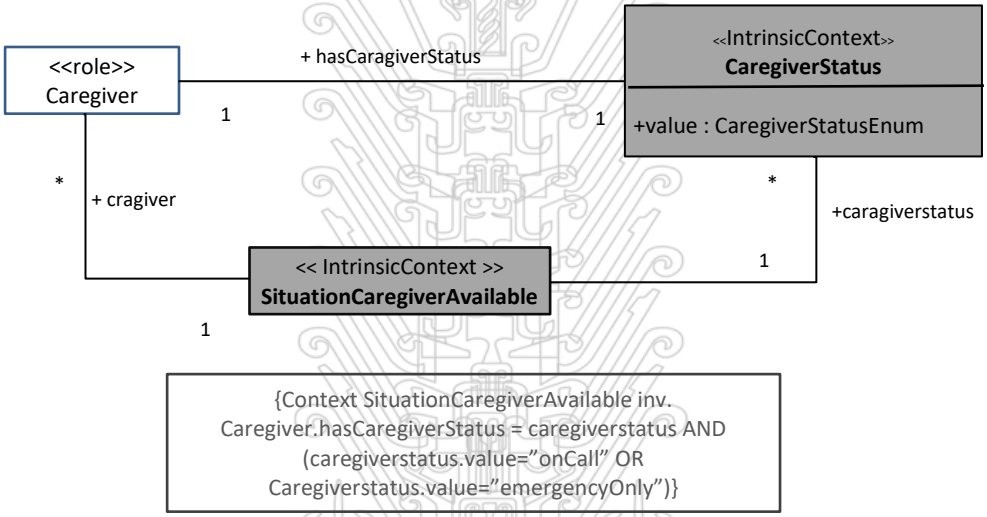
La alarma de ataque epiléptico es generada por los dispositivos conectados al cuerpo del paciente. Estos dispositivos recogen señales biológicas del paciente con el fin de predecir un ataque epiléptico. Roessingh Investigación y Desarrollo ha trabajado en un algoritmo de predicción que detecta ataques en base a ciertos patrones de frecuencia cardiaca y la variación de la presión arterial. Alarmas epilépticas se generan automáticamente a partir de bioseñales, los pacientes son capaces de apagar la alarma en caso de una predicción errónea.

La detección de ataques epilépticos genera alarmas de convulsiones, que pueden definirse como eventos primitivos. Alarma de eventos epilépticos están representados por *(EpilepticAlarm (EpilepticPatient)),* y el evento generado por el paciente cuando se apaga la alarma está representado por *(RejectEpilepticAlarm (EpilepticPatient)).* Cuando los médicos están cerca del paciente reciben una notificación de que un determinado paciente puede estar necesitando de ayuda, un cuidador puede aceptar o rechazar la solicitud. La aceptación o rechazo genera sucesos primitivos, que se origina a partir de los dispositivos de los cuidadores. Estos eventos primitivos están representados por

AcceptHelpRequest (EpilepticPatient, Caregiver), y *RejectHelpRequest (EpilepticPatient, Caregiver)*, respectivamente.

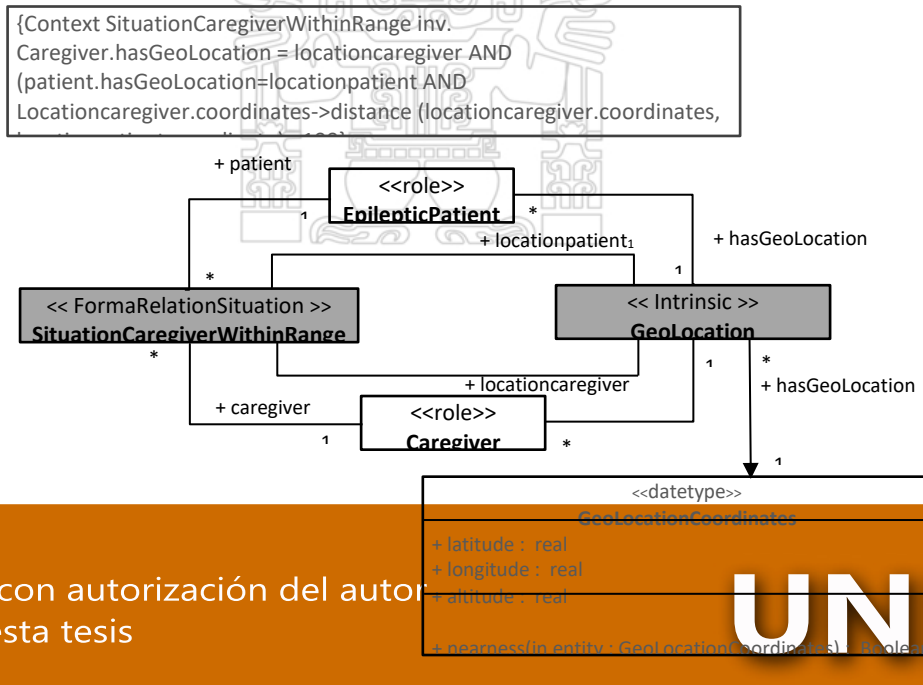
Tras el proceso de diseño, la aplicación es modelada por medio de modelos de situación. Se definen dos tipos de situaciones, que son de interés para el escenario de epilepsia, es decir *SituationCaregiverAvailable*, y *SituationCaregiverWithinRange*. El tipo de situación *SituationCaregiverAvailable* especifica si los cuidadores están disponibles cuando su estado se establece en "OnCall" o "emergencyOnly". La Figura 32 representa la especificación de esta situación utilizando nuestro enfoque de modelado.

Figura 32: Especificación de la Situación SituationCaregiverAvailable



La Figura 33 representa la especificación de la situación SituationCaregiverWithinRange, en el que un paciente y un médico están a 100 metros de distancia uno de otro.

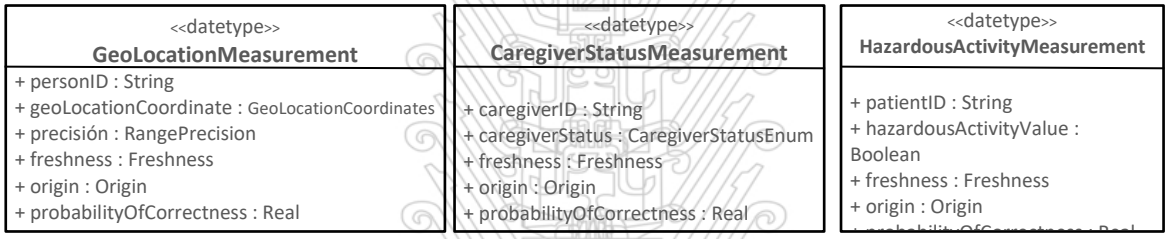
Figura 33: Rango entre paciente y medido: SituationCaregiverWithin



Modelado de la Información del Contexto

En la sección del modelado del contexto, hemos presentado los modelos conceptuales de contexto sin tener en cuenta las características de la información del contexto de la aplicación, tales como, por ejemplo, la calidad de la información del contexto, los tipos de información que se intercambia entre los componentes en nuestro contexto, a diferencia de los tipos de contexto. Al modelar los tipos de información de contexto consideramos si se detecta, se deriva o aprende, y tomamos la calidad del contexto en cuenta. Figura 34 representa los tipos de información de contexto intercambiados en el escenario epiléptico.

Figura 34: Tipos de datos de la aplicación



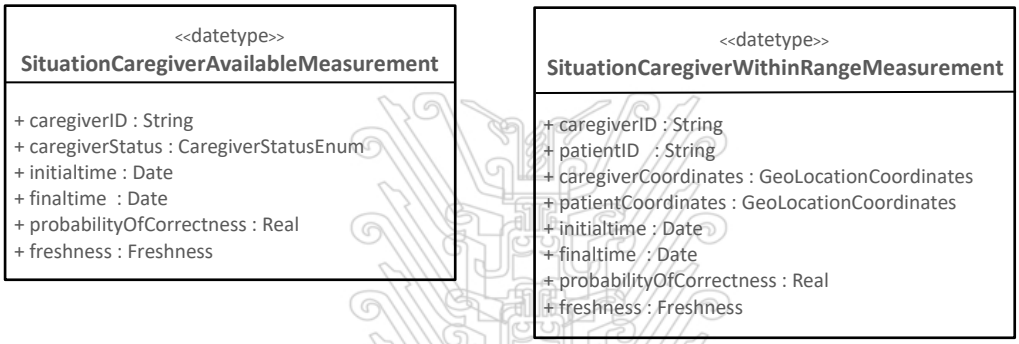
Estos tipos de datos de medición se refieren a valores seriados de los respectivos tipos de contexto. Por ejemplo, el tipo de datos *GeoLocationMeasurement* se refiere a las personas, es un identificador único (atributo *personID*), y sus medios geográficos de localización (atributo geoLocationCoordinates con su tipo de dato *GeoLocationCoordinates*). Además, se utilizan diversos atributos de calidad de contexto. Para el tipo de dato *CaregiverStatusMeasurement* definimos dos atributos, que indican el identificador único del cuidador (*caregiverID*) y su estado de disponibilidad (*caregiverStatus*). Del mismo modo, el tipo de dato *HazardousActivityMeasurement* define dos atributos, que indican la identificación del paciente (*patientID*) y si él/ella está experimentando una actividad peligrosa (*hazardousActivityValue*).

Los eventos primitivos: *EpilepticAlarm* (*EpilepticPatient*), *RejectEpilepticAlarm* (*EpilepticPatient*), *AcceptHelpRequest* (*EpilepticPatient*, cuidador), y *RejectHelpRequest* (*EpilepticPatient*, cuidador) son soportados por la aplicación. La Figura 23 muestra las estructuras de notificación de eventos, cuyo objetivo es notificar a los solicitantes de las apariciones de estos eventos. Posibles parámetros para las notificaciones de eventos se especifican como atributos de tipo de datos.

Figura 35: Tipos de datos de Notificación de eventos

Con respecto a los tipos de situación, dos tipos de mediciones son necesarias, a saber *SituationCaregiverAvailableMeasurement*, y *SituationCaregiverWithinRangeMeasurement*. Estos tipos de datos de medición se refieren a valores en serie de casos de situación, es decir, los identificadores únicos de entidades, los valores de contexto como valores de tipo de datos (por ejemplo, *GeoLocationCoordinates*), y los tiempos iniciales y finales de la situación. La Figura 36 representa estas mediciones tipos de datos.

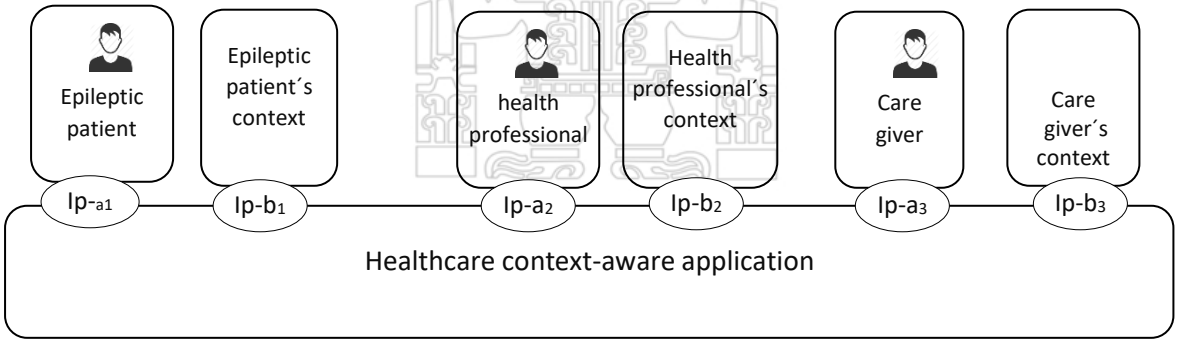
Figura 36: Medición de tipos de datos de situación



Diseño Estructural de la Aplicación

La Figura 37 muestra una visión general de la aplicación sensible al contexto mediante la representación gráfica adoptada por esta tesis. Este estilo particular para las aplicaciones y plataformas sensibles al contexto, ha sido ampliamente discutido en los patrones de arquitectura. Las interacciones que tienen lugar en los puntos de interacción del modelo ip-a_i y ip-b_i, son actividades que se realizan en cooperación entre los usuarios y la aplicación sensible al contexto respectivamente. Todos los posibles tipos de información que pueden ser establecidas tanto en ip-a y ip-b se definen en un modelo de información de contexto.

Figura 37: Visión general de la Aplicación



Con el fin de hacer frente a la complejidad y al coste-efectividad, la aplicación sensible al contexto se desarrolla utilizando los servicios ofrecidos por nuestra plataforma de manejo de contexto. La figura 38 ilustra la aplicación desarrollado con el apoyo de nuestra plataforma de manejo de contexto. La aplicación sensible al contexto utiliza los servicios

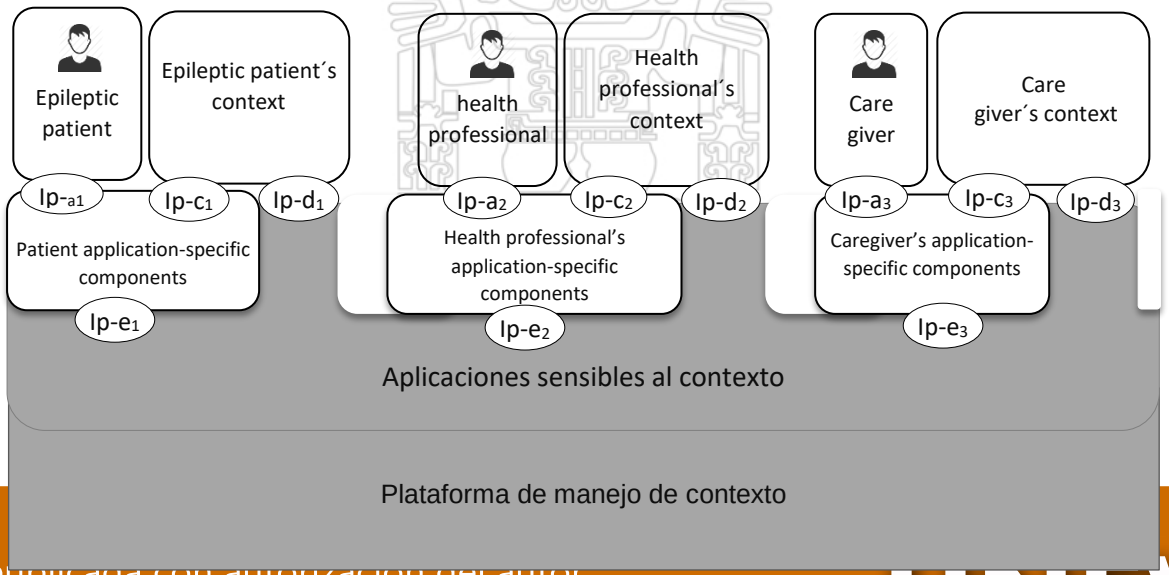
genéricos ofrecidos por la plataforma (área gris en la Figura 38) y también implementa servicios específicos, que son ofrecidos por los componentes de la aplicación. Los servicios específicos de la aplicación consisten normalmente en las funciones específicas que no valen generalizar, ya que están ligados a la finalidad de la aplicación. Estos servicios no deben ser incluidos en la plataforma de manipulación de contexto. Algunos valores de la información de contexto a menudo son sensibles a la privacidad, y por lo tanto no deben ser compartidos.

Los componentes específicos de la aplicación del paciente deben incluir también las funciones para detectar el próximo ataque epiléptico, que son bastantes específicas para ser generalizadas en la plataforma. Además, los componentes específicos de la aplicación del paciente deben ser capaz de determinar si el paciente se encuentra actualmente en una actividad peligrosa o no.

Componentes específicos de la aplicación debe incluir la funcionalidad para filtrar y procesar la información de streaming de bioseñales. Los componentes específicos de la aplicación del cuidador deben incluir la funcionalidad para capturar el estado del cuidador, y para capturar la aceptación o el rechazo del cuidador para una solicitud concreta de ayuda.

Todos los componentes específicos de la aplicación son desarrollados por los desarrolladores de aplicaciones. Estos componentes se ejecutan en el dispositivo móvil llevado por sus respectivos usuarios. Por lo tanto, para cada usuario en particular hay casos particulares de los componentes específicos de la aplicación, que se ejecutan en el dispositivo móvil del usuario.

Figura 38: Plataforma que ofrece apoyo a la aplicación sensible al contexto



En la Figura 38, los usuarios interactúan con los componentes específicos de la aplicación a través de los puntos de interacción de tipo *ip-a*. Puntos de interacción de tipo *ip-c* y *ip-d* que permite a los usuarios interactuar con los componentes específicos de la aplicación y con la plataforma, respectivamente. Puntos de interacción de tipo *ip-c* representan los mecanismos que se utilizan para capturar el contexto, que son específicos de la aplicación y no pueden ser compartidos, tales como la detección de un próximo ataque epiléptico. De forma análoga, los puntos de interacción de tipo *ip-d* representan los mecanismos que se utilizan para capturar información del contexto y pueden ser reutilizados por otras aplicaciones a través de la plataforma. Por ejemplo, el mecanismo que se utiliza para capturar la información de ubicación geográfica se puede compartir entre varias aplicaciones sensibles al contexto, y es, por lo tanto, proporcionada por el manejo de la plataforma de contexto. Los puntos de interacción de tipo *ip-e* permiten interacciones entre los componentes específicos de la aplicación y la plataforma. Esto permite, por ejemplo, los servicios específicos de la aplicación para hacer uso de la información del contexto.

Servicios de Aprovisionamiento del Contexto

Hasta ahora, hemos modelado el universo de la aplicación, especificando los tipos de contexto y situaciones pertinentes, respectivamente; identificado el contexto y la situación de tipos de datos de medición de la información, la aplicación tiene mucho cuidado en partes específicas de la aplicación y las partes genéricas, y se distingue la funcionalidad específica y genérica que se deben realizar en las partes específicas de la aplicación y en la plataforma compartida, respectivamente. Ahora debemos ser capaces de identificar los componentes del procesador de contexto (fuentes de contexto), que son necesarios para la captura de los tipos de información de contexto y situación pertinente.

Los componentes de las aplicaciones que son capaces de ofrecer las alarmas epilépticas, información de actividad del paciente, información del estado del cuidador y de aceptación del cuidador o rechazo para las notificaciones de ayuda, desempeñan el papel de fuentes de contexto. Los servicios ofrecidos por estos componentes deben estar registrados como servicio de modo que la plataforma es capaz de encontrarlas cuando sea necesario. La información sobre la ubicación geográfica puede ser proporcionada por la plataforma de manejo de contexto, que es capaz de ofrecer información de ubicación geográfica basada en triangulación celular GSM.

