UNIVERSIDAD DE SEVILLA

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

Middleware para el desarrollo de aplicaciones ubicuas en dispositivos móviles

---

Memoria de Tesis Doctoral presentada por

D. Luis Miguel Soria Morillo

para la obtención del grado de Doctor en Informática,

dirigida por los doctores:

D. Juan Antonio Ortega Ramírez

y **D. Luis González Abril**

Sevilla, junio de 2011

A mis padres, los mejores
amigos que siempre tendré

y a Rosa, por alumbrar el
camino con la luz de su corazón

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a mis directores de Tesis, Dr. D. Juan Antonio Ortega Ramírez y Dr. D. Luis González Abril, por toda la ayuda, confianza y aliento. Del mismo modo, a Francisco Velasco por transmitir toda su experiencia de la mejor forma, con una sonrisa. Así también a Juan Antonio Álvarez, por darme la oportunidad de escribir esta tesis y apoyarme para ello.

Destacar también el apoyo de mis compañeros y amigos del Departamento de Lenguajes y Sistemas Informáticos, especialmente a Ismael Cuadrado, Miguel Ángel Álvarez, Alejandro Fernández, José Ignacio Sánchez, Isabel Ramos y a todos en general, por hacerme sentir parte de la gran familia del departamento.

Asimismo, mi agradecimiento a los compañeros del colectivo ARCA, que año tras año me han aportado nuevas y fructíferas ideas durante nuestras

Quisiera trasladar, como no, un agradecimiento muy especial a mi familia y amigos, que siempre me han dado su cariño cuando más lo necesitaba, que han sido el hermano que nunca tuve. Especialmente a Daniel, Juan y Julia, que tantas alegrías están trayendo a nuestras vidas.

A todos, Gracias.

RESUMEN

En este documento se muestra una arquitectura software para la gestión y desarrollo de aplicaciones susceptibles de ser disparadas por contexto en dispositivos móviles, entendiendo por contexto toda la información externa al dispositivo que representa el estado en el que se encuentra el usuario en un momento determinado. Dicha arquitectura pretende facilitar 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. Las metodologías de desarrollo y los frameworks empleados hoy en día para la elaboración de sistemas software, presentan varias limitaciones que hacen del desarrollo de sistemas basados en contexto una tarea demasiado compleja y repetitiva, debido en gran medida a la falta de comunicación entre las aplicaciones. A pesar de todo, el número de aplicaciones que trabajan con datos del entorno del usuario se han disparado en los últimos años. Esto se ha producido principalmente por la incorporación de sensores que permiten obtener dichos datos en los teléfonos móviles de última generación, los cuales forman parte cada día más de la vida cotidiana de los individuos.

El diseño de la arquitectura propuesta permite no sólo desarrollar aplicaciones contextuales autónomas, sino también permite la interconexión y comunicación entre varias aplicaciones. Esta característica hace posible que varias aplicaciones compartan información en tiempo real y que sea posible la petición de información entre sistemas que están siendo ejecutados en el propio dispositivo. Como se verá a lo largo de este documento, esto es posible gracias al motor de eventos contextuales e informativos que han sido desarrollados en esta tesis doctoral.

Además del propio middleware, sobre él se han construido dos sistemas ubicuos que aprovechan al máximo las características del entorno desarrollado. Por un lado un sistema que permitirá reconocer con una elevada precisión la actividad física llevada a cabo por una persona y, por otro lado, una herramienta para la detección de salidas a exteriores.

En cuanto al sistema de reconocimiento de actividades, ha sido desarrollado un método innovador y eficiente para la detección de actividades físicas que un determinado usuario está realizando. Esto se hace gracias al proceso de reconocimiento desarrollado y a los sistemas de sensores que los dispositivos móviles de última generación integran en su arquitectura. Gracias a este método, el sistema es capaz de conocer con una elevada precisión la actividad física que se está llevando a cabo de una manera poco intrusiva, ya que se emplea el propio dispositivo móvil del usuario para realizar el procesamiento de la información. La inclusión de este método en el middleware de aplicaciones ubicuas, permite ampliar la gama de contextos que pueden ser manejados por el sistema.