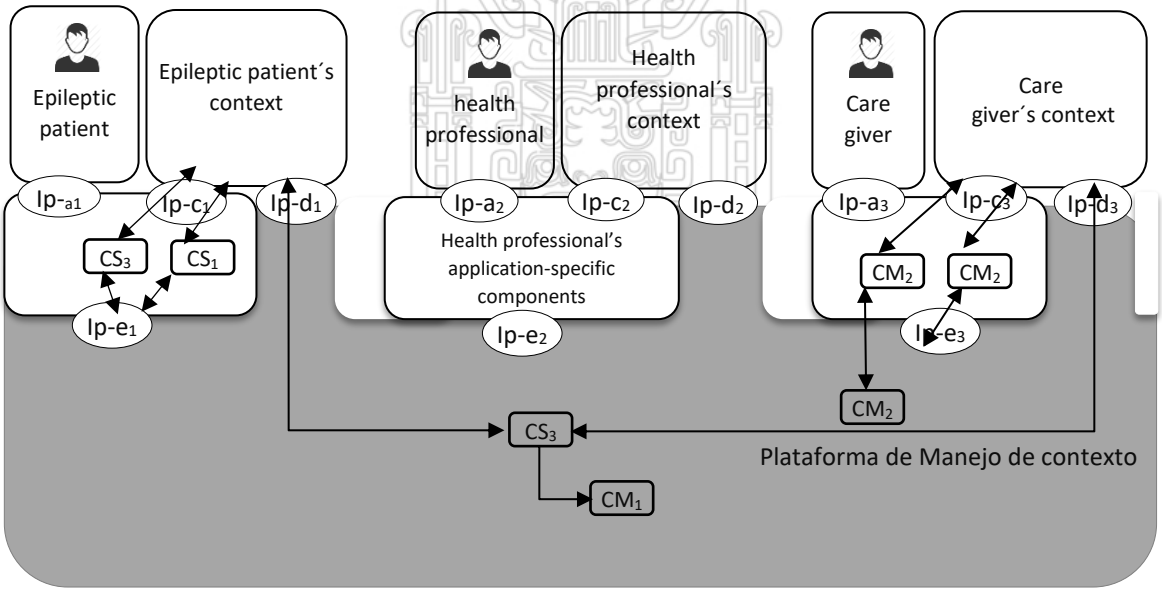
También necesitamos componentes gestores de contexto capaces de detectar los tipos de situaciones e identificarlos. Los servicios ofrecidos por estos componentes gestores de contexto también deben estar registrados en el servicio de descubrimiento de modo que la plataforma es capaz de encontrarlas cuando sea necesario.

Teniendo en cuenta estos requisitos de información contextual, y la configuración estructural de la aplicación, llegamos a la conclusión de que las siguientes fuentes de contexto deben desarrollarse: *EpilepticAlarmContextSource* (CS_1), *GeoLocationContextSource* (CS_2), *HazardousActivityContextSource* (CS_3), *CaregiverStatusContextSource* (CS_4), y el *AcceptRequestContextSource* (CS5).A excepción de la *GeoLocationContextSource* (CS_2), todas las otras fuentes de contexto son específicos, y deben llevarse a cabo como parte de las partes específicas de la aplicación.

Teniendo en cuenta los requisitos de información de situación, llegamos a la conclusión de que los siguientes gestores de contexto deben desarrollarse: *SituationWithinRangeContextManagers* (CM_1), y *SituationAvailableContextManager* (CM*2*). Estos componentes de gestores de contexto se pueden generalizar, y por lo tanto, son parte de la plataforma de manejo de contexto.

La figura 39 muestra las fuentes de contexto y gestores como parte de los componentes específicos de la aplicación, y parte de la plataforma de manejo de contexto.

Figura 39: Fuentes de contexto en los componentes específicos de la aplicación en la plataforma



Varias alternativas son posibles para la realización y poner en práctica las fuentes de contexto. Los desarrolladores de aplicaciones son libres de elegir la alternativa más preferible para sus propios fines. Un ejemplo de una posible alternativa para la EpilepticAlarmContextSource (CS_1) se implementa el algoritmo para la detección del próximo ataque epiléptico. CS_1 interactúa con el contexto del paciente a través de los

puntos de interacción de tipo ip-c_1 con el fin de recopilar señales biológicas del paciente. Una posible aplicación de la *GeoLocationContextSource* (CS_2) calcula ubicaciones geográficas de los usuarios utilizando un mecanismo de localización basada en celulares -GSM, y es implementado por los desarrolladores de la plataforma. CS_2 interactúa con los pacientes y contextos del cuidador a través de los puntos de interacción de tipo ip-d con el fin de obtener información de señalización desde sus dispositivos móviles.

Una posible aplicación de la *HazardousActivityContextSource* (CS_3) realiza una Entrada explícita proporcionada por el paciente, cuando él / ella se somete a una posible actividad peligrosa. Como ya hemos comentado anteriormente, en nuestra definición de información de contexto que no distinguen entre proporcionar información de contexto de forma manual o automática. Aunque en esta solicitud algunos valores de la información de contexto se proporcionan manualmente por el usuario, estos valores siguen representando las condiciones en el contexto del usuario. Por lo tanto, estos valores deben ser recogidos a través de los puntos de interacción de tipo ip-c_i. Del mismo modo, el *CaregiveStatusContextSource* (CS_4), y el AcceptRequestContextSource (CS_5) se pueden realizar por medio de entradas explícitas proporcionadas por el cuidador, que proporciona información de su estado actual, y si él/ella acepta o no la petición de ayuda, respectivamente.

Ambos componentes de gestores de contexto como el SituationWithinRangeContextManager (CM_1) recopila información sobre la ubicación de la GeoLocationContextSource (CS_2) con el fin de detectar si hay cuidadores cercanos o no para los pacientes. El SituationAvailableContextManager (CM_2) recoge información sobre el estado de la CaregiveStatusContextSource (CS_4) con el fin de razonar acerca de si el cuidador está disponible o no.

La figura 40 muestra las interfaces de origen de contexto, que deben ser ofrecidos por las fuentes de contexto que acabamos de definir. El *GeoLocationContextSource* proporciona notificaciones de medición de información de ubicación de vez en cuando, por ejemplo, en cada minuto. El *CaregiverStatusContextSouce* y el *HazardousActivityContextSource*, por el contrario, proporcionan notificaciones de mediciones de estado del cuidador y las medidas de actividad peligrosa cada vez que estos valores son modificadas por el cuidador y el paciente.

Figura 40: Las Interfaces de origen de contexto esperando a ofrecer datos de medición del contexto.

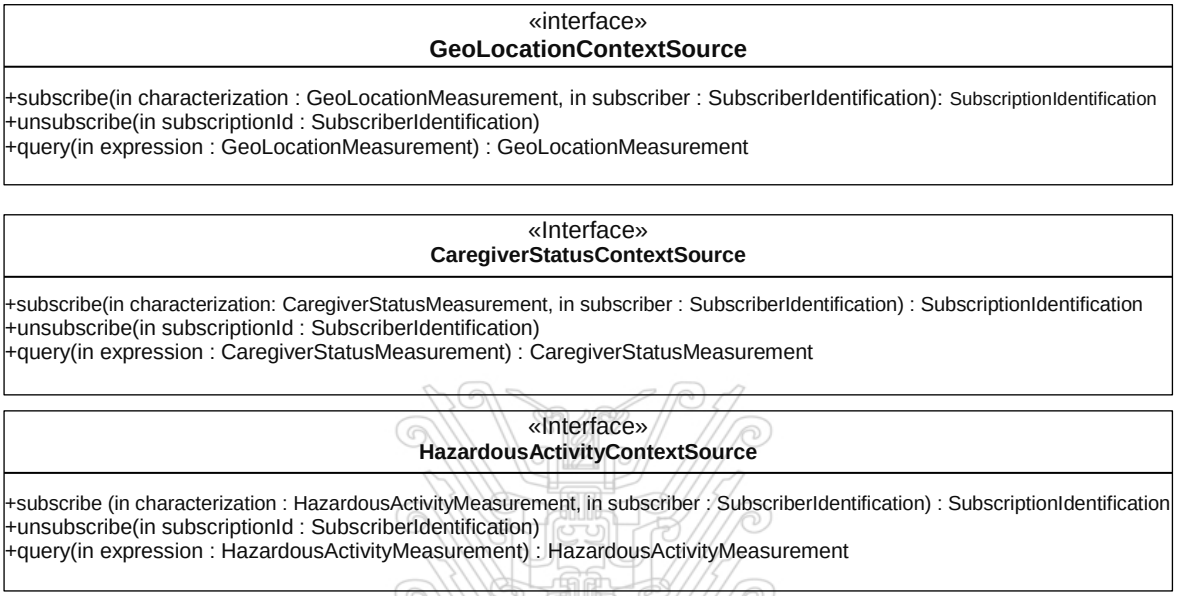
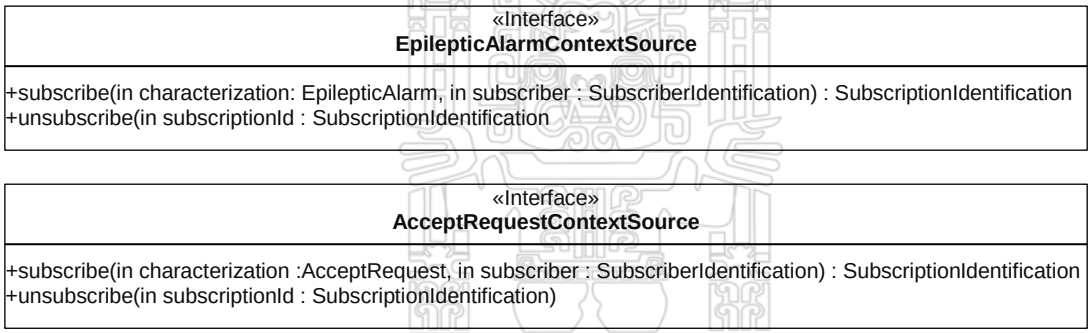


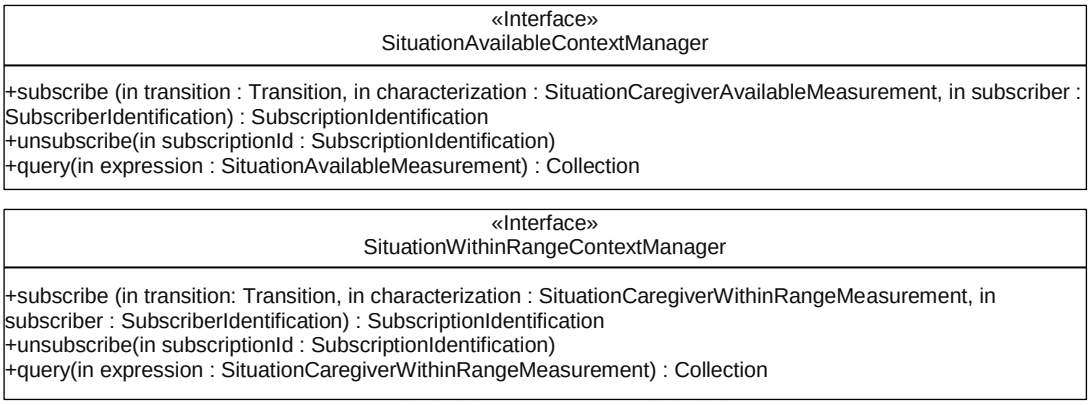
Figura 41 representa interfaces de origen de contexto que proporcionan notificaciones de eventos primarios, es decir, las interfaces de fuentes de contexto: EpilepticAlarmContextSource y AcceptRequestContextSource.

Figura 41: Las Interfaces de origen de Contexto esperando que ofrezcan datos de notificación de eventos.



La figura 42 muestra las interfaces del gestor de contexto *SituationAvailableContextManager* y la *SituationWithinRange*ContextManager

Figura 42: Situación de la Aplicación, datos de medición y el componente de gestión del contexto



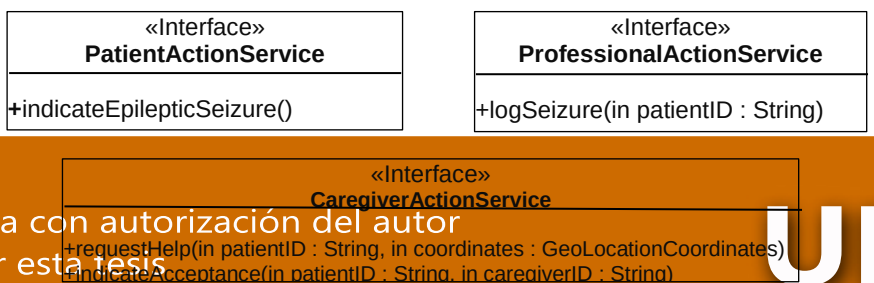
Servicios de Acción

Los componentes específicos de la aplicación que se ejecutan en los dispositivos de los usuarios deben implementar servicios de acción con el fin de recibir las invocaciones de la plataforma que gestiona el contexto. Las invocaciones, a su vez, son respuestas a determinadas ocurrencias de eventos y evaluaciones de condición. Tres tipos de servicios de acción deben ser ofrecidos, de la siguiente manera:

- *PatientActionService* (AS_1): Servicio de acción ofrecido por la acción específica de la aplicación del paciente. La aplicación Implementa la operación *indicateEpilepticSeizure*, que recibe la notificación de un posible ataque epiléptico.
- *ProfessionalActionService* (AS_2): servicio de acción ofrecido por el componente específico de la aplicación de un profesional de la salud. Implementa la operación *logSeizure*, que recibe la notificación de una alarma epiléptico y lo registra.
- *CaregiverActionService* (AS_3): servicio de acción ofrecido por el componente específico de la aplicación de un cuidador. Implementa *requestHelp* y operaciones *indicateAcceptance*, que recibe las solicitudes para ayudar a un paciente en particular, y notifica a la aceptación de un cierto cuidador.

La figura 43 ilustra la especificación de las interfaces ofrecidas por estos componentes del servicio de acción.

Figura 43: Especificación de los servicios de acción a ser ofrecido por los componentes específicos de la aplicación



Control de los Servicios

Con el fin de delegar los componentes necesarios de comportamiento reactivo a la plataforma, hay que especificar reglas ECA-DL que representan estos comportamientos. Los servicios de control y el lenguaje de la ECA-DL han sido ampliamente

1. *SeizureAlarm*: sobre una alarma ataque epiléptico, la aplicación que se ejecuta en el dispositivo del paciente debe ser notificado de la posibilidad de un ataque epiléptico y que el paciente está actualmente llevando a cabo una actividad peligrosa. la aplicación de este paciente puede lanzar un mensaje de voz advirtiendo al paciente que deje de dicha actividad en los próximos minutos.
2. *HelpRequestNotification*: alarma de ataque epiléptico, un conjunto de cuidadores debe ser notificado de que un paciente está en necesidad de ayuda. Solamente los cuidadores que están disponibles y cerca del paciente debe recibir la notificación. Al recibir dicha notificación, la aplicación que se ejecuta en el dispositivo del cuidador puede trazar en un mapa una posible ruta entre el cuidador y el paciente.
3. *HelpAcceptedNotificafion*: después de recibir una aceptación por parte de un cuidador en particular, a los médicos que habían recibido previamente la notificación de la ayuda debe ser notificado de que uno de los cuidadores ya ha aceptado ayudar a ese paciente en particular.
4. *LogSeizureAlarm*: Tras una alarma de ataque epiléptico, ingrese la alarma la solicitud del profesional de la salud para su posterior análisis.

Componente de resolución de acción

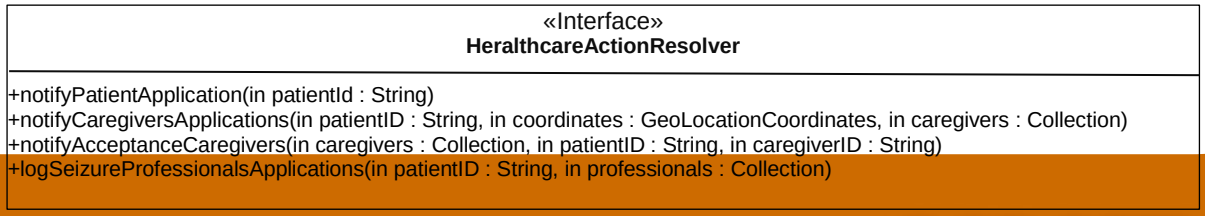
Un componente de resolución de la acción recibe invocaciones de un componente controlador destinado a despachar una serie de acciones. Hemos definido resolver la actuación médica que implementa todas las acciones posibles que se puede denominar de reglas ECA-XDL que se ejecutan en el componente controlador. Cuando se producen eventos y condiciones se hacen coincidir, invoca el controlador de acciones en la resolución de la actuación médica, que a su vez puede despachar las invocaciones apropiadas para los servicios de acción adecuados, que se implementan en los respectivos componentes específicos de la aplicación. Una sola invocación de una acción implementada en el sistema de resolución puede generar varias invocaciones de los servicios de acción que se ejecutan en los componentes de la aplicación de los usuarios.

Las siguientes acciones deben ser implementadas por el componente de resolución de la actuación médica a fin de realizar las conductas antes mencionadas:

- *notifyPatientApplication* se requiere para realizar SeizureAlarm comportamiento reactivo, que tiene por objeto una notificación de aplicación adecuada del paciente de su posible alarma de ataque. Esta acción recibe como argumento el identificador único del paciente, con el que es posible adquirir la dirección del extremo de servicio de la aplicación en el dispositivo del paciente. Cuando se adquiere la dirección del paciente, la operación *indicateEpilepticSeizure* ofrecido por el *PatientActionService* se invoca.
- *notifyCaregiversApplications* se requiere para realizar HelpRequestNotification comportamiento reactivo, que tiene por objeto notificar cuidadosamente a los cuidadores a través de las aplicaciones. Los argumentos aceptados por esta acción son: la localización del paciente, la recolección de cuidadores que necesitan ser notificado. Para cada uno de estos cuidadores, la operación *requestHelp* ofrecido por el *CaregiverActionService* se invoca.
- *notifyAcceptanceCaregivers* se requiere para realizar *HelpAcceptedNotification* comportamiento reactivo, que tiene por objeto notificar adecuadamente a las aplicaciones de los cuidadores. Esta acción recibe como argumento como matriz a los cuidadores que necesitan ser notificado, el paciente que necesita ayuda; y el cuidador que aceptó ayudar. Para cada uno de los cuidadores que necesitan ser notificados, la operación *indicateAcceptance* ofrecido por el *CaregiverActionService* se invoca.
- *logEpilepticAlarm* se requiere para realizar *LogSeizureAlarm* comportamiento reactivo, que tiene por objeto notificar a las aplicaciones de los médicos que se ha producido una alarma epiléptica. Esta acción recibe como argumento el identificador único del paciente que padece la convulsión, y una colección con los identificadores de los profesionales médicos. La operación *logSeizure* ofrecido por el *ProfessinalActionService* es invocada.

La figura 44. Ilustra la especificación de la interfaz de resolver la actuación médica que despacha las invocaciones requerida para las implementaciones. Estas implementaciones se distribuyen y se ejecutan en los dispositivos de los usuarios.

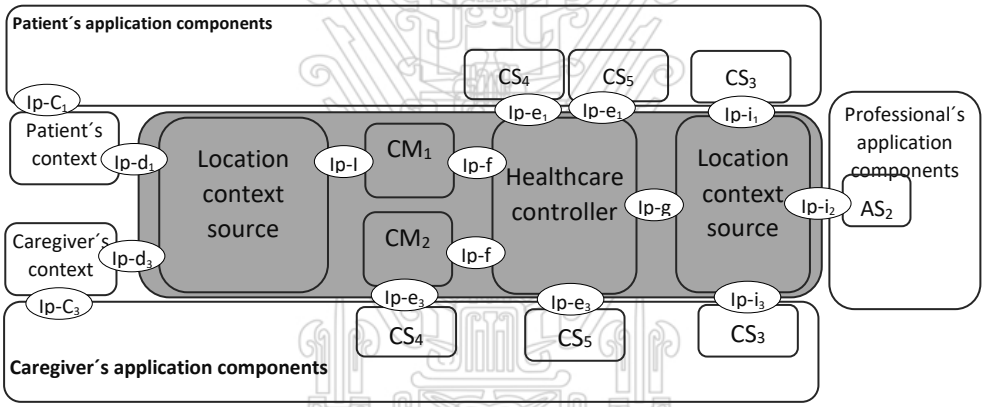
Figura 44: Especificación de los servicios de acción para la aplicación



Configuración de Componentes

Figura 45 representa la configuración completa de los componentes que son necesarios para realizar la aplicación sensible al contexto. Al incorporar el componente de asistencia sanitaria al controlador, el componente de resolución de la actuación, y los componentes de servicio *específicos de la aplicación*. Tales tipos de mejoras han sido ampliamente discutidas en el capítulo 4. El área gris indica la parte de manejo de plataforma de contexto. Como ya se ha comentado, las partes específicas de la aplicación consisten en funciones específicas que no deben ser generalizadas en la plataforma. Estos incluyen varias fuentes de contexto y los servicios de acción específicos de la aplicación. Componentes gestores de contexto y la fuente de contexto de ubicación se implementan como parte de la plataforma.