Respecto al desarrollo de un sistema de detección de salidas a exteriores, indicar que este permite determinar el momento en el que un usuario ha entrado a interiores y, lo que es aún más importante, cuándo sale del recinto. Mediante este sistema es posible reducir de una manera drástica el consumo de baterías en aplicaciones que precisan de un posicionamiento continuo del usuario. Como consecuencia de este ahorro energético, se aumenta el tiempo de uso entre recargas consecutivas, incrementando a su vez la comodidad del usuario y mejorando la experiencia en este tipo de aplicaciones, además de reducir la emisión de CO_2 a la atmósfera como consecuencia del ahorro energético ofrecido por el sistema.

Para la realización de esta tesis se aplicó una metodología básica que siguió el procedimiento deductivo para la elección del tema. El planteamiento del problema se basó en el desarrollo de un middleware para la generación y la comunicación de sistemas móviles basados en contexto así como en la obtención de una nueva aproximación a la detección de actividades físicas llevadas a cabo por un usuario y a la detección de salidas mediante la aplicación del dicho middleware. Los objetivos de la tesis son, por un lado facilitar el desarrollo de aplicaciones contextuales debido a la gran demanda que en el presente y en el futuro tendrán este tipo de sistemas; y por otro lado, aumentar la gama de información contextual capaz de ser obtenida mediante el uso de los sensores integrados en los últimos modelos de dispositivos móviles.

ABSTRACT

In this thesis project is presented a software architecture for manage and develop applications that can be triggered by the context in mobile devices, understanding the context as any external information that represents the state in which the user is in a given time, like the activity that users are carrying out, the temperature of their environments or the place in which users are located. This architecture aims to facilitate communication between other applications, in order to manage the user environment information in an effective and safe way. Development methodologies and middleware used today for developing software systems have several limitations that make the development of systems based on context too complex. Nevertheless, at present, numbers of applications that work with data from the user environment have increased exponentially in recent years. This fact has occurred largely due to sensors embedded in the latest generation of mobile devices that can obtain such data. In addition to an application development middleware based on context in mobile devices, will be presented different techniques for acquiring information about the execution context of applications.

The proposed design allows not only developing autonomous contextual applications, but also allows the interconnection and communication between some applications. This feature allows sharing information by multiple applications in real time and allows the submission of information between systems that are running on the device itself. As discussed throughout this document, this is possible thanks to the event-and-contextual information engine that the author has developed in this document.

Finally, has been developed an innovative and efficient method for the detection of physical activities that a particular user is carrying out. This is done thanks to sensor systems that next generation mobile devices integrated into their architectures. Using this method, the system is able to know the activity carried out in a least intrusive as possible, due to the information processing is made in the user's own mobile device. This method associated with the development middleware, can extend the range of contexts that can be handled by the system.

To carry out this work we applied a basic methodology followed the deductive procedure for choosing the topic. The problem statement was based on the development of a middleware for the generation and communication of context-based mobile systems as well as obtaining a new approach to the detection of physical activities carried out by a user. The objectives of the author's research was to provide contextual application development due to high demand at present and in future will have this type of system.

To conclude this summary, it is noteworthy that the main findings of the investigation were the new method of communication between mobile applications based on a core-runtime designed by the author as well as an effective and energy efficient method of determining physical activity in a user, the most significant being the minimum intrusion and the short period of training the system.