Figura 45: Configuración de los Componentes para la Aplicación



Se ha propuesto un único componente controlador, esto podemos evidenciar en los comportamientos reactivos de la aplicación. Sin embargo, también sería posible utilizar varias instancias del componente controlador, cada uno conteniendo reglas específicas ECA-DL. Puesto que las reglas ECA-DL son independientes unos de otros, que pueden ser distribuidos arbitrariamente a través de diferentes instancias de componentes del controlador.

Reglas ECA-DL

Cada una de las reglas ECA-XDL se analizan a continuación, tiene por objeto especificar el comportamiento requerido por la aplicación. Estas reglas se ofrecen para el componente controlador durante la ejecución de la plataforma, es decir, en el tiempo de funcionamiento de componentes del controlador. Suponemos para el escenario de aplicación de la asistencia sanitaria que las reglas ECA-XDL que aquí se presentan se definen en la aplicación en tiempo de diseño y en tiempo de ejecución de la plataforma.

Sin embargo, también es posible que las reglas de ECA-XDL se definen en tiempo de ejecución de la aplicación.

Por ejemplo, el profesional de la salud podría estar provisto de una interfaz de usuario que le permita definir comportamientos reactivos deseados, lo que podría traducirse a la ECA-XDL por algún componente específico que se ejecuta en el dispositivo del profesional médico. En este caso, las reglas ECA-XDL se proporcionarían al componente controlador de componentes específicos de la aplicación. Hemos hablado de estas configuraciones alternativas anteriormente. Además, otras normas, distintas a las que aquí se presentan, también podrían ser especificadas. El único requisito es que los valores del contexto, situación, eventos y acciones contempladas por las normas deben estar disponibles para el componente controlador.

La siguiente regla ECA-XDL (*ECARule1*) tiene como objetivo la realización de la *SeizureAlarm* comportamiento reactivo. Se utiliza una cláusula de alcance para definir que esta regla se debe aplicar a todos los pacientes epilépticos registrados en el componente de controlador. La cláusula chequea si este paciente está realizando actualmente alguna actividad potencialmente peligrosa. Cuando el evento ocurre y la condición de la cláusula es verdadera, la acción *notifyPatientApplication* se invoca.

```
Scope (EpilepticPatient.*; p)
{
        Upon EpilepticAlarm (p)
        When p.hasHazardousActivity.hazardousvalue
        Do NotifyPatientApplication (p)
}
```

La siguiente regla ECA-XDL (*ECARule2*) tiene como objetivo la realización del comportamiento reactivo [*HelpRequestNotification]*. Esta regla utiliza la misma cláusula de alcance que la regla anterior. Al recibir la notificación de eventos de la alarma para cualquier paciente epiléptico, la acción *NotifyCaregiversApplications* se invoca. La cláusula select selecciona todos los médicos del paciente que están dentro del alcance y disponibles. La colección de resultados de esta selección se pasa como uno de los argumentos de la acción.

```
Scope (EpilepticPatient.*; p){
        Upon EpilepticAlarm (p)
        Do NotifyCaregiversApplications ( p, p.hasGeoLocation.coordinates,
                Select (Caregiver.*; care; isCaregiverOf (care, p) and SituationWithinRange
                (p, care) and SituationCaregiverAvailable (care)))
```

}

La siguiente regla ECA-XDL (*ECARule3*) tiene como objetivo la realización del comportamiento reactivo de la acción *HelpAcceptedNotification*. Esta regla utiliza la misma cláusula de alcance que las reglas anteriores. La cláusula especifica una composición de evento, en el que Ev1 (*EpilepticAlarm (p))* debe ocurrir seguido de Ev2 (*AcceptRequest (p)*). Esto significa que este evento compuesto sucede cada vez que hay una alarma de ataque epiléptico, seguido de una aceptación para ayudar al paciente. Como se muestra en la Figura 23, la acción *AcceptRequest* acepta dos argumentos, un cuidador y un paciente. En este ejemplo, sólo un paciente se proporciona como argumento. De esta manera, podemos filtrar las notificaciones de eventos de la acción *AcceptRequest* para el paciente p, de cualquier cuidador. Cuando se produzca este evento compuesto, la acción *notifyAcceptanceCaregivers* se invoca. La cláusula select selecciona todos los médicos del paciente que están dentro del alcance, y son diferentes del cuidador que aceptó la solicitud. La identificación del cuidador que aceptó la solicitud se obtiene mediante el acceso a uno de los atributos del evento, es decir *Ev2.caregiverID*. La colección de resultados de esta selección se hace pasar como uno de los argumentos para la acción.

```
Scope (EpilepticPatient.*; p){
        Upon Ev1: EpilepticAlarm (p); Ev2: AcceptHelpRequest (p)
        Do notifyAcceptanceCaregivers (
                Select (CareGiver.*; care; isCareGiverOf (care, p) and SituationWithinRange (care, p) and
                        care <> Ev2.caregiverID), p, Ev2.caregiverID)
}
```

La siguiente regla ECA-XDL (*ECARule4*) tiene como objetivo la realización del comportamiento reactivo *LogSeizureAlarm*. Al recibir una notificación de alarma epiléptica para cualquier paciente epiléptico, la acción *logEpilepticAlarm* se invoca. La cláusula select selecciona todos los profesionales de la salud del paciente.

```
Scope (EpilepticPatient.*; p) {
        Upon EpilepticAlarm (p)
        Do logEpilepticAlarm (p, Select (HealthProfessional.*; prof; isHealthProfessionalOf (prof, p)))
}
```

Despliegue

La especificación de detección de la situación debe ser desplegada en los gestores de contexto, y las reglas ECA-XDL en el componente controlador. La implementación de reglas ECA-XDL consiste en someter a una representación textual de la norma con el componente controlador. Para la especificación de la situación, sólo con las reglas Jess y código Java se pueden implementar en un componente de gestor de contexto. .

Gestión de Políticas Para el Conocimiento del Contexto

En general, la administración de políticas en una aplicación orientada al contexto se refiere al control del acceso a la información del contexto, la imposición de la privacidad del usuario y la gestión de relaciones de confianza entre los usuarios finales y los componentes.

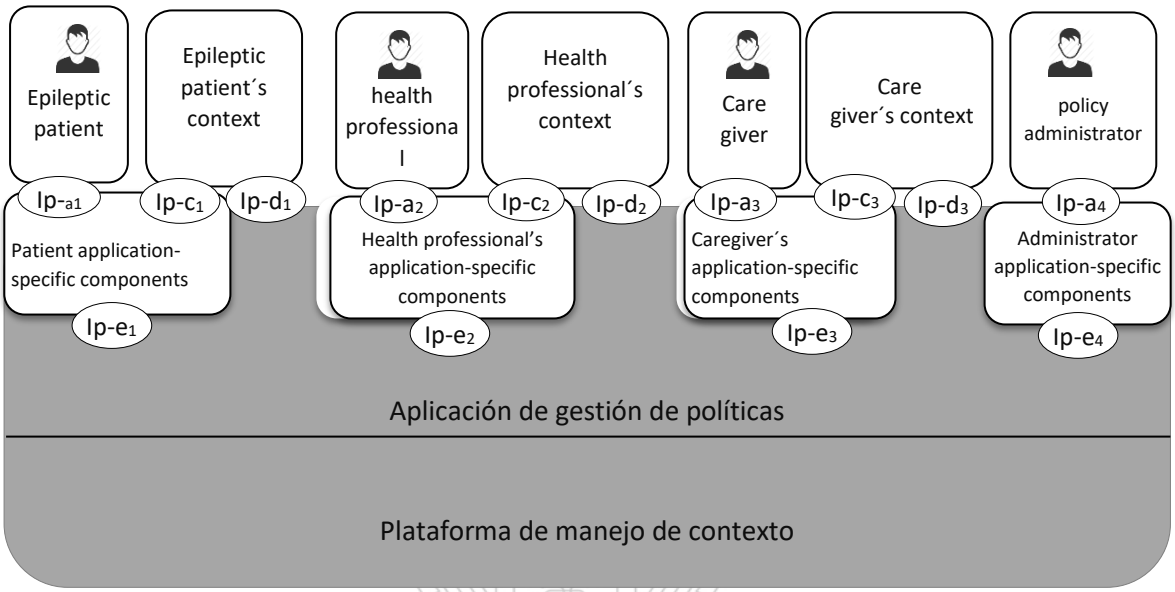
Las aplicaciones y plataformas orientadas al contexto pueden manejar potencialmente miles de entidades (por ejemplo, usuarios finales, proveedores de servicios, proveedores de contexto), lo que dificulta la gestión de las políticas. En tales casos, es deseable una herramienta de gestión de políticas que asista y automatice la gestión de las políticas. Sin embargo, las herramientas estándar de administración de políticas no ofrecen soporte para la administración de directivas contextuales, que permitan definir políticas basadas en la información del contexto de los usuarios. Estas herramientas suelen ofrecer soporte para la definición de políticas estáticas, en las que las entidades para las cuales se aplican estas políticas se conocen de antemano.

Por el contrario, las aplicaciones y plataformas sensibles al contexto requieren una gestión dinámica de políticas, en la que puedan ser creadas y destruidas cuando las entidades entran o salen del contexto. Para abordar este tema, proponemos aquí un mecanismo dinámico de gestión de políticas que define políticas basadas en la información de contexto de los usuarios. Este mecanismo se realiza mediante nuestro marco de detección de situaciones y los servicios de control. Para este escenario en particular, reutilizamos los modelos de contexto y situación ya presentados en la figura 24 y los modelos de información de contexto y situación presentados en la figura 32.

Diseño estructural de la aplicación de gestión de políticas

La Figura 46 muestra cómo la plataforma de manejo de contexto ofrece soporte a la aplicación de administración de políticas. Esta aplicación utiliza servicios genéricos ofrecidos por la plataforma y también implementa servicios específicos (componentes específicos de aplicaciones). La Figura 46 amplía a la Figura 45 incluyendo un usuario de administrador de políticas y sus componentes específicos de aplicación del administrador. Por medio de puntos de interacción de tipo ip-a_4, el administrador define las políticas de interés para esta aplicación, además de información de configuración tal como comandos de inicio. Los componentes específicos del administrador de políticas asignan los comportamientos recopilados del administrador a las reglas ECA-DL, que se ofrecen a la plataforma del controlador a través de los puntos de interacción de ip-e_4. Además, los puntos de interacción del tipo ip-e_4 permiten a la aplicación del administrador reunir información para gestionar actividades.

Figura 46: Visión general del diseño estructural de la aplicación de gestión de políticas de contexto

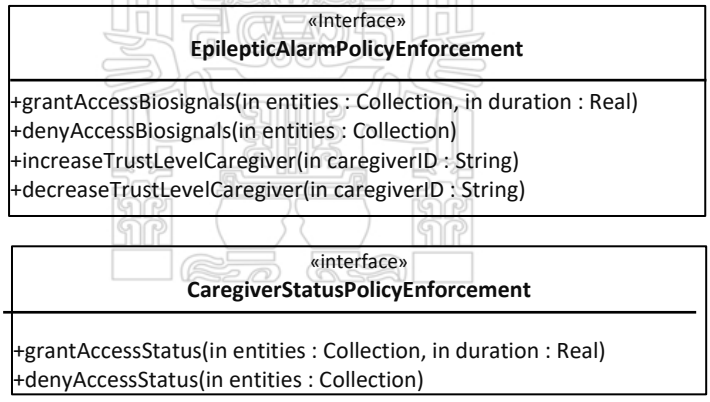


Componentes de aplicación de políticas

En la aplicación de administración de políticas, los componentes de origen de contexto también implementan interfaces de implementación de políticas. Estas interfaces permiten configurar fuentes de contexto para cumplir con requisitos de políticas particulares, que dependen del contexto y la situación.

La Figura 47 muestra las interfaces de implementación de políticas que deben implementarse en el contexto.

Figura 47: implementación de políticas por componentes



La operación grantAccessBiosignals otorga acceso a la información del paciente a una colección de entidades (cuidadores y profesionales de la salud), que se proporciona como argumento a esta operación. La operación denyAccessBiosignals configura el

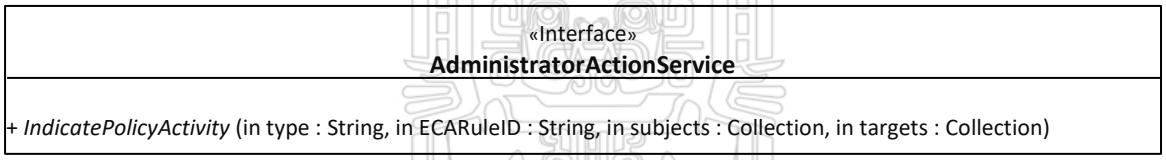
EpilepticAlarmContextSource para denegar el acceso a la información del paciente a una colección particular de entidades, que se pasa como argumento a esta operación.

Con respecto a la interfaz CaregiverStatusPolicyEnforcement, definimos dos operaciones, a saber, grantAccessStatus y denyAccessStatus. La operación grantAccessStatus concede acceso a la información del estado de este proveedor en particular a la colección de entidades, durante un cierto período de tiempo. Del mismo modo, la operación denyAccessStatus niega el acceso a la información del estado del cuidador a una determinada colección de entidades.

Servicio de acción del administrador de políticas

Los componentes específicos de la aplicación del administrador implementan la funcionalidad que permite al administrador tener una vista global de las políticas que se aplican y de las entidades a las que se refieren. Por lo tanto, los componentes específicos de la aplicación del administrador deben estar informados de todas las políticas que se aplican en la plataforma. Para realizar esto, definimos un componente *AdministratorActionService* (AS_4) que recibe invocaciones de la plataforma de manejo de contexto. Este componente de servicio de acción ofrece operaciones para informar a los componentes específicos de la aplicación del administrador de las actividades relacionadas con la política. La Figura 48 muestra la interfaz implementada por el componente *AdministratorActionService*.

Figura 48: Interfaz de servicio del administrador



La operación *indicatePolicyActivity* informa el tipo de política (por ejemplo, el acceso a la información del paciente). Los sujetos y los objetivos son entidades que desempeñan diferentes funciones dependiendo del tipo de política. Por ejemplo, en una política de control de acceso, un sujeto es la entidad que se concede o se deniega acceso a la información sobre un destino. En las políticas de gestión, los sujetos se refieren a las entidades que mantienen valores de confianza sobre objetivos. En nuestro ejemplo, un paciente que mantiene valores de confianza sobre los cuidadores es el sujeto, mientras que los cuidadores son los objetivos.

Servicios de control

Con el fin de delegar en la plataforma las piezas requeridas de comportamiento reactivo, las reglas ECA-XDL que representan estos comportamientos deben ser definidas en primer lugar. Los siguientes comportamientos reactivos son requeridos por la aplicación de administración de políticas.

- *GrantAccess*: en el caso de la alarma epiléptica del paciente, los cuidadores que están disponibles y cercanos deben tener derecho a acceder a los bioseñales del paciente. Además, la información sobre el estatus de los cuidadores que están cerca debe estar disponible para el paciente durante 30 minutos.
- *DenyAccess*: a la notificación de aceptación de un cuidador, los otros cuidadores que habían recibido previamente la solicitud de ayuda deben tener los derechos de acceso a los biosignales del paciente.
- *IncreaseTrust*: en la notificación de aceptación de un cuidador, la solicitud del paciente debe aumentar su valor de confianza en el cuidador que aceptó la solicitud de ayuda.
- *DecreaseTrust*: a la notificación de rechazo de un médico, la solicitud del paciente debe disminuir su valor de confianza en el cuidador que rechazó una solicitud de ayuda.

Por razones de gestión, el administrador debe tener una vista global de las actividades de gestión de políticas que se están produciendo dentro de la plataforma. Con el fin de informar a los componentes específicos de la aplicación del administrador de tales actividades, cada vez que se implementa o se elimina una política, se debe invocar el servicio de acción del administrador de directivas.

Componente de resolución de acciones de gestión de políticas

A diferencia de la aplicación anterior en la que el componente *resolver* de acciones de atención sólo invoca servicios de acción en partes específicas de la aplicación, en la aplicación de gestión de políticas se invocan fuentes de contexto (CS_3 y CS_4) como resultado de la ejecución de reglas ECA-XDL. Las acciones siguientes son implementadas por el componente el resolver de acciones de gestión de políticas para realizar los comportamientos antes mencionados.

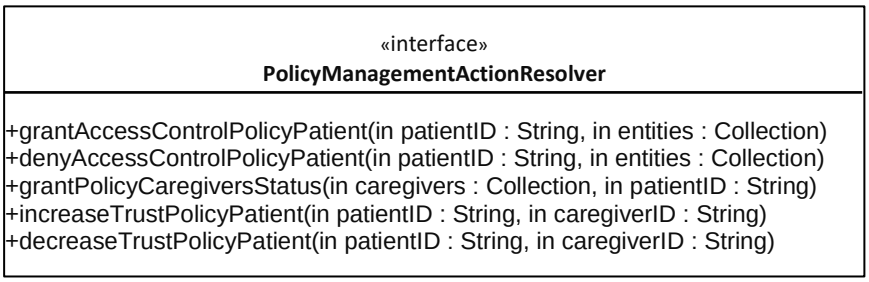
- *GrantAccessControlPolicyPatient*: se utiliza para realizar el comportamiento reactivo *GrantAccess* con el fin de implementar políticas de control de acceso sobre los bioseñales del paciente. Recibe como argumentos el paciente a configurar, y la recolección de entidades para las cuales se deben conceder bio-elementos. Invoca la

operación *grantAccessBiosignals* en el *EpilepticAlarmContextSource* que se ejecuta en el dispositivo del paciente.

- *DenyAccessControlPolicyPatient*: se utiliza en el comportamiento reactivo *DenyAccess* para eliminar las políticas de control de acceso sobre las bio-señales del paciente. Recibe como argumentos el paciente a configurar, y la recolección de entidades para las cuales se debe denegar los bio-elementos. Invoca la operación *denyAcccessBiosignals* en el *EpilepticAlarmContextSource* que se ejecuta en el dispositivo del paciente.
- *GrantPolicyCaregiversStatus*: se utiliza para realizar el comportamiento reactivo *GrantAccess* con el fin de implementar políticas de control de acceso sobre la información de estado del cuidador. Recibe como argumentos una colección de cuidadores a configurar, y al paciente al que se debe otorgar la información sobre el estado. Para cada cuidador, se invoca la operación *grantAccessStatus* en el *CaregiverStatusAvailabilityContextSource* que se ejecuta en los dispositivos de los cuidadores.
- *IncreaseTrustPolicyPatient*: se utiliza para realizar el comportamiento reactivo *IncreaseTrust* con el fin de implementar las políticas de gestión de confianza sobre los valores de confianza del cuidador. Recibe como argumentos un paciente a configurar, y el cuidador para el cual se debe aumentar el nivel de confianza. Invoca la operación *increaseTrustLevelCaregiver* en el *EpilepticAlarmContextSource* que se ejecuta en el dispositivo del paciente.
- *DecreaseTrustPolicyPatient*: se utiliza para realizar el comportamiento reactivo *DecreaseTrust* con el fin de implementar las políticas de gestión de confianza sobre los valores de la confianza del cuidador. Recibe como argumentos un paciente a configurar, y el cuidador para el cual se debe disminuir el nivel de confianza. Invoca la operación *decreaseTrustLevelCaregiver* en el *EpilepticAlarmContextSource* que se ejecuta en el dispositivo del paciente.

Cada una de estas acciones invoca a *indicatePolicyActivity* en el servicio de acción que se ejecuta en el dispositivo para informar al administrador de la actividad de la política en particular que se está llevando a cabo actualmente. La Figura 45 muestra la especificación de la interfaz del componente de resolución de acciones para la aplicación de administración de políticas.

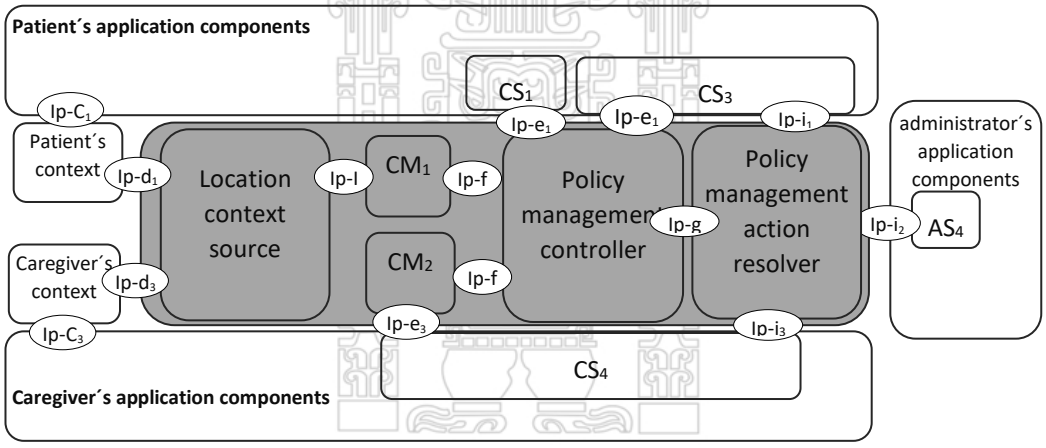
Figura 49: Especificación de la interfaz de resolución de acciones para la aplicación de administración de políticas



Configuración de Componentes

La Figura 45 muestra la configuración completa de los componentes que son necesarios para realizar la aplicación de administración de políticas. Refina la arquitectura presentada en la Figura 41, que incorpora el componente del controlador de administración de políticas, el componente de resolución de acciones de administración de políticas y los componentes de servicio de acción específicos de la aplicación. Las invocaciones a la aplicación del administrador se realizan por medio de puntos de interacción del tipo ip-i_2. La fuente de contexto CS_5 se ha omitido en la Figura 45 para mayor claridad

Figura 50: Configuración de componentes para la aplicación de administración de políticas



ECA-XDL

Cada una de las siguientes reglas ECA-XDL tiene como objetivo especificar un comportamiento particular requerido por la aplicación de administración de políticas. Estas reglas se ofrecen al controlador de administración de políticas mediante la aplicación del administrador en tiempo de ejecución de la plataforma. El punto de interacción que permite que la aplicación del administrador proporcione reglas de ECA-XDL a la plataforma no se muestra en la Figura 51 en aras de la claridad.

La siguiente regla ECA-XDL (ECARule5) tiene como objetivo realizar el comportamiento reactivo *GrantAccess*. Al recibir notificación del evento de una alarma epiléptica para cualquier paciente epiléptico, se invocan las acciones *grantAccessControlPolicyPatient* y *grantPolicyCaregiversStatus*.

```
Scope (EpilepticPatient.*; p)
{
    Upon EpilepticAlarm (p)
    Do grantAccessControlPolicyPatient (p,
            Select (Caregiver.*; care; isCaregiverOf (care, p) and SituationWithinRange (p,
        care) and SituationCaregiverAvailable (care)));
        grantPolicyCaregiversStatus (Select (Caregiver.*; care; isCaregiverOf (care, p) and
            SituationWithinRange (p, care) and SituationCaregiverAvailable (care)), p)
    }
```

La siguiente regla ECA-XDL (ECARule6) tiene como objetivo realizar el comportamiento reactivo *DenyAccess*. La cláusula define que una alarma epiléptica de convulsiones debe ser seguida por la aceptación de un cuidador para ayudar al paciente que tiene la convulsión. Al producirse este evento compuesto, se invoca la acción *denyAccessControlPolicyPatient*. Esta operación niega el control de acceso a todos los cuidadores a los que se les concedió previamente acceso a las bio-señales del paciente, excepto para el cuidador que aceptó la solicitud de ayuda.

```
Scope (EpilepticPatient.*; p)
{
    Upon Ev1: EpilepticAlarm (p); Ev2: AcceptHelpRequest (p)
            Do denyAccessControlPolicyPatient (p,
            Select (CareGiver.*; care; isCareGiverOf (care, p) and SituationWithinRange (care,
            p) and SituationCaregiverAvailable (care)
        and care <> Ev2.caregiverID), p, Ev2.caregiverID));
}
```

La siguiente regla ECA-XDL (ECARule7) tiene como objetivo realizar el comportamiento reactivo *IncreaseTrust*. Sobre la cláusula se define el mismo evento compuesto que ECARule6. Al producirse este evento compuesto, se invoca la acción *increaseTrustPolicyPatient*, que aumenta el valor de confianza de un cuidador en particular.

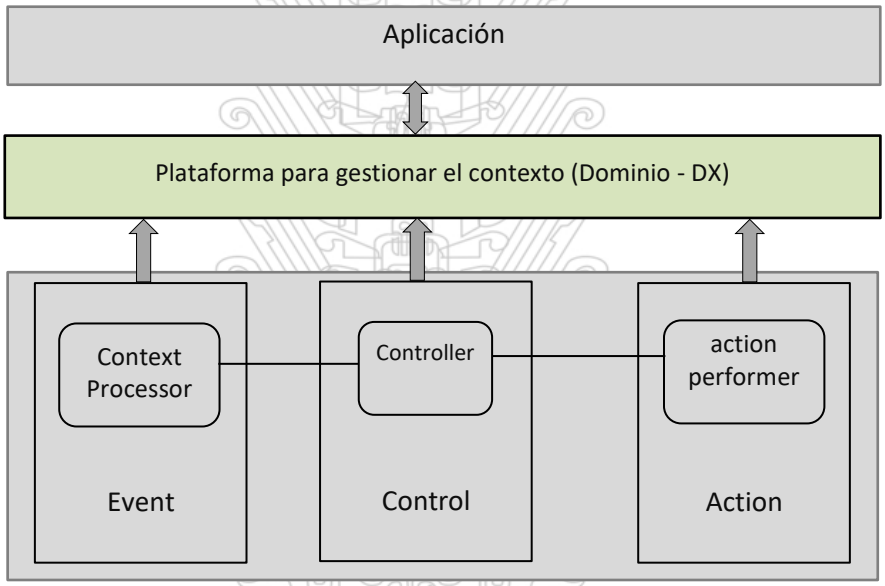
```
Scope (EpilepticPatient.*; p)
{
    Upon Ev1: EpilepticAlarm (p); Ev2: AcceptHelpRequest (p)
            Do increaseTrustPolicyPatient (p, Ev2.caregiverID)
}
```

La siguiente regla ECA-XDL (ECARule8) tiene como objetivo realizar el comportamiento reactivo *DecreaseTrust*. El alcance de la cláusula aplica a todos los

cuidadores del sistema. Al recibir una notificación del evento *RejectHelpRequest*, se invoca la acción *decreaseTrustPolicyPatient*. La identificación del paciente se recupera como uno de los parámetros de la notificación de evento *RejectHelpRequest*.

```
Scope (Caregiver.*; c)
{
    Upon Ev:RejectHelpRequest (c)
    Do decreaseTrustPolicyPatient (Ev.patientID, c)
}
```

Figura 51: Arquitectura de Software Para el Desarrollo de Aplicaciones Sensibles al Contexto - ECA-DX-Modelo Propuesto



Estrategia de Prueba de Hipótesis

Variables

Arquitectura de Software

Aplicaciones sensibles al contexto

Población

En primer lugar, se determinó el marco muestral y la unidad de análisis para luego proceder limitar la población. Para el cual como universo tenemos al personal que trabaja como desarrolladores de software en la Universidad Nacional del Santa - UNS y la Universidad Católica Los Ángeles de Chimbote - ULADECH, estas universidades se encuentran geográficamente ubicadas en la región de Ancash.

Tabla 4: Relación de expertos que validaron el instrumento UNS

UNVIERSIDAD NACIONAL DEL SANTA

DIVISION DE SISTEMAS

ITEM	APELLIDOS Y NOMBRES	CARGO

Tabla 5: Relación de expertos que validaron el instrumento ULADECH

UNVIERSIDAD CATOLICA LOS ANGELES DE CHIMBOTE

DIVISION DE SISTEMAS

ITEM	APELLIDOS Y NOMBRES	CARGO
1		
2		
3		

4
5
6
7
8
9
10
11
12
13
14
15
16

Muestra

En base al total del personal de desarrollo de software de las dos universidades consignadas en la población, suma un total de 21 expertos. Considerando que la población es menor a 50 individuos, la muestra es igual a la población(Castro, 2003). Por lo que la cantidad mínima representativa para validar el estudio será de 21 expertos.

Técnicas de Investigación

Entre las principales técnicas de investigación utilizadas en la presente investigación, se consideró la encuesta, que permitió validar los resultados arrojados de la aplicación prototipo, dicho instrumento ha sido desarrollado a partir de las variables e indicadores de acuerdo a la investigación.

Instrumentos de recolección de datos

Considerando las características de la investigación se ha utilizado la encuesta personal basado en un cuestionario de preguntas cerradas, para poder cumplir con la muestra.

Tabla 6: Juicio de Experto para validar Propuesta Tecnológica.

JUICIO DE EXPERTO PARA VALIDAR LA PROPUESTA TECNOLOGICA
"ARQUITECTURA DE SOFTWARE PARA DESARROLLO DE APLICACIONES SENSIBLE AL CONTEXTO"

INSTRUCCIONES:

Coloque en cada casilla un aspa(X) correspondiente a la pregunta, según los criterios que a continuación se detallan. Las dimensiones a evaluar son: arquitectura, soporte, diseño, servicios, aplicaciones sensibles al contexto, contexto e interacción.

Dominio y Preguntas	Respuestas			Observación o recomendación
	3 bueno	2 regular	1 deficiente	
ARQUITECTURA				
1.-				
2.-				
3.-				
4.-				
5.-				
6.-				
7.-				
SOPORTE				
1.-				
2.-				
3.-				
4.-				
5.-				
6.-				
7.-				

INTERACCIÓN

1.-

2.-

3.-

4.-

5.-

6.-

7.-

8.-

9.-

10.-

11.-

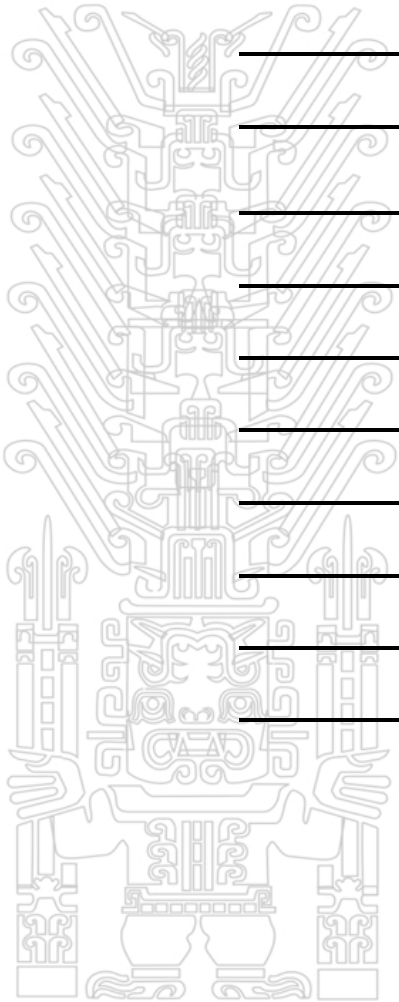
12.-

13.-

14.-

15.-

16.-



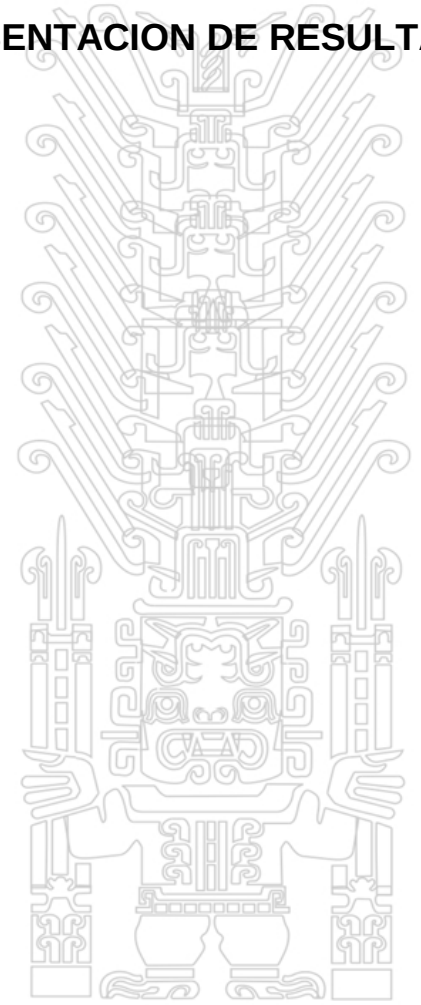
Procesamiento y análisis de datos

Para el procesamiento de datos se utilizó el software SPSS v22, donde se registraron las encuestas aplicadas, bajo un formato establecido para obtener la información en base al cuestionario.

Para el análisis y prueba de hipótesis, se emplea la técnica de análisis multivariantes, con el apoyo del software SPSS v22.

CAPITULO IV

PRESENTACION DE RESULTADOS



Prototipo de la Aplicación

Se ha construido un prototipo que implementa la aplicación de atención médica. Los Objetivos del prototipo son:

- Para demostrar la factibilidad de nuestro enfoque. El prototipo sirve para demostrar que las abstracciones propuestas en esta tesis (contexto y modelado de la situación) y la plataforma de manejo del contexto pueden ser construidas en un sistema de computación.
- Permitir la evaluación del desempeño y la escalabilidad del enfoque propuesto.

Sistema de Configuración

El prototipo implementa un producto de diseño que incluye: código Java que implementa el contexto y el modelo de situación y las reglas de Jess para la detección de situaciones y la arquitectura presentada en la Figura 41 implementa los tipos de datos de medición de información del contexto, y la especificación de las interfaces de los componentes.

Hemos implementado los componentes utilizando Java como el lenguaje de programación, Java RMI, para permitir la distribución de los componentes, y Eclipse (Oracle, 2014) como el entorno de desarrollo. Se ha implementado en el prototipo todos los componentes que se identificaron en el trabajo de diseño, las fuentes de contexto, los gestores de contexto, el controlador, el resolver de acciones y los servicios de acción (ver arquitectura en la Figura 40). Los componentes específicos de la aplicación que no son ni fuentes de contexto ni servicios de acción no se han implementado, es decir, solo se ha implementado interfaces de usuario final y otras funciones específicas del usuario.

Fuentes de contexto

Se ha simulado la producción de información de contexto y eventos. El GeoLocationContextSource genera aleatoriamente coordenadas de ubicación geográfica para sus usuarios, pero respetando una variación máxima de los cambios de ubicación. El EpilepticAlarmContextSource y AcceptRequestContextSource también generan aleatoriamente eventos EpilepticAlarm y AcceptRequest, respectivamente. De forma similar, CaregiverStatusContextSource y HazardousActivityContextSource cambian aleatoriamente el estado de disponibilidad de los cuidadores y si el usuario está realizando o no una actividad potencialmente peligrosa. Las fuentes de contexto no implementan los modelos de contexto propuestos, ya que generan valores aislados simples, que aún no están combinados con otros valores de contexto. Se asume, sin embargo, que los usuarios en el sistema (cuidadores, pacientes y otros profesionales de la salud) son identificados de manera única por un identificador, que se comparte entre los componentes.

Gestores de Contexto

Los gestores de contexto detectan las situaciones SituationCaregiverWithinRange y SituationCaregiverAvailable se han implementado en la especificación de la situación a la realización de la misma. El SituationCaregiverWithinRangeContextManager reúne información de contexto del GeoLocationContextSource a través de una interfaz de devolución de llamada que permite al GeoLocationContextSource enviar información de ubicación al componente SituationCaregiverWithinRangeContextManager siempre que una nueva ubicación genere valor. Cuando se recibe información de ubicación desde el origen del contexto, SituationCaregiverWithinRangeContextManager actualiza objetos Java locales que se sombrean automáticamente en la memoria de trabajo de Jess. Los valores de ubicación se intercambian usando el tipo de datos GeoLocationMeasurement representado en la Figura 31.

El SituationCaregiverWithinRangeContextManager detecta cuando los pacientes y los cuidadores están cerca unos de otros por medio de las reglas de Jess que se ejecutan en el motor local de Jess. Con el fin de detectar este tipo de situación, este gestor de contexto implementa parte del modelo, de acuerdo con la especificación del SituationCareGiverWithinRange, representado en la Figura 30.

Del mismo modo, el componente SituationCaregiverAvailableContextManager implementa el modelo de situación. Con el fin de razonar sobre la disponibilidad de los cuidadores, que reúne la información del contexto de la CaregiverStatusContextSource a través de una interfaz de devolución de llamada, que permite al CaregiverStatusContextSource enviar la información de estado del cuidador al SituationCaregiverAvailableContextManager siempre cuando el estado de un determinado cuidador no haya cambiado. La información del estado del cuidador se intercambia siguiendo el tipo de datos CaregiverStatusMeasurement descrito en la Figura 30. El componente SituationCaregiverAvailableContextManager ejecuta un motor local de Jess, que detecta cuando la situación comienza y deja de mantenerse.

Controlador

El componente de controlador de salud recopila información de las fuentes del contexto y administradores para recopilar notificaciones de eventos y valores de condición. Dado que el controlador integra la información de contexto y de la situación, el controlador implementa el contexto completo y los modelos de situación, como se presenta en la Figura 30, 31 y 32. Hemos implementado interfaces de devolución de llamada en el controlador, lo que permite que las fuentes de contexto y los gestores se encargan de impulsar el contexto y la situación de la información en el controlador. Cuando se invocan

estos métodos de devolución de llamada, la información del contexto y de la situación es recibida por el controlador e incluida en la memoria de trabajo local de Jess. Las reglas de la ECA-XDL están funcionando continuamente en el motor local de Jess.

El controlador implementa las estructuras de eventos necesarias para realizar el consumo y la composición del mismo, como se presenta anteriormente. De acuerdo con las reglas ECA-XDL definidas para la aplicación de atención médica, se esperan dos tipos de notificaciones de eventos: EpilepticAlarm y AcceptRequest. Todas las notificaciones provenientes de las fuentes de contexto deben registrarse con el controlador. Dado que las normas de la ECA-XDL están delimitadas, existen registros de eventos EpilepticAlarm para cada paciente epiléptico y para cada regla que se refiere a esto evento. Supongamos, por ejemplo, que sólo el Sr Pedro está actualmente como un paciente epiléptico, y sus cuidadores son María y Alicia. El controlador mantiene los registros de eventos, como se muestra en la figura 46. De esta manera, cuando se recibe una notificación de un evento de alarma epiléptica, como por ejemplo EpilepticAlarm (Sr Pedro), el controlador compara esta notificación con las inscripciones disponibles. Dado que hay cuatro coincidencias, el controlador incluye cuatro instancias del evento EpilepticAlarm en la memoria de trabajo, una para cada regla que se refiere a ese evento. Similarmente, cuando Se reciben notificaciones de eventos de AcceptRequest (Sr Pedro, cuidador), el controlador encuentra dos coincidencias, una para cada cuidador, y crea dos instancias de este evento en la memoria de trabajo.

En total, nuestro prototipo ejecuta tres instancias del motor de Jess, una para cada componente del gestor de contexto y otra para el componente del controlador. Las fuentes de contexto no se implementan utilizando los motores de Jess.