SIGLAS, ABREVIATURAS Y ACRÓNIMOS

GPS: Global Positioning System

SOA: Service Oriented Architectura

OWL: Ontology Web Language

WSDL: Web Services Description Language

XML: Extensible Markup Language

BPMN: Business Process Modeling Notation

RDF: Resource Description Framework

E-OTD: Diferencia temporal observada en el enlace

TOA: Diferencia temporal de llegada

WiFi: Wireless Fidelity

GSM: Sistema Global de Comunicaciones Móviles

MLP: Perceptrón multicapa (*Multi Layer Perceptron*)

FFT: Transformada rápida de Fourier *(Fast Fourier Transform)*

RAW: Datos obtenidos directamente a partir de la fuente sin ningún procesamiento

RNA: Red neuronal artificial

HCI: Interacción Persona-Computador

CONTENIDO

ÍNDICE DE GRÁFICAS Y FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

- Mark Weiser -

No cabe duda que desde que en 1944 surgiera el primer computador programable de la historia, el llamado Z3 creado por Konrad Zuse, la computación y su impacto en la sociedad han sufrido profundos cambios. Sin ir más lejos, realizando una comparativa entre el ancestro de los sistemas informáticos actuales y cualquier dispositivo móvil presente en la vida cotidiana de millones de usuarios, podremos observar esta evolución. El computador Z3 estaba formado por 2200 relés electromecánicos, los cuales le dotaban de un peso de una tonelada. Poseía una frecuencia de reloj de 5Hz y una longitud de palabra de 22 bits, lo que hacía que el tiempo medio empleado en realizar una operación de división o multiplicación fuera de unos 3 segundos. Sin embargo, hoy en día un dispositivo móvil genérico está constituido por miles de minúsculos componentes eléctricos, lo cual hace que su peso esté en torno a 130 gramos. Estos componentes dotan a los dispositivos de una frecuencia de operación de hasta 1Ghz, es decir, unas 200 millones de veces más veloz que la frecuencia a la que trabajaba el Z3. En cuanto a almacenamiento, el panorama no dista demasiado de lo expuesto anteriormente, ya que el Z3 era capaz de almacenar 700 bits de datos, frente a los varios gigabytes de almacenamiento de información que un dispositivo móvil puede llegar a ofrecer en un espacio mucho más reducido. Esta diferencia de tamaño se puede observar en la *Figura 1*, la cual muestra el Z3 y un dispositivo móvil de última generación. Asimismo dicha reducción de tamaño ha hecho posible la movilidad, lo cual se ve reflejado en que según estudios realizados en el año 2009, un 67% de la población posee dispositivos móviles y las previsiones son que este porcentaje siga creciendo hasta llegar a un 80% en 2014.

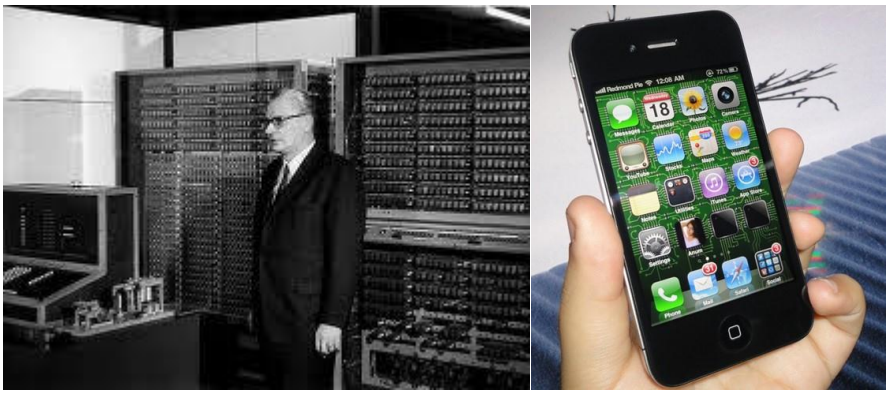


Figura 1: A la izquierda el Z3 diseñado por Konrad Zuse. A la derecha el iPhone 4, un dispositivo móvil de última generación