Tabla 7: Ejemplo de registros de eventos que se ejecutan con el controlador

Event Description	ruleID	Parameters
EpilepticAlarm	ECARule1	Sr. *Carlos*
	ECARule2	Sr. *Carlos*
	ECARule3	Sr. *Carlos*
	ECARule4	Sr. *Carlos*
	ECARule5	Sr. *Carlos*, María
	ECARule6	Sr. *Carlos*, Alicia

Realización del Esfuerzo

En la presente tesis argumento que el enfoque propuesto facilita y simplifica el desarrollo de aplicaciones sensibles al contexto, ya que proporciona un marco que deriva sistemáticamente la realización de la aplicación. Este marco apoya la derivación de las siguientes partes de la aplicación: *modelos de contexto, detección de la situación y comportamientos reactivos*. Con el fin de tener una indicación de cómo nuestro enfoque facilita y simplifica el desarrollo de aplicaciones, se ha intentado cuantificar la especificación y los esfuerzos de realización.

Con el fin de proporcionar una cuantificación aproximada de los esfuerzos de especificación, contamos el número de diagramas de clase UML, y el número de líneas de código ECA-XDL (LDC). Para cuantificar aproximadamente los esfuerzos de realización, para ello contamos las líneas de código en Java y Jess que se derivan sistemáticamente de las especificaciones antes mencionadas. Para contar el código Java hemos utilizado Metrics (Eclipse Metrics, 2013), un plug-in Eclipse de código abierto que reúne varias métricas de proyectos. La tabla 8 muestra estos valores de cuantificación.

Tabla 8: Comparación entre la especificación y los esfuerzos de realización

UML clases(units)	OCL(LDC)	ECA-DX (LDC)	Java(LDC)	Jess(LDC)
23	9	21	663	126

Los resultados presentados en la tabla 8 indican que los esfuerzos requeridos para la fase de realización son una parte sustancial del desarrollo de la aplicación. Por lo tanto, nuestro enfoque alivia la realización de la aplicación, facilita y simplifica una parte sustancial del desarrollo de la aplicación.

Contrastación de Hipótesis

Prueba de Hipótesis General

H1: La arquitectura de software propuesta permitirá el desarrollo estándar de aplicaciones sensibles al contexto.

Ho: La arquitectura de software propuesta no permitirá el desarrollo estándar de aplicaciones sensibles al contexto.

Nivel de Arquitectura de Software*Nivel Aplicaciones sensible al contexto tabulación cruzada

			Nivel Aplicaciones sensible al contexto		Total
			Regular	Bueno	
Nivel de Arquitectura de Software	Deficiente	Recuento	0	1	1
		%	0,0%	4,8%	4,8%
	Regular	Recuento	4	3	7
		%	19,0%	14,3%	33,3%
	Bueno	Recuento	1	12	13
		%	4,8%	57,1%	61,9%
Total		Recuento	5	16	21
		%	23,8%	76,2%	100,0%

Pruebas de chi-cuadrado

	Valor	gl	Sig. asintótica (2 caras)
Chi-cuadrado de Pearson	6,462[a]	2	,040

a. 4 casillas (66,7%) han esperado un recuento menor que 5. El recuento mínimo esperado es ,24.

Como el p valor es 0.040 < 0.05 rechazamos la hipótesis nula y aceptamos la Hipótesis alternativa, es decir que la Arquitectura de Software se relaciona significativamente con aplicaciones sensibles al contexto a un nivel del 5% de significancia.

Prueba de Hipótesis Especifica 1

H1: La arquitectura de software, permitirá un desarrollo ordenado, eficaz y estándar para la implementación de sistemas sensibles al contexto.

Ho: La arquitectura de software, no permitirá un desarrollo ordenado, eficaz y estándar para la implementación de sistemas sensibles al contexto.

Nivel de Arquitectura*Nivel Aplicaciones sensible al contexto tabulación cruzada

			Nivel Aplicaciones sensible al contexto		
			Regular	Bueno	Total
Nivel de Arquitectura	Deficiente	Recuento	1	1	2
		% del total	4,8%	4,8%	9,5%
	Regular	Recuento	1	7	8
		% del total	4,8%	33,3%	38,1%
	Bueno	Recuento	3	8	11
		% del total	14,3%	38,1%	52,4%
Total		Recuento	5	16	21
		% del total	23,8%	76,2%	100,0%

Pruebas de chi-cuadrado

	Valor	gl	Sig. asintótica (2 caras)
Chi-cuadrado de Pearson	1,393[a]	2	,049
Razón de verosimilitud	1,361	2	,050
Asociación lineal por lineal	,012	1	,914
N de casos válidos	21		

a. 4 casillas (66,7%) han esperado un recuento menor que 5. El recuento mínimo esperado es ,48.

Como el p valor es 0.049 < 0.05 rechazamos la hipótesis nula y aceptamos la Hipótesis alternativa, es decir que la dimensión Arquitectura influye significativamente en el desarrollo de aplicaciones sensibles al contexto a un nivel del 5% de significancia.

Prueba de Hipótesis Específica 2

H1: El soporte y diseño de la arquitectura permitirá el diseñar el procesador de información del contexto desde una perspectiva de la ingeniería.

Ho: El soporte y diseño de la arquitectura no permitirá el diseñar el procesador de información del contexto desde una perspectiva de la ingeniería.

Nivel de Soporte y Diseño*Nivel Aplicaciones sensible al contexto tabulación cruzada

			Nivel Aplicaciones sensible al contexto		
			Regular	Bueno	Total
Nivel de Soporte y Diseño	Deficiente	Recuento	1	2	3
		% del total	4,8%	9,5%	14,3%
	Regular	Recuento	2	4	6
		% del total	9,5%	19,0%	28,6%
	Bueno	Recuento	2	10	12
		% del total	9,5%	47,6%	57,1%
Total		Recuento	5	16	21
		% del total	23,8%	76,2%	100,0%

Pruebas de chi-cuadrado

	Valor	gl	Sig. asintótica (2 caras)
Chi-cuadrado de Pearson	,788[a]	2	,675
Razón de verosimilitud	,782	2	,676
Asociación lineal por lineal	,615	1	,433
N de casos válidos	21		

a. 5 casillas (83,3%) han esperado un recuento menor que 5. El recuento mínimo esperado es ,71.

Como el p valor es 0.675> 0.05 rechazamos la hipótesis alternativa y aceptamos la Hipótesis nula, es decir que la dimensión Soporte y Diseño no influye significativamente en el desarrollo de aplicaciones sensibles al contexto a un nivel de significancia mayor al 5%.

Prueba de Hipótesis Específica 3

H1: Con la arquitectura propuesta los usuarios finales podrán mejorar su productividad mediante el uso de ciertas aplicaciones que no requieren interacción humana.

Ho: Con la arquitectura propuesta los usuarios finales no podrán mejorar su productividad mediante el uso de ciertas aplicaciones que no requieren interacción humana.

Nivel de Interacción*Nivel de Arquitectura de Software tabulación cruzada

			Nivel de Arquitectura de Software			Total
			Deficiente	Regular	Bueno	
Nivel de Interacción	Regular	Recuento	0	4	1	5
		Recuento esperado	,2	1,7	3,1	5,0
		%	0,0%	19,0%	4,8%	23,8%
	Bueno	Recuento	1	3	12	16
		Recuento esperado	,8	5,3	9,9	16,0
		%	4,8%	14,3%	57,1%	76,2%
Total		Recuento	1	7	13	21
		Recuento esperado	1,0	7,0	13,0	21,0
		%	4,8%	33,3%	61,9%	100,0%

Pruebas de chi-cuadrado

	Valor	gl	Sig. asintótica (2 caras)
Chi-cuadrado de Pearson	6,462[a]	2	,040
Razón de verosimilitud	6,441	2	,040
Asociación lineal por lineal	2,535	1	,111
N de casos válidos	**21**		

a. 4 casillas (66,7%) han esperado un recuento menor que 5. El recuento mínimo esperado es ,24.

Como el p valor es 0.040 < 0.05 rechazamos la hipótesis nula y aceptamos la Hipótesis alternativa, es decir que la dimensión de Interacción influye significativamente en el desarrollo de aplicaciones sensibles al contexto a un nivel del 5% de significancia.

Análisis e Interpretación

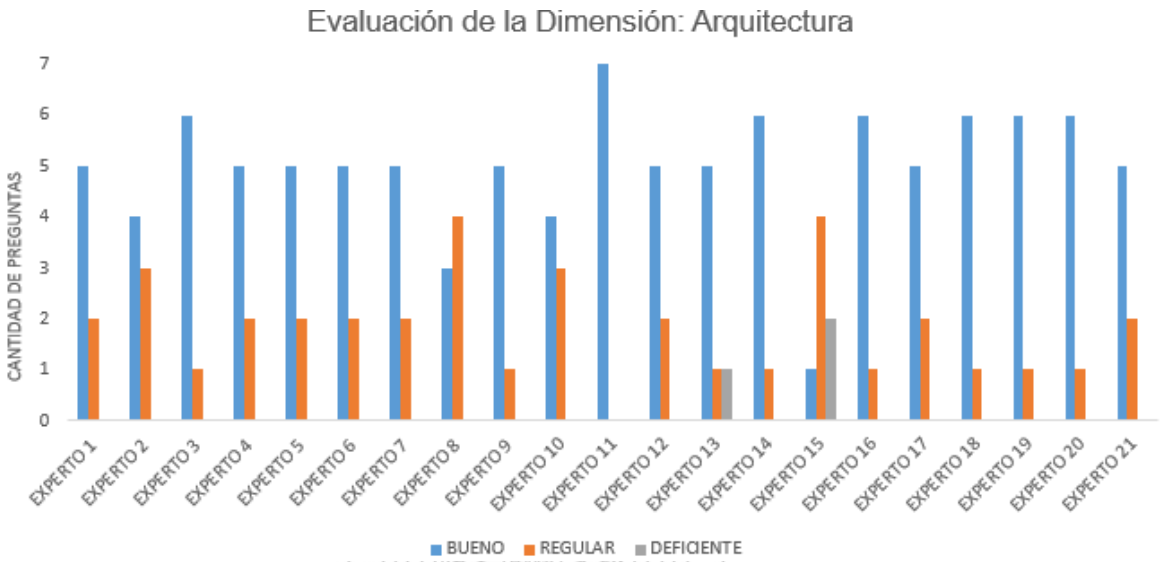
Después de haber desarrollado la aplicación prototipo se ha invitado a los expertos (21) para evaluar dicha aplicación, teniendo en cuenta los criterios para cada uno de las dimensiones. Las dimensiones son: Arquitectura, Soporte y Diseño e interacción.

Como primera dimension evaluada es la arquitectura que presenta el modelo para el desarrollo de aplicaciones sesnibles al contexto, para ello se considero siete criterios para calificar la arquitectura como(Bueno, Regular y Deficiente). Los criterioes evalaudos para esta dimension son:

- El modelo "ECA-DX" es aplicable para el desarrollo de aplicaciones sensibles al contexto.
- El Modelo "ECA-DX" proporciona un vocabulario para representar y compartir el conocimiento del contexto en un dominio informático omnipresente.
- El modelo "ECA-DX" basado en ontologías para la información, permite describir contextos semánticos de manera que es independiente del lenguaje de programación, sistema operativo o middleware subyacente.
- El presente modelo ayuda solucionar los problemas recurrentes asociados a la gestión de la información del contexto y reaccionar de forma proactiva a los cambios del contexto.
- El modelo propuesto presenta una estructura por cadenas jerárquicas de fuentes de contexto y gestores.
- El modelo "ECA-DX" presenta una estructura de componentes para apoyar el diseño e implementación de aplicaciones.
- El modelo "ECA-DX" cubre el manejo de sensores, y el prototipado y evaluación del sistema resultante.

Como podemos apreciar en la siguiente figura, para esta dimensión la valoración de los expertos en su totalidad ha sido favorable calificándolo como "Bueno", excepto el experto 15 que ha valorado el dominio como "Regular".

Figura 52: Evaluación de la Dimensión ARQUITECTURA del modelo propuesto.



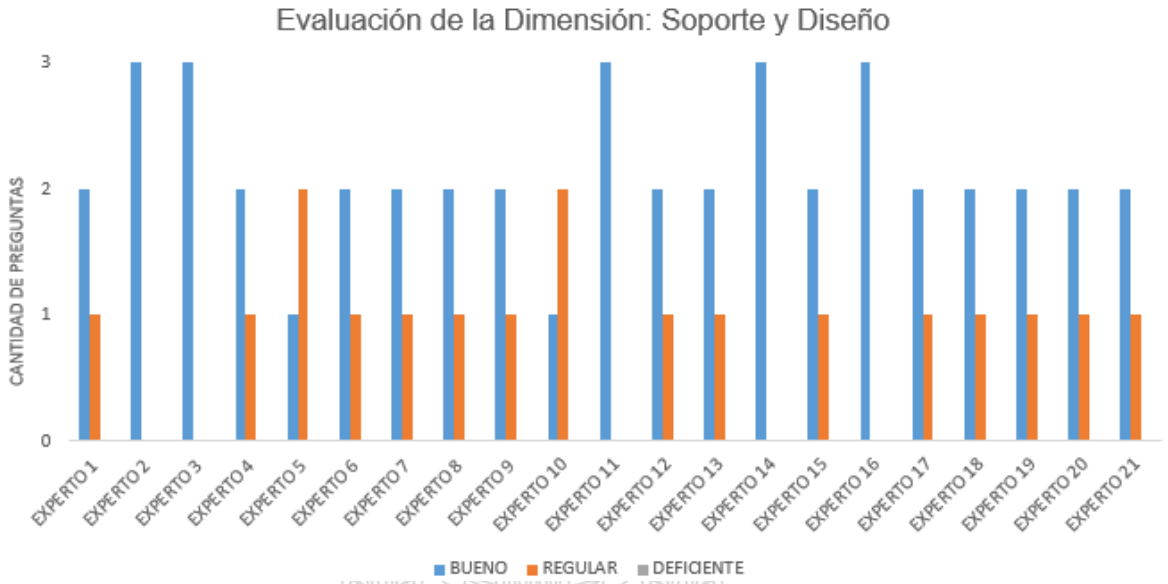
Como segundo punto evaluado es la dimensión de “Soporte y Diseño”, esta dimensión básicamente busca evaluar el soporte y diseño tecnológico para el modelo presentado, para ello se consideró siete criterios básicos para calificar dicha dimensión, los criterios son:

- El modelo propuesto ofrece soporte para la obtención de información del contexto de forma autónoma.
- Con el modelo “ECA-DX” el componente de motor de reglas es el núcleo de la arquitectura de controlador.
- Con el modelo “ECA-DX” los componentes pueden ser implementados en la parte superior de un motor de reglas de Jess, que es capaz de determinar cuando las condiciones son satisfechas por el conjunto actual de los datos disponibles de la memoria de trabajo.
- El modelo “ECA-DX” está diseñado con el fin de desacoplar los propósitos de las acciones, implementaciones y coordinar la composición de las mismas.
- Con el modelo “ECA-DX” los proveedores de diseño de aplicaciones y servicios son los actores responsables del desarrollo, comercialización y mantenimiento de las aplicaciones sensibles al contexto.
- Con el modelo “ECA-DX” el diseño del procesador de contexto se realiza desde una perspectiva de ingeniería.

- El modelo “ECA-DX” predomina las técnicas de interfaces multimodales donde se revela la preferencia de los usuarios por enfoques híbridos que combinen configuraciones automática y manuales.

En la figura 53 se aprecia la valoración que realizaron los 21 expertos con respecto a la dimensión “Soporte y Diseño” del modelo propuesto, en ello se puede identificar que la totalidad de expertos calificaron como “Bueno” para los siete criterios evaluados.

Figura 53: Evaluación de la Dimensión SOPORTE Y DISEÑO del modelo propuesto



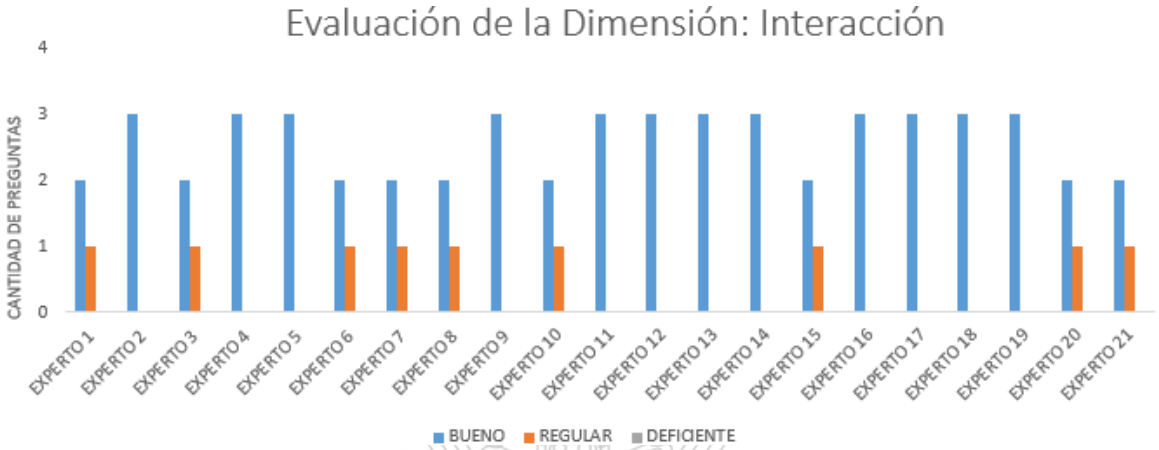
Como última dimensión evaluada es el de “*Interacción*”, esta dimensión tiene como objetivo mejorar la comunicación a través de las aplicaciones sin la interacción humana, esto con la ayuda del procesador del contexto para capturar la información de los usuarios a través de los puntos de interacción. Para validar el presente domino se utilizó dieciséis criterios que se describe a continuación.

- Con el modelo propuesto los usuarios finales pueden mejorar su productividad en el trabajo y mejorar su calidad de vida mediante el uso de ciertos tipos de aplicaciones sensibles al contexto.
- Con el modelo “ECA-DX” la comunicación se produce a través de las aplicaciones sin la interacción humana.
- A través del modelo “ECA-DX” los componentes del contexto como el procesador, es responsable de capturar el contexto del usuario a través de los puntos de interacción.

- El proveedor de servicios tiene como objetivo mantener una alta disponibilidad del controlador, procesador de contexto interno y componentes de la acción.
- Con el modelo "ECA-DX" la colaboración entre los componentes y procesadores de contexto permite la disponibilidad de información potencialmente más rica.
- Mediante el Modelo "ECA-DX" el componente controlador tiene por objeto controlar la flexibilidad, extensibilidad y capacidad de adaptación de la plataforma de manejo de contexto.
- Mediante el Modelo "ECA-DX" la activación de las reglas se produce en tiempo de ejecución de la plataforma.
- El modelo "ECA-DX" proporciona un vocabulario común y la comprensión de los principios de diseño.
- El modelo "ECA-DX" ayuda a la construcción de software complejo y heterogéneo.
- El modelo "ECA-DX", permite desarrollar aplicaciones sensibles, modulares y escalables.
- El Modelo "ECA-DX" es capaz de predecir el tiempo de la ubicación del usuario mediante la observación de los movimientos.
- El modelo "ECA-DX" es capaz de crear información del contexto a base de diversas fuentes de información de manera flexible.
- Las herramientas para el desarrollo de aplicaciones sensibles al contexto son accesibles y estándares.
- Mediante el presente modelo "ECA-DX" la localización se realiza a través del GPS, sensor del bluetooth como sensores de contexto.
- Mediante el presente modelo "ECA-DX" para superar las inexactitudes de GPS, utiliza información de las actividades que realiza el usuario, la cual se pone en relación con la ubicación.
- La generación de librerías específicas, orientadas al manejo de ciertas tareas sensible al contexto, que puedan ser incluidas en una aplicación para facilitar su desarrollo es también una preocupación en este campo.

En la figura 54, se puede apreciar que la evaluación de la dimensión "*Interacción*" del modelo propuesto, es calificado como "Bueno" por la totalidad de los expertos, esto es una indicación que el modelo ECA-DX, aporta a la comunicación a través de aplicaciones sensibles al contexto sin la interacción humana.

Figura 54: Evaluación de la Dimensión INTERACCIÓN del modelo propuesto.



Después de haber realizado el análisis por separado el resultado de cada uno de las dimensiones (Arquitectura, Soporte y Diseño e Interacción), se concluye que, en las tres dimensiones evaluadas, los 21 expertos han coincidido en calificar como "Bueno" al modelo propuesto, que representa el 78.48% de los criterios evaluados, y un 21.51% de los criterios han sido calificados como "Regular". Con este resultado se demuestra que la arquitectura propuesta en esta tesis facilita el desarrollo de aplicaciones sensibles al contexto, y con ello contribuye a solucionar los problemas recurrentes encontrados al momento integrar la información con las fuentes externas del contexto.

Con el fin de evaluar la escalabilidad y el rendimiento de la aplicación de atención médica, se ha definido el parámetro de evaluación del tiempo de reacción. El tiempo de reacción se define como el período de tiempo entre generar un evento y finalmente invocar una acción de una regla ECA-DX que se refiere a ese evento. Por ejemplo, podemos evaluar el tiempo de reacción entre una ocurrencia de evento EpilepticAlarm y desencadenar ECARule1, o podemos evaluar el tiempo de reacción entre una ocurrencia de evento AcceptRequest y desencadenar ECARule3. Por lo tanto, el tiempo de reacción consiste en el tiempo de procesamiento que tiene lugar entre una ocurrencia de evento, y la invocación de la acción. Capturamos el tiempo de reacción en EpilepticAlarmContextSource, que recibe una invocación de devolución de llamada del controlador cuando se activan las reglas ECARule2 y ECARule3. Estas invocaciones de callback están incluidas en la parte de acción de las reglas de Jess derivadas de ECARule2 y ECARule3, con el propósito de evaluar el rendimiento. Por lo tanto, cada vez

que se ejecuta una de estas reglas, el motor Jess del controlador invoca el componente EpilepticAlarmContextSource, que es capaz de calcular el tiempo de reacción. Estas invocaciones de devolución de llamada no forman parte del comportamiento de la aplicación normal.

Configuración General

Se ha medido en nuestra evaluación el tiempo de reacción entre la aparición de un EpilepticAlarm, y el desencadenamiento de ECARule2 y ECARule3. El tiempo de reacción entre una aparición del EpilepticAlarm y el desencadenante de ECARule3 es mayor que entre la misma ocurrencia de EpilepticAlarm y el desencadenamiento de ECARule2. Esto se debe a la composición de los eventos definidos en la ECARule3, que requiere que un evento AcceptRequest se produzca después de EpilepticAlarm.

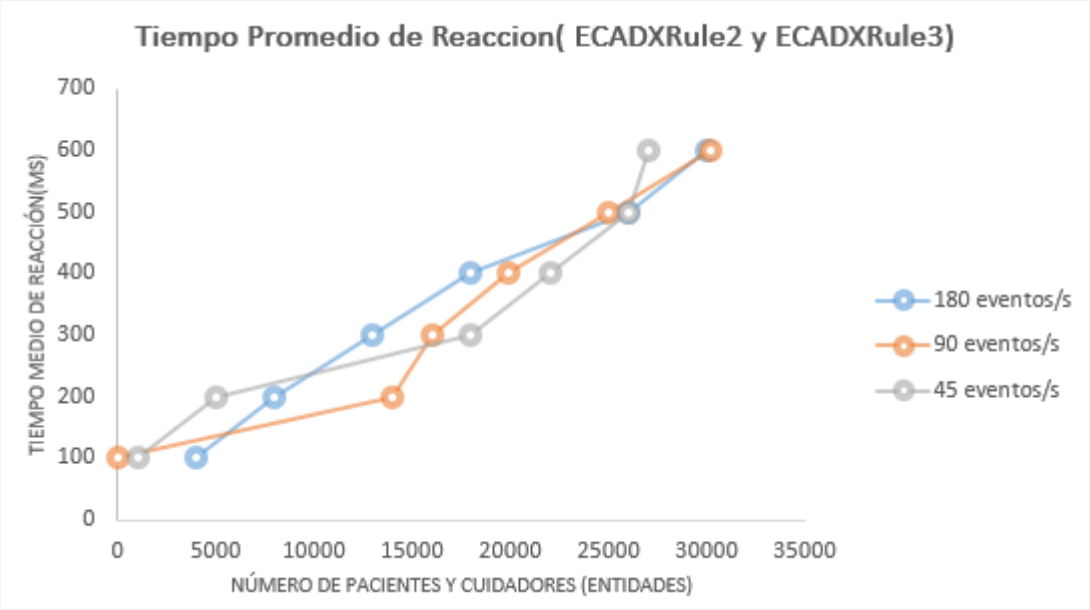
Hemos recogido las mediciones del tiempo de reacción para diferentes números de pacientes y cuidadores, y para diferentes cantidades de eventos generados por segundo. Para cada par (número de entidades, número de eventos por segundo), hemos recogido 100 mediciones del tiempo de reacción, para lo cual calculamos un valor promedio de tiempo de reacción. El tiempo de reacción medio incluye los tiempos de reacción para *ECARule2* y *ECARule3*.

Finalmente se ha realizado pruebas con el fin de evaluar cómo se comporta la aplicación en circunstancias extremas, como, con un número extremo de usuarios y eventos. Estos tipos de pruebas a menudo se denominan pruebas de esfuerzo. Las pruebas de esfuerzo tienen como objetivo evaluar la robustez y la disponibilidad del sistema bajo carga pesada. Con el fin de realizar estas pruebas de esfuerzo, se ha observado y medido los tiempos de reacción cuando el sistema se carga con hasta 35000 entidades, y está generando hasta 450 eventos por segundo.

Configuración Centralizada

La figura 55 muestra un gráfico que demuestra el incremento del tiempo de reacción promedio en milisegundos (eje Y) con respecto al número de entidades (eje X). Se muestran tres curvas, cada una representando un número particular de eventos generados por segundo. Para estas mediciones todos los componentes se ejecutan en una sola máquina con 4 GB de memoria.

Figura 55: Tiempos de reacción promedio de ECARule2 y ECARule3 en una configuración centralizada



Se comprobó que aproximadamente el 85% del tiempo de reacción consiste en el tiempo de procesamiento requerido por el motor de Jess. El otro 15% se debe al mecanismo de consumo de eventos implementado por el componente controlador. El último puede ser mejorado optimizando las estructuras de eventos usando, por ejemplo, tablas hash.

La forma de estas curvas puede explicarse considerando cómo funciona Jess. Jess implementa el algoritmo Rete, que construye una red de nodos en la memoria, cada nodo representa uno o más patrones. Los hechos que se están añadiendo a la memoria de trabajo son procesados por esta red de nodos. En los bordes inferiores de la red están los nodos que representan reglas individuales. Cuando un conjunto de hechos se filtra hasta los bordes de la red, ha pasado todas las pruebas del lado izquierdo de una regla particular y la regla del lado derecho se ejecuta. El algoritmo *Rete* recuerda los resultados de las pruebas pasadas a través de las iteraciones del bucle de reglas. Sólo se prueban nuevos hechos con respecto a cualquier regla.

Cuando aumentamos el número de entidades, el tamaño de la memoria de trabajo aumenta proporcionalmente al número de entidades y sus atributos de contexto. Como se ha mencionado, el aumento de la memoria de trabajo degrada de forma lineal el rendimiento del motor Jess, lo que aumenta consecuentemente el tiempo de reacción. Además, las notificaciones de eventos también se agregan a la memoria de trabajo. Aunque no se mantienen durante mucho tiempo debido al consumo de eventos y el

intervalo de detección, la adición y eliminación de eventos requiere reorganizar la red *Rete* de nodos, lo que también consume tiempo de procesamiento y aumenta el tiempo de reacción. Estos comportamientos se reflejan en las formas de dos curvas en la figura 47, es decir, los 45 y 90 eventos por segundo. Para estas curvas, el tiempo de reacción es aproximadamente lineal con respecto al número de entidades en el sistema.

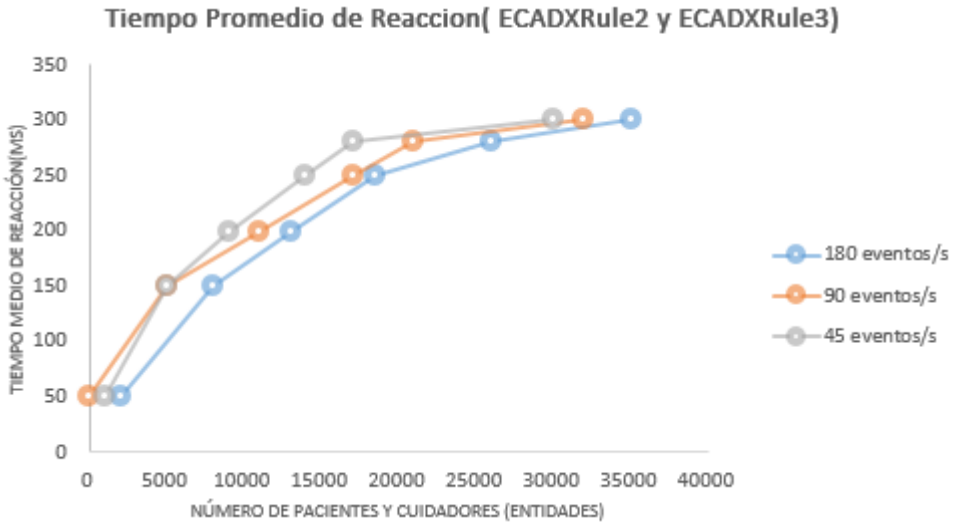
En la curva 180 eventos, para más de 7500 entidades, vemos que el tiempo de reacción es altamente impredecible, lo que indica un punto de saturación. Esto se explica por el hecho de que Jess no puede procesar los eventos lo suficientemente rápido como para mantenerse al día con la frecuencia en la que se generan los eventos (180 eventos / s). Por lo tanto, las notificaciones de eventos no se pueden procesar en el orden en que llegan, lo que lleva a la extenuación de eventos. Para manejar este punto de saturación, es probablemente necesario un mecanismo de eventos de colas de eventos.

Cuando el sistema llega a un punto de saturación y el tiempo de espera para procesar eventos es mayor que el intervalo de ventana de detección definido para una determinada regla, es posible que se produzca un comportamiento incorrecto. Por ejemplo, si definimos un intervalo de ventana de detección de 1 segundo para ECARule2. Supongamos que, debido a la saturación del sistema, una notificación EpilepticAlarm tarda 3 segundos en ser procesada. Esto significa que este evento se descarta antes de que sea consumido por ECARule2, porque se cayó fuera del intervalo de la ventana de detección. En este ejemplo, ECARule2 debería haber sido activado, lo que caracteriza el comportamiento incorrecto del sistema. Se recomienda definir un valor de intervalo de ventana de detección grande, que puede acomodar los retrasos causados por la saturación del sistema. Si hay restricciones de temporización de eventos específicos que se definen en una regla, como "event1 se dispara, 50 segundos después debería producirse *event2* ", estas restricciones deberían incluirse como condiciones en la cláusula When de la regla.

Configuración Distribuida

La figura 56 muestra un gráfico similar al de la figura 49. Para este experimento distribuimos los componentes en dos máquinas diferentes con 3GB de memoria y 4GB de memoria. En la máquina de 3GB ejecutamos todas las cinco fuentes de contexto, y en la máquina de 4GB los gestores de contexto y el controlador.

Figura 56: Tiempos de reacción promedio de ECARule2 y ECARule3 en una configuración distribuida

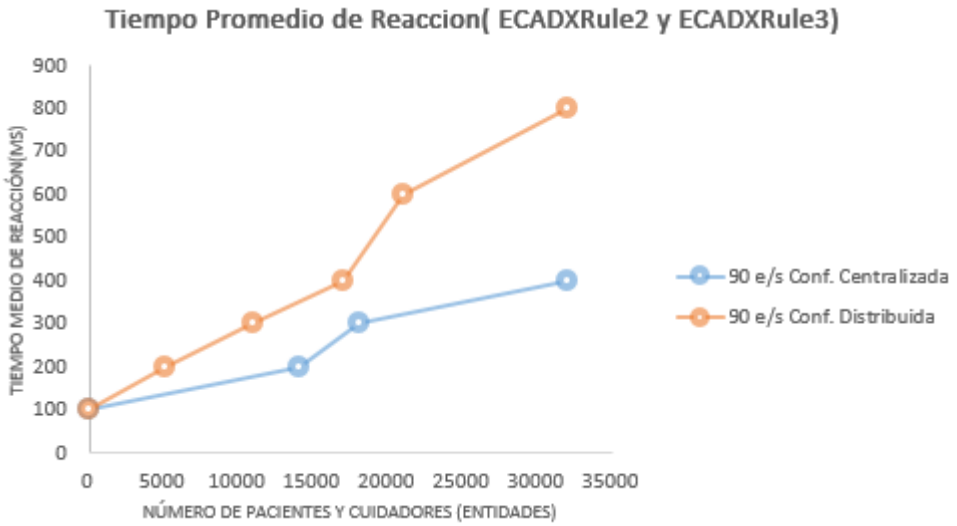


El gráfico muestra que los tiempos de reacción promedio en una configuración distribuida mejoran considerablemente. Además, el punto de saturación no se ha alcanzado incluso con un mayor número de entidades, y una mayor frecuencia de generación de eventos. Este comportamiento puede explicarse por otra característica de Jess, que es el uso de la memoria. Se dice que Jess es una "aplicación intensiva en memoria" (Sandia, 2013), ya que Rete es un algoritmo que explícitamente negocia espacio para la velocidad. Por lo tanto, el rendimiento de Jess se ve afectado por la disponibilidad de memoria. En la configuración distribuida, los tres motores funcionan en máquinas separadas con mayor disponibilidad de memoria, lo que mejora el rendimiento de estos motores. Además, las fuentes de contexto consumen una cantidad considerable de memoria para simular grandes cantidades de información del contexto y eventos. En la configuración distribuida, los motores no necesitan competir con las fuentes de contexto para la memoria.

La Figura 57 compara los tiempos de reacción promedio de las configuraciones centralizada y distribuida cuando se generan 90 eventos por segundo. Al principio (hasta 7500 entidades), la configuración centralizada funciona mejor. Puesto que en este punto no se requiere mucha memoria, la latencia de red para intercambiar eventos más el tiempo de procesamiento requerido por los motores en la configuración distribuida es mayor que el tiempo de procesamiento requerido en la configuración centralizada. Por lo tanto, hasta este punto, el tiempo de reacción en la configuración centralizada es mejor que en la configuración distribuida. Este comportamiento cambia cuando aumentamos el número de entidades, debido al aumento en el uso de memoria.

La disponibilidad de memoria también explica por qué las curvas en la configuración distribuida son más predecibles que las curvas en la configuración centralizada. En la configuración centralizada, en la que los recursos son más limitados, los motores de Jess son más sensibles a otras actividades del sistema operativo, como el intercambio, causando variaciones impredecibles en el rendimiento.

Figura 57: Comparación de los tiempos de reacción entre la configuración Centralizada y Distribuida



CAPITULO V

DISCUSIÓN

Discusión

En esta sección se demuestra la viabilidad del enfoque de desarrollo presentado en esta tesis. Se ha proporcionado el proceso de diseño completo de la aplicación, que se ocupan de diferentes requisitos de la aplicación de gestión de la salud. Además, hemos implementado un prototipo para la aplicación sanitaria, que demostró que el enfoque puede ser implementado en un sistema informático. También hemos llevado a cabo evaluaciones que nos permitieron sacar conclusiones relevantes con respecto a las cuestiones de rendimiento y escalabilidad. Hemos considerado una condición médica particular en nuestra aplicación de asistencia sanitaria, es decir, epilepsia. Nuestro diseño y prototipo se pueden adaptar fácilmente a otras condiciones médicas, tales como condiciones médicas relacionadas con el corazón. En tales escenarios, un paciente que sufre de una enfermedad cardíaca puede ser controlado e informado de posibles alteraciones anormales en las señales cardíacas.

Los párrafos siguientes analizan el enfoque de desarrollo a la luz de los requisitos definidos. Se describe brevemente cómo el enfoque cumple con cada uno de estos requisitos.

- ✓ *El soporte para el desarrollo y el despliegue rápidos de aplicaciones se* logra con la derivación sistemática de la realización de la aplicación a partir de la especificación de la aplicación. Dado que nuestro enfoque alivia la realización de la aplicación, facilita y simplifica una parte sustancial del desarrollo de la aplicación. Además, los comportamientos reactivos de la aplicación se pueden implementar en tiempo de ejecución de la plataforma, por medio de las especificaciones de la ECA-DX.
- ✓ *La flexibilidad y la extensibilidad* se logra con la posibilidad de: añadir varios comportamientos reactivos a la aplicación en tiempo de ejecución; ampliar la plataforma con actividades de detección de situaciones que no se hayan definido en el momento de la especificación, y extender la plataforma con componentes de control y de servicio de acción que no se han definido en tiempo de diseño.
- ✓ *El apoyo a la adaptación* se ha conseguido mediante el marco de detección de la situación y los servicios de control. El marco de detección de situaciones permite a las aplicaciones no sólo utilizar la información del contexto para reaccionar a petición del usuario, sino también para tomar iniciativas como resultado de actividades de razonamiento del contexto (que se ejecutan continuamente). En este sentido, el marco de situación permitió la adaptación de la aplicación además de la adaptación de la aplicación reactiva. Los servicios de control también realizan reactividad en nombre de las aplicaciones, brindando apoyo explícito para la adaptación.

- ✓ *El apoyo a la distribución del contexto y el razonamiento de la situación* se logra mediante la distribución de fuentes de contexto, gestores de contexto y componentes del controlador. En esta sección se ha demostrado la distribución usando el estilo arquitectónico orientado al servicio.
- ✓ *El rendimiento* se evalúa recopilando mediciones del tiempo de reacción del prototipo de atención. Debido al algoritmo de Rete, Jess procesa eficientemente las reglas, lo que conduce a tiempos de reacción satisfactorios para nuestros experimentos de atención médica (en el orden de milisegundos). A nuestro leal saber y entender, los tiempos de reacción de este orden serían aceptables para varios tipos de aplicaciones sensibles al contexto.
- ✓ *La escalabilidad* de nuestro enfoque de desarrollo se ha evaluado en la aplicación de atención médica mediante el aumento del número de entidades, el número de eventos generados por segundo y el número de reglas. Hemos demostrado que las escalas de enfoque para un número arbitrario de las normas ECA-DX, ya que las normas ECA-XDL pueden ser distribuidos a través de varios componentes del controlador, que se pueden agregar a la demanda. También hemos demostrado que el sistema se comporta satisfactoriamente para un gran número de entidades y eventos.

Conclusiones

El conocimiento del contexto ha emergido como una característica importante y deseable en aplicaciones ubicuas. Esta característica se refiere a la capacidad de las aplicaciones para utilizar la información sobre el entorno del usuario (contexto) con el fin de adaptar los servicios a la situación actual del usuario y las necesidades. Se ha argumentado en la presente tesis que el diseño de aplicaciones sensibles al contexto es una tarea desafiante, que justifica el desarrollo de nuevos métodos, abstracciones y plataformas.