A pesar de todo, la evolución que los dispositivos electrónicos han experimentado desde entonces, ha hecho posible que cada día más, la informática forme parte de la vida de las personas de todo el planeta. De hecho, muchas de las actividades que realizamos de manera cotidiana, como enviar un correo electrónico, navegar por internet o realizar una insignificante llamada telefónica, no serían posibles sin la ayuda de estos dispositivos. Además, el avance de la tecnología en la construcción de computadores, ha dado lugar a que en la última generación de dispositivos se haya vislumbrado una

serie de necesidades y aplicaciones potenciales que en generaciones anteriores no eran viables. Tales necesidades están estrechamente relacionadas con la manera en la que el usuario y la computadora se comunican y relacionan.

La gran gama de sensores que han sido incluidos en los dispositivos móviles de última generación tales como el iPhone 4, Google Nexus One, HTC Desire HD y otros, han proporcionado un nuevo método de interacción persona-computador, caracterizado precisamente por la creciente independencia del usuario y la automatización de los procesos en base a determinados criterios. Desde que en 1963 Ivan Sutherland marcara el inicio de la informática gráfica (Sutherland, 1963), la interacción persona-computador ha sufrido un desarrollo espectacular, comparable al que anteriormente tuvo lugar con las computadoras. La interacción entre humanos y dispositivos electrónicos es especialmente necesaria en muchos campos y no es la intención de esta tesis profundizar en ellas (Alan Dix, 2004), (International Symposium on Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN '94, 1994), (Improving Human Computer Interaction through Spoken Natural Language, 2007). Sin embargo sí es interesante reseñar la continua evolución de estas técnicas en pro de la facilidad del uso por parte de los humanos. La tecnología ha permitido que esta interacción sea cada vez más natural para el usuario.

En este ámbito de la interacción persona-computador (HCI), cabe destacar que las aplicaciones requieren cada vez de un número más elevado de información. La mayoría de las veces la información debe ser proporcionada manualmente por el usuario, lo cual presenta una serie de problemas entre los que destacan la incomodidad del usuario debido a la elevada cantidad de información a introducir y a los complejos métodos de introducción de texto en cierto tipo de dispositivos, especialmente en dispositivos móviles principalmente por su reducido tamaño (Investigating text input methods for mobile phones, 2006) (An evaluation of mobile phone text input methods, 2002). Además está presente la subjetividad inherente que existe cuando es el propio usuario el que introduce los datos (The acceptance of information, its subjective cost and the measurement of distortion, 1982) (Research on the dynamic comprehensive evaluation based on information integrated, 2010). Por estos y otros motivos que se expondrán a lo largo de esta tesis, la computación del futuro, la que está llegando hoy día a la mayoría de los hogares, está adquiriendo un sesgo hacia la automatización de la obtención de la información. Esto se lleva aa cabo principalmente mediante la adquisición de datos desde sensores embebidos en los propios dispositivos de una manera transparente al usuario en mayor o menor medida.

En el contexto tecnológico actual, se hacen continuas referencias al término computación ubicua y sistemas contextuales, sobre el cual se sientan las bases de esta tesis doctoral. Sin embargo, a pesar de su extendido uso, el concepto no es sencillo de explicar y de hecho existen diferentes interpretaciones del mismo, por lo que en primer lugar comenzaremos explicando su origen. El término de computación ubicua no es nuevo en absoluto, sino que fue Mark Weiser en 1991 quien introdujo por primera vez este concepto (The Computer for the 21st Century, 1991). En los trabajos de Weiser se daban las primeras pinceladas de lo que años más tarde sería una auténtica revolución en el campo de las tecnologías de la información, ya que dichos sistemas estarían íntimamente ligados a la vida cotidiana del usuario. Dejando temas morales aparte, es indudable que la idea de Weiser ha cambiado la forma de relacionarse (Spiderweb: A social mobile network, 2010), de trabajar, de aprender (Ubiquitous E-learning System for Dynamic Mini-courseware Assemblying and Delivering to Mobile Terminals, 2009), de interactuar con los dispositivos (Travel assistance device: utilising global positioning system-enabled mobile phones to aid transit riders with special needs, 2010) e incluso de divertirse de la sociedad (Game-Based Learning with Ubiquitous Technologies, 2009).