Nuestro objetivo fue proporcionar una solución integrada para el desarrollo de aplicaciones sensibles al contexto. El objetivo principal fue facilitar el desarrollo de aplicaciones sensibles al contexto, centrándose en dos aspectos: ofrecer abstracciones de modelado de contexto y proporcionar apoyo infraestructural mediante una arquitectura. Las abstracciones de modelado del contexto proporcionan a los desarrolladores de aplicaciones fundamentos conceptuales apropiados que pueden ampliarse y especializarse para los requisitos específicos de las aplicaciones. La arquitectura propuesta permite delegar la funcionalidad de la aplicación a la plataforma, reduciendo el esfuerzo de desarrollo de aplicaciones (como tiempo y costos). Esto permite a los desarrolladores de aplicaciones centrarse mejor en su negocio principal, en lugar de molestarse con los detalles de realización de la aplicación.

Se ha demostrado que la arquitectura propuesta en esta tesis verdaderamente facilita el desarrollo de aplicaciones sensibles al contexto. Sin embargo, el éxito de la conciencia del contexto no puede lograrse utilizando únicamente el enfoque de desarrollo. También son necesarios resultados de otras iniciativas de investigación complementarias, que quedan fuera del alcance de esta tesis, como: (a) la tecnología de sensores, que permite recopilar diversos valores de atributos de contexto de los entornos de usuarios, (b) middleware de comunicación, que permite la distribución transparente de las partes de la aplicación, (c) interfaces de usuario, que entregan servicios orientados al contexto a los usuarios finales de una manera útil, (d) seguridad y privacidad, que garantizan que la información sensible a la privacidad se utilice correctamente, y (e) las tecnologías móviles, que permiten incorporar computacional en dispositivos móviles y de comunicación.

Recomendaciones

Las recomendaciones se enfocada básicamente en:

1. Abstracciones de modelos de contexto: que incluyen modelos conceptuales de contexto y sus respectivas realizaciones.
2. Arquitectura de manejo de contexto: que incluye los servicios genéricos que se ha diseñado para soportar el desarrollo de aplicaciones sensibles al contexto.

Modelo Estructural del Contexto:

El proceso de identificación del contexto relevante consiste en determinar las "condiciones" de las entidades en el universo del discurso de la aplicación (por ejemplo, un usuario o su entorno) que son relevantes para una aplicación contextual o una familia de tales aplicaciones. Hemos sostenido a lo largo de esta tesis que el universo de la aplicación de discurso debe ser adecuadamente caracterizado. Como resultado de este proceso de caracterización, se obtiene un modelo conceptual de contexto. Argumentamos que la definición de dicho modelo de contexto debe preceder al diseño detallado de una aplicación contextual.

Como parte del modelado conceptual del contexto, se ha propuesto fundamentos conceptuales básicos para el modelado del contexto, que permite a los diseñadores de las aplicaciones contextuales representar elementos relevantes del universo en las aplicaciones sensibles. Estas bases conceptuales deben facilitar la especificación de modelos de contexto que sean más claros y más fáciles de entender. En los fundamentos conceptuales propuestos, los diseñadores de aplicaciones tienen instrucciones de separar los conceptos de entidad y contexto. Además, el contexto debe caracterizarse como intrínseco o relacional.

Dado que el modelado conceptual se centra en soportar estructuraciones e instalaciones inferenciales que están psicológicamente fundamentadas, la adecuación de nuestra técnica de modelado de contexto está determinada por su contribución al entendimiento común del contexto entre las partes interesadas de una aplicación contextual. Por lo tanto, se justifica opciones de modelado con resultados de ontologías fundamentales, que están en línea con las teorías conceptuales en filosofía y ciencias cognitivas.

Modelos de Situación:

El modelo de contexto estructural permite a los diseñadores de aplicaciones representar todos los posibles estados de una aplicación del universo, sin discriminar las situaciones particulares que pueden ser de interés para las aplicaciones. Se ha introducido el

concepto de situación. Una situación es un concepto compuesto cuyos constituyentes son entidades y sus condiciones de contexto. Las situaciones amplían los modelos de contexto, ya que pueden estar compuestas por clases más elementales de las condiciones del contexto, y además pueden estar compuestas de situaciones existentes.

También se ha propuesto un nuevo enfoque basado en modelos para la especificación de situaciones. Las situaciones se especifican utilizando diagramas de clase UML enriquecidos con restricciones de OCL para definir las condiciones bajo las cuales se permite que existan situaciones de cierto tipo.

Como parte de la fase de modelado de información de contexto, se considera: (a) cómo se detecta el contexto, (b) cómo se produce, aprende, infiere y utiliza la información de contexto, y (c) la validez y la información de Calidad de Contexto (Idcc). La Idcc se refiere a la meta-información que describe la calidad de la información del contexto.

Arquitectura de Contexto

En la presente tesis se propone arquitectura Event-Control-Action-ECA que puede ser aplicado de manera beneficiosa en el desarrollo de aplicaciones sensibles al contexto, la arquitectura ECA ha permitido efectivamente la distribución de responsabilidades entre las distintas partes empresariales en una plataforma de servicios que conoce el contexto. La aplicación de estos principios de diseño mejora en gran medida la extensibilidad y flexibilidad de la plataforma, ya que los procesadores de contexto y los componentes de acción pueden desarrollarse y desplegarse por separado. Además, la definición del comportamiento de la aplicación a través de reglas de condición permite el despliegue dinámico de los comportamientos de aplicación de contexto y configuración de la plataforma en tiempo de ejecución.

Arquitectura de Manejo de Contexto

En esta arquitectura, se define el procesador de contexto, controlador y componentes de acción.

Los componentes del procesador de contexto recopilan información del entorno del usuario (contexto). Con base en las mediciones de información del contexto, los componentes del procesador realizan el razonamiento del contexto y generan eventos de contexto y situación que se ofrecen a los otros componentes de la plataforma.

El componente de controlador tiene como objetivo ejecutar comportamientos específicos de aplicación del contexto dentro de la plataforma. En este ámbito, los comportamientos de la aplicación pueden describirse como reglas lógicas, que se denominan reglas de Condición-Acción-Condición que son consistentes con la

arquitectura Event-Control-Action. Con el fin de realizar los comportamientos de la aplicación, el componente controlador recoge el contexto de los componentes del procesador de contexto. Cuando se cumple la combinación de condiciones de contexto definida por los comportamientos específicos de la aplicación, el componente controlador invoca las acciones necesarias en los componentes de acción.

Trabajos futuros

En la presente tesis se ha identificado tres futuras investigaciones en las siguientes áreas: *Manejo del contexto y la situación*, automatización de Procesos y Plataformas de manejo de contexto.

Manejo del contexto y la situación: dado que la tecnología de los sensores es imperfecta por definición, como parte de nuestras abstracciones de modelado de contexto y situación, se ha definido parámetros de calidad simples que especifican la calidad de la información de contexto con respecto a la probabilidad de corrección y precisión. No se ha discutido en esta tesis cómo se evalúa la Calidad de Contexto (Cdc), cómo se puede derivar la calidad de la información de la situación a partir de la información de Cdc, y cómo mejorar la Cdc utilizando, por ejemplo, la redundancia de fuentes de contexto. Estos aspectos de la Cdc que requiere una investigación más profunda. Además, no se ha incorporado Cdc en el prototipo propuesto. Las extensiones a nuestro enfoque también deben incluir la implementación de abstracciones.

En la presente tesis se ha estudiado el razonamiento del contexto mediante el marco de detección que detecta situaciones particulares de interés (pasadas o actuales). No se ha abordado las actividades de razonamiento utilizando técnicas de aprendizaje y predicción. Las ocurrencias pasadas del contexto y de la información de la situación se pueden extrapolar para predecir los comportamientos futuros del usuario y de la aplicación. Del mismo modo, el patrón de ocurrencias de la situación podría ser utilizado en un proceso de aprendizaje con el fin de anticiparse a todo tipo de comportamientos de la aplicación. *La investigación futura es necesaria para incorporar el aprendizaje y la predicción en nuestro contexto de modelización de las abstracciones*. En investigaciones futuras se deben estudiar mecanismos más complejos para descartar registros históricos de situación que ya no se utilizarán. La solución actual utiliza el tiempo de vida para descartar registros históricos. Una solución alternativa es eliminar todos los datos históricos a los que no se refiere ninguna situación activa. Esto requiere una inspección sobre las dependencias del tipo de situación, que tiende a ser compleja.

Automatización de procesos

En el enfoque de realización tanto para la detección de situaciones como para la ejecución de reglas de ECA-XDL, se considera a Jess como la tecnología subyacente. Con el fin de mapear la situación y las especificaciones de la regla ECA-XDL para el código de Jess, se proporciona asignaciones que se pueden utilizar para derivar sistemáticamente las reglas de las especificaciones UML, OCL y ECA-XDL. Se ha automatizado parcialmente las asignaciones de las especificaciones de ECA-DX a Jess.

Como parte del trabajo futuro, nuestro enfoque debe ser extendido para proporcionar transformaciones automatizadas de especificaciones a realizaciones tanto para la detección de la situación como para reglas ECA-DL. Preferentemente, los procesos de automatización deben basarse en el enfoque MDA, con el fin de facilitar el mantenimiento de reglas cada vez que se libere una versión más reciente del motor Jess, y la aplicación de las normas ECA-DX en una plataforma basada en normas diferente.

Finalmente, los futuros tipos de datos de medición, de origen de contexto y componentes del gestor de contexto deberían generarse automáticamente a partir de los modelos conceptuales. Como se ha argumentado, el enfoque proporciona directrices para llenar la brecha entre los modelos conceptuales y los modelos de información de contexto. Se ha observado que la derivación de los modelos de información de los modelos conceptuales sigue ciertos patrones. Por lo tanto, debería ser posible derivar automáticamente los tipos de datos de medición de los modelos conceptuales.

Plataforma de manejo de Contexto

Por razones de alcance, dos tipos de servicios han sido explícitamente ignorados en esta tesis, los servicios de descubrimiento y los servicios de acción. Un servicio de descubrimiento permite que se encuentren servicios distribuidos que no se conocen de antemano utilizando diversas características del servicio, como el tipo de información ofrecida por el servicio, la calidad de la información, la calidad del servicio, etc. Se han llevado a cabo esfuerzos paralelos en el campo del descubrimiento de servicios sensibles al contexto (Hesselman et al., 2006). Nuestro mecanismo de descubrimiento simplificado actual podría ampliarse con estos esfuerzos a fin de permitir un descubrimiento más rico del contexto Aprovisionamiento, control, acciones y servicios específicos de la aplicación, en tiempo de ejecución de la plataforma.

Los servicios de acción que se ha presentado en esta tesis ofrecen acciones sencillas, que pueden ser invocadas desde el componente controlador. En la práctica, las acciones suelen tener interdependencias, como cuando una acción sólo se puede habilitar cuando se ha ejecutado otra acción. Los servicios de acción que se propone podrían ampliarse para permitir composiciones complejas e interdependencias de servicios de acción. El futuro trabajo que podría investigarse es cómo *ECA-DL* se puede ampliar para permitir la composición de la acción de especificación basada en la norma de servicios de lenguajes de especificación, como BPEL (Jordan & Alves, 2007).

Con respecto a los aspectos cuantitativos transversales, se ha evaluado el rendimiento y la escalabilidad, pero no se ha considerado la fiabilidad de los componentes de la plataforma de manejo del contexto. En trabajos futuros, se podría

estudiar cómo abordar la fiabilidad en caso de que ocurran circunstancias inesperadas, como cuando el aprovisionamiento de contexto o los servicios de control dejan de funcionar. En esta tesis también se ha discutido brevemente la importancia de las cuestiones de privacidad, seguridad y confianza. Dado el carácter sensible a la privacidad de la información del contexto, creo que la aceptación de la conciencia del contexto por parte de los usuarios de aplicaciones dependerá en gran medida de cómo los usuarios pueden confiar en que las aplicaciones usan su información de contexto de manera privada y segura. Estas cuestiones también se identifican para trabajos futuros.

Por último, para demostrar la idoneidad de la propuesta en otros dominios, las aplicaciones sensibles al contexto deben desarrollarse en la parte superior de nuestra plataforma para diferentes áreas de aplicación.

Referencias Bibliográficas

Abowd, G. D., Dey, a K., Brown, P. J., Davies, N., Smith, M. E., Steggles, P., & Session, P. (1999). Towards a better understanding of context and context-awareness. *CHI 2000 Workshop on the What Who Where When and How of Contextawareness*, *4*(What, Who, Where, When and How of Context-Awareness). https://www.hiit.fi/files/admin/publications/Publications/CAPS2005_proceedings.pdf

Anind, D., Abowd, G., & Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, *16*(2), 97–166. https://dl.acm.org/citation.cfm?id=1463110

Bardram, J. E. (2004). The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications, 18.

Biegel, G., & Cahill, V. (2004). A framework for developing mobile, context-aware applications. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004.* https://www.hiit.fi/files/admin/publications/Publications/CAPS2005_proceedings.pdf

Bradley, N., & Dunlop, M. (2002). Understanding Contextual Interactions to Design Navigational Context-Aware Applications. *Human Computer Interaction with Mobile Devices: 4th International Symposium, Mobile HCI 2002 Pisa, Italy, September 18--20, 2002 Proceedings*, 349–353. http://www.springer.com/gp/book/9783540408215

Bradley, N., & Dunlop, M. (2005). Toward a Multidisciplinary Model of Context to Support Context-Aware Computing. *Human-Computer Interaction*, *20*(4), 403–446. https://doi.org/10.1207/s15327051hci2004_2

Brumitt, B., Meyers, B., Krumm, J., Kern, A., & Shafer, S. (2000). Easyliving: Technologies for intelligent environments. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.8033&rep=rep1&type=pdf

Camarena, S. J., Trueba, E. A., Martínez, R. M., & López, G. M. de L. (2012). Automatización de la codificación del patrón modelo vista controlador(MVC) en proyectos orientados a la Web. *Ciencia Ergo Sum*, *19*(3). Retrieved from http://www.redalyc.org/html/104/10423895005/

Canelón, R., Losavio, F., Matteo, A., & Chirinos, L. (2009). Modelo conceptual para modelación de aplicaciones móviles sensibles al contexto. *Revista de La Facultad de Ingeniería Universidad Central de Venezuela*, *24*(2), 93–103.

Cangrejo, L. (2014). Sensibilidad al contexto y a la localización en la computación ubicua. Retrieved from http://52.0.140.184/revsistemas1/index.php/ediciones-revista-sistemas/edicion-no-132/item/167-uno-sensibilidad-al-contexto-y-a-la-localización-en-la-computación-ubicua

Castro, M. F. (2003). El proyecto de investigación y su esquema de elaboración. Retrieved from http://www.urbe.edu/UDWLibrary/InfoBook.do?id=9590

Causa, E. (2009). Algoritmos para el análisis de formas y reconocimiento de patrones bitonales Palabras claves Captura de movimiento por substracción de video y el problema de los, 1–19.

Chen, H., Finin, T., & Joshi, A. (2003). An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, *18*(3), 197–207. https://www.cambridge.org/core/journals/knowledge-engineering-review/article/an-ontology-for-contextaware-pervasive-computing-environments/AA9BDD7E22480F3478A523014AC6B794

Costa, P. D. (2007). *Architectural Support for Context-Aware Applications: From Context Models to Services Platforms*. Centre for Telematics and Information Technology.

Demuth, B. (2012). OCL : Object Constraint Language. *Language*, *36*(May), 1–11. https://doi.org/10.1145/1921532.1921543

Floréen, P., Lindén, G., Niklander, T., & Raatikainen Floréen, K. (2005). PROCEEDINGS OF THE WORKSHOP ON CONTEXT AWARENESS FOR PROACTIVE SYSTEMS CAPS 2005, 16–17. Retrieved from https://www.hiit.fi/files/admin/publications/Publications/CAPS2005_proceedings.pdf

Du, W., & Wang, L. (2008). Context-aware application programming for mobile devices. *Proceedings of the 2008 C3S2E Conference on - C3S2E '08*, 215. https://doi.org/10.1145/1370256.1370292

Eclipse Metrics. (2013). Eclipse Metrics plugin. Retrieved from https://sourceforge.net/projects/metrics/

Fahy, P., & Clarke, S. (2000). CASS -Middleware for Mobile Context-Aware Applications.

Federico, Z. P. L. B. (2006). *Integrando Sensibilidad al Contexto Mediante Aspect Oriented Programming*.

Fernández(Accenture), J. A. (2011). Arquitectura Orientada a Servicios (SOA) Cómo reformular la Arquitectura Corporativa. *M2PressWIRE*, Creación;2011;Recuperado:8 diciembre 2015. Retrieved from http://www.accenture.com/SiteCollectionDocuments/Local_Spain/PDF/SOA.pdf%5Cnhttp://search.ebscohost.com/login.aspx?direct=true&db=nfh&AN=16PU716768326&lang=es&site=ehost-live

Fernandez, A. G., & Estrada, E. L. (2005). Sensores Magneticos e Inductivos, 149. Retrieved from http://www.uaeh.edu.mx/docencia/Tesis/icbi/licenciatura/documentos/Sensores magneticos.pdf

Ferreira, L. P. (1992). *Architectural Notes - A Framework for Distributed Systems Development*. Centre for Telematics and Information Technology.

Fetzer, A. (2007). *Context and Appropriateness*. (A. Fetzer, Ed.) (Vol. 162). Amsterdam: John Benjamins Publishing Company. https://doi.org/10.1075/pbns.162

Freeman, E. (1984). *Strategic management : a stakeholder approach*. Retrieved from https://books.google.com.pe/books?hl=es&lr=&id=NpmA_qEiOpkC&oi=fnd&pg=PR5&dq=edward+freeman+stakeholders&ots=6-kkJ6S3OH&sig=U0NSDP9GRBCryq3GrIu6Afa1mg8#v=onepage&q=edward freeman stakeholders&f=false

Friedman, E. (2008). The Rule Engine for the Java ™ Platform Version 7.1p2 November 5, 2008. *Sandia National Laboratories*.

Gamarra, J. (2008). Recomendaciones para la adopción de SOA, 8.

Gellersen, H.-W., Schmidt, A., & Beigl, M. (2001). Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts. Retrieved from https://www.teco.edu/~michael/publication/monet.pdf

Georgakopoulos, D., & Papazoglou, M. (2009). Service-Oriented Computing.

Gómez, E. (2011). Modelo arquitectural para aplicaciones móviles usando el enfoque de

líneas de producción dinámica de software.

Gonzales, M. (2011). *Estudio de Arquitecturas de Redes Orientadas a Servicio*. Universidad Politecnica de Catalunya.

González, F. N., Alejandres, H. H., & González, J. G. (2015). Arquitectura de un sistema de recomendación semántico sensible al contexto para entornos tipo campus. *Ciencias de La Información*, *46*(1), 11–17. Retrieved from https://biblat.unam.mx/es/revista/ciencias-de-la-informacion/articulo/arquitectura-de-un-sistema-de-recomendacion-semantico-sensible-al-contexto-para-entornos-tipo-campus

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Model*. *PhD thesis* (Vol. 15). https://doi.org/10.1007/978-3-642-31095-9_45

Gutiérrez, J. (2006). ¿ Qué es un framework web?, 1–4. Retrieved from http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Gutiérrez, J. (2006). ¿Qué es un framework web? Retrieved from http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Haya, P. A. (2006). *Tratamiento De Información Contextual En Entornos Inteligentes*. *Tesis Doctoral*. Universidad Autónoma de Madrid. Retrieved from http://amilab.ii.uam.es/lib/exe/fetch.php?media=doc:phaya06tratamiento.pdf

Hesselman, A., Tokmakoff, P., Pawar, S. I., Hesselman, C., Tokmakoff, A., Pawar, P., & Iacob, S. (2006). Discovery and composition of services for context-aware systems. *Proceedings of the 1st European Conference on Smart Sensing and Context (EuroSCC'06)*, 67–81. https://doi.org/10.1.1.101.6969

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., & Retschitzegger, W. (2003). Context-awareness on mobile devices - the hydrogen approach. In *36th Annual Hawaii International Conference on System Sciences.* Retrieved from https://www.hiit.fi/files/admin/publications/Publications/CAPS2005_proceedings.pdf

Imen, I., & Faouzi, M. (2013). A Pervasive System Architecture For Smart Environments, *3*(5), 77–89.

Izquierdo, A. S. (2013). *Integración de Fuentes de con Informacion de contexto y sistemas Interactivos*. Universidad Politecnica de Madrid.

Jordan, D., & Alves, A. (2007). Web Services Business Process Execution Language Version 2 . 0. *Language*, *11*(April), 1–264. https://doi.org/10.1146/annurev.biophys.37.032807.125832

Kalle, L., & Yoo, Y. (2002). Issues and Challenges in Ubiquitous Computing. *COMMUNICATIONS OF THE ACM*, *45*(12).

Khalil, A., & Connelly, K. (2005). Context-Aware Configuration: A Study on Improving Cell Phone Awareness. In *Modeling and Using Context* (Vol. 3554, pp. 197–209). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.2481&rep=rep1&type=pdf

Kotz, D., & Chen, G. (2000). A Survey of Context-Aware Mobile Computing Research. *Dartmouth Computer Science Technical Report*, *3755*, 1–16. Retrieved from https://secure.urkund.com/view/document/25858473-787506-348577/download

Kun, Y., Shumao, O., Manooch, A., & Nektarios, G. (2005). Policy-based model-driven engineering of pervasive services and the associated OSS. *BT Technology Journal*, *23*(3), 162–174. Retrieved from http://privatewww.essex.ac.uk/~kunyang/publications/2005/BTTJ2005.pdf

Lacerda, G. S., Ribeiro, V. G., & Silveira, S. R. (2013). Intervenção em TI de organização pública : um relato de experiência empregando, 14.

Loayza, A., Proaño, R., & Camacho, D. O. (2013). Aplicaciones sensibles al contexto. Tendencias actuales. (Current trends in context-aware applications.), 95–110. Retrieved from http://ingenieria.ute.edu.ec/enfoqueute/

Maatjes, N. C., Pires, L. F., Costa, P. D., & Sinderen, M. J. Van. (2007). Automated transformations from ECA rules to Jess. *Transformation*.

Manuel, V., & Coello, M. (2013). *Plataforma middleware para recomendadores contextuales en plataformas móviles*. Universidad Complutense de Madrid.

Map, P. (2004). BPEL Designer Tutorial Tutorial 1 : Developing a Hello World BPEL Process. *World*, 1–22.

Marini, E. (2012). El Modelo Cliente/Servidor.

McCarthy, J. (1993). Notes on formalizing context. *Science*, *13*, 555–560.

https://doi.org/10.1.1.35.8117

Miraoui, M., Tadj, C., & Amar, C. Ben. (2011). ARCHITECTURAL SURVEY OF CONTEXT-AWARE SYSTEMS IN PERVASIVE COMPUTING ENVIRONMENT, 9.

Molina, J. M., Corchado, J. M., & Bajo, J. (2008). Ubiquitous Computing for Mobile Environments. *Issues in Multi-Agent Systems: The AgentCities.ES Experience*, 33–57.

Mondragón, O. H., & Solarte, Z. M. A. (2004). *Arquitectura para la creación de Servicios ubicuos orientados a salud*. Bogota: Springer.

Morillo, L. M. S. (2011). Middleware para el desarrollo de aplicaciones ubicuas en dispositivos móviles.

Oracle. (2014). Java Remote Method Invocation (Java RMI). Retrieved November 28, 2016, from http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

Popkin, S. S. (2007). Modelado de Sistemas com UML, 24.

Poslad, S. (2009). *Ubiquitous Computing: Smart Devices, Environments and Interaction*.

Punie, Y. (2005). The future of Ambient Intelligence in Europe: The need for more everyday life. *Communications & Strategies*, *5*(141), 141–165.

Pwc.com. (2014). The Wearable Future, 50.

Quality&Programming. (2012). ¿Qué es un componente de software? Retrieved from http://qualityandprogramming.blogspot.pe/2012/04/que-es-un-componente-de-softwarepara-el.html

Raento, M., Oulasvirta, A., Petit, R., & Toivonen, H. (2005a). ContextPhone: A Prototyping Platform for Context Aware Mobile Applications. *IEEE Pervasive Computing*, *4*(2), 51–59. https://doi.org/10.1109/MPRV.2005.29

Ramos, S. I., & Lozano, M. D. (2000). *Ingeniería del software y bases de datos, tendencias actuales*. (Universidad de Castilla-La Mancha, Ed.). Ediciones de la Universidad de Castilla-La Mancha.

Red, B., & Green, B. (2012). Model-View-Controller. *Structure*, (Mvc), 125–143. https://doi.org/ISBN: 978-0-471-95869-7

Rizzo, I. (2011). *Documentación de Estilos Arquitectónicos en Sistemas Web Colaborativos Sensibles al Contexto*. Universidad Nacional De Rosario.

Rodríguez, S. S. V., & Holgado, J. a T. (2008). Servicios Sensibles al Contexto en Sistemas de Computación Ubicua. *II Simposio En Desarrollo de Software, Universidad de Granada*, 235–248.

Roessingh. (2000). Roessingh Research and Development. Retrieved from http://www.rrd.nl/

Sánchez, R. (2012). Aplicación de geo-localización Forns IGP iOS / Android. *Aplicaci´ on de Geo-Localizaci´ on Forns IGP*.

Sandia, L. (2013). the Rule Engine for the Java Platform. Retrieved from http://herzberg.ca.sandia.gov/jess/

Schilit, B., & York, N. (1995). Context-Aware Computing Applications, 85–90. https://doi.org/10.1109/WMCSA.1994.16

Sloman, M., Thomas, M., Sparck-Jones, K., Crowcroft, J., Kwiatkowska, M., Garner, P., … Chalmers, D. (2006). Discussion on Robin Milner's first computer journal lecture: Ubiquitous computing: Shall we understand it? *Computer Journal*, *49*(4), 390–399. Retrieved from http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/computer2006.pdf

Smith, Y., Rengifo, P., Mendoza, J. A., Dirley, E., & Correa, C. (2015). Desarrollo Dirigido por Modelos (MDD) en el Contexto Educativo Model-Driven Development (MDD) in the Educational Context. *Scientia et Technica Año XX*, *20*(2).

Steven, S. M. (2013). Effects of environmental context on human memory. *The SAGE Handbook of Applied Memory*, 162–182. Retrieved from http://tamuweb.tamu.edu/faculty/stevesmith/SmithMemory/SmithSageChapter.pdf

Tahuiton, M. J. (2011). *Arquitectura de Software para aplicaciones Web*. Centro de Investigación y dde Estudios Avanzados.

The Open Group Architecture Framework. (2015). Generalidades del marco de referencia de Arquitectura Empresarial de The Open Group. Retrieved from https://colombiadigital.net/actualidad/articulos-informativos/item/8163-que-es-togaf.html

Tom, Z. (2007). Embodiment and Context Proceedings of the `Workshop on Context. In *European Conference on Cognitive Science*. Manchester: [IEEE Xploe].

Tusell, F., Bárcena, M. J., Núñez, V., & González, C. (2012). Análisis Multivariante, 164.

Uso de sensores. (n.d.). Retrieved August 15, 2017, from https://leonardolopez29.wordpress.com/3er-grado/bimestre-5/tareas/uso-de-sensores/

Wargo, J. M. (2015). *PhoneGap Essentials- Building Cross- Platform Mobile apps*. *PhD Proposal* (Vol. 1). https://doi.org/10.1017/CBO9781107415324.004

Wegdam, M. (2005). AWARENESS: A project on Context AWARE mobile NEtworks and ServiceS. *Published in the Proceedings of the 14t H Mobile & Wireless Communications Summ It 2005, 19-23 June 2005, Dresden*.

Weiser, M. (1991a). The computer for the 21st century. *Sci Am*, *265*(3), 94–104. https://doi.org/10.1038/scientificamerican0991-94

Weiser, M. (1991b). The Computer for the 21st Century, *Vol. 256*, 94--104.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Commun. ACM*, *36*(7), 75–84. https://doi.org/10.1145/329124.329127

Yang, L. T., Syukur, E., & Loke, S. (2013). *Handbook on mobile and ubiquitous computing : status and perspective*. Taylor & Francis.

ANEXOS

Ficha técnica de los instrumentos a utilizar

01: ENCUESTA A APLICAR

JUICIO DE EXPERTO PARA VALIDAR LA PROPUESTA TECNOLOGICA “ARQUITECTURA DE SOFTWARE PARA DESARROLLO DE APLICACIONES SENSIBLE AL CONTEXTO”.

INSTRUCCIONES:

Coloque en cada casilla un aspa(X) correspondiente a la pregunta, según los criterios que a continuación se detallan. Las dimensiones a evaluar son: arquitectura, soporte, diseño, servicios, aplicaciones sensibles al contexto, contexto e interacción.

Dominio y Preguntas	Respuestas			Observación o recomendación
	3 Bueno	2 Regular	1 Deficiente	
ARQUITECTURA				
El modelo “ECA-DX” es aplicable para el desarrollo de aplicaciones sensibles al contexto.				
El Modelo “ECA-DX” proporciona un vocabulario para representar y compartir el conocimiento del contexto en un dominio informático omnipresente.				
El modelo “ECA-DX” basado en ontologías para la información, permite describir contextos semánticos de manera que es independiente del lenguaje de programación, sistema operativo o middleware subyacente.				
El presente modelo ayuda solucionar los problemas recurrentes asociados a la gestión de la				

información del contexto y reaccionar de forma proactiva a los cambios del contexto				
El modelo propuesto presenta una estructura por cadenas jerárquicas de fuentes de contexto y gestores.				
El modelo “ECA-DX” presenta una estructura de componentes para apoyar el diseño e implementación de aplicaciones.				
El modelo “ECA-DX” cubre el manejo de sensores, y el prototipado y evaluación del sistema resultante.				
SOPORTE Y DISEÑO				
El modelo propuesto ofrece soporte para la obtención de información del contexto de forma autónoma.				
Con el modelo “ECA-DX” el componente de motor de reglas es el núcleo de la arquitectura de controlador.				
Con el modelo “ECA-DX” los componentes pueden ser implementados en la parte superior de un motor de reglas de Jess, que es capaz de determinar cuando las condiciones son satisfechas por el conjunto actual de los datos disponibles de la memoria de trabajo.				
El modelo “ECA-DX” está diseñado con el fin de desacoplar los propósitos de las acciones, implementaciones y coordinar la composición de las mismas.				
Con el modelo “ECA-DX” los proveedores de diseño de aplicaciones y servicios son los actores responsables del desarrollo, comercialización y mantenimiento de las aplicaciones sensibles al contexto.				
Con el modelo “ECA-DX” el diseño del procesador de contexto se realiza desde una perspectiva de ingeniería.				

El modelo “ECA-DX” predomina las técnicas de interfaces multimodales donde se revela la preferencia de los usuarios por enfoques híbridos que combinen configuraciones automática y manuales.				
INTERACCIÓN				
El proveedor de servicios tiene como objetivo mantener una alta disponibilidad del controlador, procesador de contexto interno y componentes de la acción.				
Con el modelo “ECA-DX” la colaboración entre los componentes y procesadores de contexto permite la disponibilidad de información potencialmente más rica.				
Mediante el Modelo “ECA-DX” el componente controlador tiene por objeto controlar la flexibilidad, extensibilidad y capacidad de adaptación de la plataforma de manejo de contexto.				
Mediante el Modelo “ECA-DX” la activación de las reglas se produce en tiempo de ejecución de la plataforma.				
El modelo “ECA-DX” proporciona un vocabulario común y la comprensión de los principios de diseño.				
El modelo “ECA-DX” ayuda a la construcción de software complejo y heterogéneo				
El modelo “ECA-DX”, permite desarrollar aplicaciones sensibles, modulares y escalables				
El Modelo “ECA-DX” es capaz de predecir el tiempo de la ubicación del usuario mediante la observación de los movimientos.				
El modelo “ECA-DX” es capaz de crear información del contexto a base de diversas fuentes de información de manera flexible.				
Las herramientas para el desarrollo de aplicaciones sensibles al contexto son accesibles y				

estándares.				
Mediante el presente modelo “ECA-DX” la localización se realiza a través del GPS, sensor del bluetooth como sensores de contexto.				
Mediante el presente modelo “ECA-DX” para superar las inexactitudes de GPS, utiliza información de las actividades que realiza el usuario, la cual se pone en relación con la ubicación.				
La generación de librerías específicas, orientadas al manejo de ciertas tareas sensible al contexto, que puedan ser incluidas en una aplicación para facilitar su desarrollo es también una preocupación en este campo.				
Con el modelo propuesto los usuarios finales pueden mejorar su productividad en el trabajo y mejorar su calidad de vida mediante el uso de ciertos tipos de aplicaciones sensibles al contexto				
Con el modelo “ECA-DX” la comunicación se produce a través de las aplicaciones sin la interacción humana.				
A través del modelo “ECA-DX” los componentes del contexto como el procesador, es responsable de capturar el contexto del usuario a través de los puntos de interacción.				

Muchas gracias por su apoyo.

Grado Académico: _____ Nombres y Apellidos: _____ Firma: _____

02: SOLICITUD DIRIGIDA A EXPERTO

Sr. Ing. _____

Es grato dirigirme a Usted para manifestarle mi saludo cordial. Dada su experiencia profesional y méritos profesionales, le solicito su inapelable colaboración como experto para la validación del contenido de los ítems que conforma el instrumento (Anexo), que tiene como finalidad validar la investigación titulada: "*Arquitectura de Software para el desarrollo de aplicaciones sensibles al Contexto*", para obtener el grado académico de Doctor en Ingeniería de Sistemas.

Para efectuar la validación del instrumento, Usted deberá leer cuidadosamente cada enunciado y sus correspondientes alternativas de respuesta, en donde se pueda valorar de acuerdo a lo que ha podido verificar con la aplicación como prototipo.

Se agradece cualquier sugerencia respecto al contenido u otro aspecto que considere relevante para mejorar el mismo.

Atentamente.

_____

Mg. Orlando Iparraguirre Villanueva

Email:

Definición de términos

Rete: El algoritmo Rete es un algoritmo de reconocimiento de patrones eficientemente para implementar un sistema para producción de reglas. Fue creado por el Dr. Charles L. Forgy en la Carnegie Mellon University. Su primera referencia escrita data de 1974, y apareció de forma más detallada en su tesis doctoral en 1979 y un artículo científico de 1982. Rete es hoy en día la base de muchos sistemas expertos muy famosos, incluyendo CLIPS, Jess, Drools y Soar(Causa, 2009)

OCL: (Object Constraint Lenguage), es un lenguaje para descripción forma de expresiones en los modelos UML. Sus expresiones pueden representar invariantes, precondiciones, post-condiciones, inicializaciones, guardias, reglas de derivación, así como consultas a objetos para determinar sus condiciones de estado (Demuth, 2012)

MAD: Arquitectura dirigida por modelos, es un acercamiento al diseño de software, propuesto y patrocinado por el Object Management Group (OMG). MDA se ha concebido para dar soporte a la ingeniería dirigida a modelos de los sistemas de software.(Smith, Rengifo, Mendoza, Dirley, & Correa, 2015)

BPEL: Lenguaje de Ejecución de Procesos de Negocio con Servicios Web, BPEL, es un lenguaje de programación en gran escala generalmente se refiere al desarrollo de software de gran tamaño que involucra grandes procesos de desarrollo, evolución y mantenimiento. El desarrollo de BPEL nace de la necesidad de manejar lenguajes distintos entre la programación a gran escala y la programación detallada, ya que en su esencia ambos tipos de desarrollo requieren de distintos grados de comunicación con otros servicios(Map, 2004)

MVC: Es el patrón Modelo, Vista y Controlador (MVC) es el más extendido para el desarrollo de aplicaciones donde se deben manejar interfaces de usuarios, éste se centra en la separación de los datos o modelo, y la vista, mientras que el controlador es el encargado de relacionar a estos dos (Camarena, Trueba, Martínez, & López, 2012).

FRAMEWORK: es un término compuesto, en donde Frame se traduce como Marco, y Work como trabajo.

SOA: Arquitectura orientada a Servicios (SOA, siglas en inglés *Service Oriented Architecture*) es un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos.

INDSCAS: Ingeniería del dominio basado en calidad de software.

PARCTAB: Fue uno de los primeros sistemas desarrollados para ofrecer un marco modular sensible al contexto (Gellersen, Schmidt, & Beigl, 2001)

ECA: Event-Control-Action (Evento – Control - Acción).

CS: Representan cajas de fuentes de contexto.

CM: Representan instancias de gestores de contexto.

IP: Puntos de interacción que acepta las especificaciones de la regla ECA.

DL: Lenguaje de dominio.

TOKENS: Bloque primitivo de texto estructurado.

JVM: Máquina Virtual Java.

GUI: Es una interfaz gráfica, conocida también como GUI (del inglés Graphical user interface), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz

WORKFLOW: Flujo de trabajo (en inglés) es el estudio de los aspectos operacionales de una actividad de trabajo: como se estructuran las tareas, como se realizan, cual es s orden correlativo, como se sincronizan, como fluye la información que soporta las tareas y como se hace seguimiento al cumplimiento de las tareas.

WEARABLE: (del inglés wearable device) se conoce como tecnología vestible que describen aquellas prendas de vestir, y complementos que incorporan elementos tecnológicos, electrónicos, etc.(Pwc.com, 2014)

SYMBIAN: fue un sistema operativo propiedad de Nokia, y que en el pasado fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encontraban Nokia, Sony Mobile Communications, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc.

SENSOR: Es un objeto cuyo propósito es detectar eventos o cambios en su entorno y envía la información a la computadora que proporcione la salida correspondiente. Un sensor es un dispositivo que convierte los datos del mundo real (analógico).

SYSTEM: Sistema.

GEOLOCALIZACIÓN: Entendemos por geolocalización al conjunto de técnicas que permiten determinar la posición geográfica de un elemento (un ordenador, un teléfono móvil o cualquier dispositivo capaz de ser detectado) en el mundo real y hacer uso de esa información. Esta tecnología requiere de la perfecta sincronización entre hardware y software, es necesario un dispositivo con GPS o conexión a internet y un software que permita hacer uso de ellos en esta dirección(Sánchez, 2012)

IOS: es un sistema operativo móvil, originalmente desarrollado para iphone(iPhone OS), después se ha usado en dispositivos de iPod touch y el iPad.

ANDROID: es un sistema operativo basado en el núcleo Linux, fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets; y también para relojes inteligentes, televisores y automóviles.

HTML: sigla en inglés de HiperText Markup Language (lenguaje de marca de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.

SMS: servicio de mensajes cortos o servicio de mensajes simples, más conocido como SMS (por las siglas del inglés *Short Message Service*), es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos, conocidos como mensajes de texto.

GPS: Sistema de Posicionamiento Global, más conocido por sus siglas en inglés, *GPS* (siglas de *Global Positioning System*), es un sistema que permite determinar en toda la Tierra la posición de un objeto (una persona, un vehículo) con una precisión de hasta centímetros (si se utiliza GPS diferencial).

XML: siglas en inglés de *eXtensible Markup Language*, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por El World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

JDK: Java Development Kit, es una implementación para las plataformas java: edición estándar, Edición empresarial o Edición micro, realizado por la corporación de oracle

LISP: es una familia de lenguajes de programación de computadora de tipo multiparadigma con una larga historia y una sintaxis completamente entre paréntesis.

MULTIVARIANTE: es una técnica de análisis de proyección sobre variables latentes, tiene muchas ventajas sobre los métodos de regresión tradicionales: se puede utilizar la información de múltiples variables de entrada, aunque éstas no sean linealmente independientes(Tusell, Bárcena, Núñez, & González, 2012).