Conseguir que la próxima generación de aplicaciones haga uso de esta tecnología no es una tarea fácil, ya que los paradigmas de programación y las arquitecturas actuales presentan una serie de inconvenientes a la hora de dar soporte a todos estos métodos automáticos para la captura de la información. Muchos de estos problemas pasan por la reutilización de la funcionalidad, puesto que cada vez más los sistemas presentan una serie de relaciones basadas en el uso de los recursos de información. En este sentido ciertos sistemas proporcionarán la información de partida a otros que se ejecutan en el mismo dispositivo o incluso en otros dispositivos conectados con éste. De esta forma, los sistemas pueden generar nueva información a partir de los datos procedentes de los sensores y del propio usuario, a la vez que otros sistemas hacen uso de la información obtenida anteriormente para crear nuevas funcionalidades.

El contenido de esta tesis doctoral pretende fomentar el desarrollo de estas aplicaciones gracias a la creación de nuevas herramientas que facilitan su creación. Concretamente se presentará una arquitectura conceptual y un middleware de desarrollo que hace posible no sólo una generación más sencilla, estructurada, segura y eficiente de aplicaciones que hacen uso de la información contextual, sino un método que permite la interconexión de las aplicaciones que emplean estas tecnologías, favoreciendo la compartición de la información y la reutilización en general. El resultado es un desarrollo claramente más ordenado, reusable, eficaz y sobre todo sencillo de aplicaciones contextuales, además de la posibilidad de establecer una comunicación estandarizada entre los servicios que se están siendo ejecutados por el usuario en el dispositivo.

Es obvio que el facilitar la comunicación y la creación de aplicaciones basadas en contexto no es suficiente para elaborar sistemas realmente útiles para el usuario. Además de esta tarea es necesario definir y diseñar una amplia gama de contextos posibles que sean capaces de determinar diferentes parámetros del propio usuario o de su entorno. Históricamente estos contextos se centraban en el posicionamiento del usuario o su localización, aunque cada vez son más los datos que estos sistemas son capaces de obtener automáticamente mediante el uso de sensores o a través de la conexión con otras fuentes externas de información, como por ejemplo redes sociales o servicios web. En este sentido, como caso práctico del uso del middleware y generar nueva información contextual de base, durante esta tesis se presentan dos aplicaciones avanzadas que hacen uso de este middleware para generar la funcionalidad. Por un lado se expone un método innovador y eficiente para la detección de la actividad física llevada a cabo por un usuario gracias al estudio de la señal procedente de los propios sensores del dispositivo y por otro, un sistema capaz de detectar salidas y entradas en lugares cerrados lo cual, como se verá más adelante, reducirá de manera significativa el gasto energético de los sistemas de posicionamiento actuales.

El hecho de detectar la actividad que un usuario está realizando es de especial interés en sistemas de monitorización o control de usuarios. Por medio de este sistema, el dispositivo es capaz de reconocer la actividad que está llevando a cabo el usuario de manera autónoma, sin ninguna interacción con el sujeto. Esto hace que sea aplicable en numerosos campos como por ejemplo en la medicina, aplicado a la detección de caídas o patrones irregulares de movimiento, o en el ámbito de la seguridad laboral. En este contexto existen varios trabajos previos que están estrechamente relacionados con el expuesto en este trabajo (Activity Recognition from Accelerometer Data on a Mobile Phone, 2009), (SmartBuckle: human activity recognition using a 3-axis accelerometer and a wearable camera, 2009) y (Activity Recognition using Wearable Sensors for Elder Care, 2008). Sin embargo, se verá a lo largo de la tesis del autor los beneficios que aporta el nuevo método obtenido frente a los existentes previamente.

El proceso de reconocimiento de actividades se llevará a cabo gracias a los datos proporcionados por los sensores de acelerometría, los cuales se encuentran embebidos en la mayoría de dispositivos móviles del mercado actual. Un acelerómetro no es más que un instrumento electrónico cuyo fin es medir las aceleraciones producidas sobre el cuerpo en el que se encuentre instalado. En la actualidad es posible construir acelerómetros de tres ejes en un solo chip de silicio, lo cual hace que el nivel de precisión y versatilidad de los mismos sean muy elevados, de ahí su incorporación masiva en dispositivos móviles.

Como se comentaba anteriormente, esta información es muy empleada y de gran utilidad para sistemas de e-Health cuyo objetivo es por ejemplo, monitorizar y controlar la actividad que llevan a cabo personas de edad elevada[1], con enfermedades que afecten a la conducta o usuarios que presenten movilidad restringida (A home-based care model for outpatient cardiac rehabilitation based on mobile technologies, 2009). Sin embargo, el abanico de sistemas que podrían hacer uso de esta información es extremadamente amplio. Ejemplo de estos sistemas serían reproductores musicales portátiles capaces de reproducir una música acorde con la actividad que estamos realizando, un sistema de monitorización continua de actividades, sistemas de posicionamiento en interiores basados en la longitud recorrida a partir de las actividades realizadas (Indoor wayfinding: developing a functional interface for individuals with cognitive impairments., 2008), (SurroundSense : Mobile Phone Localization via Ambience Fingerprinting, 2009) o televisores interactivos que muestren un resumen de la actividad física realizada por el usuario a lo largo del día.

Por otro lado, el posicionamiento y localización de personas son aplicaciones de actualidad en el ámbito de los sistemas ubicuos, a partir de las cuales se están desarrollando multitud de sistemas de información. El posicionamiento personal consiste en la determinación del lugar donde se encuentra un determinado usuario en términos de coordenadas geográficas (latitud y longitud) o bien en términos semánticos (descripción del lugar). Llevando a la ubicuidad esta aplicación, tendríamos un sistema capaz de determinar la posición de un usuario en cualquier lugar y en cualquier momento. Sin embargo, esto presenta una serie de problemas derivados del alto consumo energético producido por los sensores de posición.

Otro contexto muy extendido es la localización de un determinado usuario o dispositivo. En la mayoría de estos trabajos, el posicionamiento se lleva a cabo mediante el uso del *Sistema de posicionamiento global (GPS)*, el cual permite conocer las coordenadas geográficas en las que se encuentra el sujeto. Sin embargo, el principal inconveniente de este sistema es la inutilidad cuando el sensor de posicionamiento se encuentra en un lugar techado. Esto se debe a que la señal empleada por el sensor para la localización proviene de una serie de satélites sin los cuales el sistema no resulta útil. Este inconveniente es intrínseco a la técnica, sin embargo existe otro problema más al emplear este sistema, que es el consumo energético.

Los sensores de posicionamiento *GPS* consumen una gran cantidad de energía al obtener la posición y este consumo aumenta aún más si no es posible obtener señal debido a que el usuario se encuentra bajo un lugar techado. Este consumo se traduce en una menor duración de las baterías del dispositivo, obligando al usuario a realizar recargas continuas del terminal. Dicho problema se presentan en la práctica totalidad de sistemas que emplean esta técnica de posicionamiento, lo cual influye negativamente en la efectividad de la localización continua y en la experiencia del usuario con el sistema. Sería de gran utilidad desarrollar un sistema que de manera automática permita activar o desactivar el sensor de posicionamiento en función de si el usuario se encuentra o no en un lugar

[1] http://www.keruve.com

techado. Con esto se podría aumentar de manera drástica el tiempo de vida útil del dispositivo entre recargas consecutivas y por otro lado, reducir el consumo energético derivado de la recarga excesiva de las baterías del dispositivo.

Durante el presente documento, se presenta un sistema que permite conocer no sólo si el usuario se encuentra o no en un lugar techado, sino además capaz de detectar en qué momento se produce la salida al exterior, pudiendo así iniciar el sistema de posicionamiento *GPS* evitando el uso innecesario del mismo. Este sistema se diseña en base a las especificaciones del middleware expuesto en este documento, poniendo así de manifiesto la utilidad de esta herramienta.

En general se puede observar que el potencial de las aplicaciones ubicuas es insospechado y cada día los avances en este campo son formidables. No podemos olvidar que la computación ubicua surge de la necesidad de facilitar la vida a los usuarios mediante la capacidad de asesorar, gestionar y actuar de manera automática en base al entorno que rodea al individuo y a las propias necesidades del usuario. Todo ello hace que la tecnología pase de ser una mera herramienta empleada por el sujeto a un elemento transparente y totalmente personalizado al usuario que mejora la calidad de vida de las personas mediante el uso de los últimos avances tecnológicos.

PROPUESTA

Lo que conduce y arrastra al mundo no son las máquinas sino las ideas.

- Víctor Hugo -

Debido en gran medida al incremento de aplicaciones que emplean información contextual para alcanzar sus objetivos, se hace insostenible un desarrollo artesano, desestructurado e independiente de cualquier aplicación que use estas tecnologías. De hecho, si así fuera, estaríamos descartando la posibilidad de conexión con aplicaciones que hayan sido realizadas por terceros y en general, con cualquier sistema que ofrezca una información adicional a la manejada por la aplicación desarrollada. Por este motivo, en la actualidad se hace imprescindible establecer una arquitectura de desarrollo de aplicaciones basadas en contexto, que permita solventar el problema citado. En este trabajo se presentará una arquitectura basada en capas que posibilite a cualquier aplicación que siga dicha arquitectura, la comunicación y el intercambio de información con otras aplicaciones. Por otro lado, el sistema permitirá el uso de información generada por terceros de una manera transparente, externalizando toda la gestión de la información dejándola en manos del middleware.

Este middleware será el encargado de notificar los cambios en la información contextual provenientes de los diferentes proveedores de contexto. Por tanto, dicho sistema gestor deberá tener información de las aplicaciones que se están ejecutando, de las aplicaciones disponibles y de las distintas conexiones y relaciones existentes entre dichas aplicaciones y los proveedores del contexto. Todo esto será definido más adelante y será estudiado como un servicio de información, que será el núcleo del sistema desarrollado.

Puesto que las aplicaciones no sólo necesitan compartir información, sino conocer el estado de otras aplicaciones, el sistema desarrollado debe tener la capacidad de disparar nuevas aplicaciones en base a las condiciones presentes en el contexto de ejecución. Es decir, será posible ejecutar diferentes aplicaciones cuando se den unas determinadas condiciones sobre el contexto del usuario (humedad, temperatura, actividad, luminosidad, posición, etc.). De esta forma, el núcleo del sistema que

desarrollaremos también tendrá la necesidad de almacenar los contextos de disparo de cada una de las aplicaciones. Esta funcionalidad es conocida como disparo de aplicaciones por contexto (*Context triggered applications).*

Debido al gran interés que presenta en los sistemas ubicuos actuales, se desarrollará un sistema capaz de detectar la actividad física que un determinado usuario está realizando en un momento dado. Aunque como dijimos anteriormente, esto es un tema donde la literatura existente es relativamente extensa, el sistema se ha desarrollado con unos fines muy específicos: conseguir la mayor precisión posible con el menor coste energético. Debemos tener en cuenta que el principal problema de los dispositivos móviles actuales no es la potencia, ni la memoria, ni siquiera la complejidad de uso, sino la energía que consumen y el tiempo de vida de sus baterías. Con el sistema desarrollado conseguimos por un lado reducir el coste energético y por otro, realizar una composición de varios métodos de análisis, obtención, filtrado y clasificación de señales con resultados óptimos para la clasificación de la actividad física.

Como método de validación de los resultados obtenidos mediante las diferentes técnicas empleadas para el reconocimiento de actividades, se ha desarrollado un protocolo que mediante la colaboración de otro usuario, permite al sistema obtener automáticamente el índice de calidad del método, entendiendo por calidad la asociación entre fiabilidad, eficiencia, coste temporal y precisión del método bajo estudio.

Además será necesario realizar una comparación entre las distintas arquitecturas de ejecución que pueden ser empleadas para el reconocimiento de actividades. En concreto se estudiarán las predicciones realizadas de manera off-line, es decir, en una computadora de forma externa al dispositivo, en un ambiente semiautónomo mediante la conexión entre el dispositivo móvil y un servidor que tendrá la principal carga computacional del método de reconocimiento y, en último lugar, un modelo totalmente autónomo en el que todas las labores de reconocimiento se realizan en el propio dispositivo.

Seguidamente se desarrollará un método capaz de detectar la salida a exteriores de un usuario, empleando de nuevo el potencial que nos brinda el middleware presentado. En este caso se intentará reducir el coste energético provocado por el posicionamiento continuo de un individuo

Para finalizar serán expuestas una serie de aplicaciones que muestran el potencial del middleware, el núcleo, el reconocedor de actividades y el sistema de detección de salidas, los cuales permitirán a las aplicaciones desarrolladas trabajar en común beneficiándose del sistema de notificaciones, alertas y disparos por contexto que el middleware ofrece.

ESCENARIO DE APLICACIÓN

A continuación se expone un escenario de uso del middleware de desarrollo de aplicaciones basadas en contexto y los dos proveedores de contextos para reconocimiento de actividades y detección de salidas. Para ello analizaremos la interacción de Daniel, un usuario de dispositivo móvil de última generación que ha adquirido una aplicación desarrollada con el middleware expuesto.

Hoy es sábado, son las 10:30 de la mañana y hace un precioso día veraniego. Habitualmente a esta hora, Daniel acude al parque próximo a su domicilio para correr unos minutos, y hoy no iba a ser menos. Nuestro amigo a punto de salir de casa, coge su dispositivo móvil de última generación y

los auriculares para escuchar música. Mientras baja las escaleras de su casa para ir hacia el parque, el detector de salidas a exteriores. En cuanto Daniel sobrepasa el portal que da acceso a la calle, el sistema reconoce una salida y activa el dispositivo GPS, permitiendo obtener la localización de Daniel. Dado que es aficionado a la música, lleva su dispositivo repleto de las últimas canciones publicadas en las listas de éxitos. Una vez en el parque, Daniel conecta los auriculares al dispositivo y comienza a correr a un ritmo de calentamiento. Gracias al sistema de reconocimiento de actividad física, el dispositivo detecta que Daniel está corriendo a un ritmo pausado, por lo que comunica al reproductor que el tipo de música a reproducir debe ser relajada. Sin embargo, tras 4 minutos de calentamiento, Daniel sube el ritmo de la carrera. El sistema de reconocimiento de actividades detecta este evento, por lo que lanza un mensaje al reproductor para que comience la reproducción de unos temas con más ritmo. Tras 30 minutos de carrera, nuestro amigo decide volver a casa y darse un baño. De nuevo, al entrar en el hogar, el dispositivo detecta la entrada a interiores, por lo que el sistema desconecta el dispositivo GPS y marca el lugar como punto de entrada.

Tras tomar el baño, nuestro amigo se sienta unos minutos a descansar en el sofá del salón, justo enfrente de la televisión, que se decide a encender inmediatamente. Daniel selecciona el widget que le permite mostrar las actividades llevadas a cabo y muestra en la pantalla el recorrido llevado a cabo en el parque. Además, observa la velocidad a la que ha realizado la actividad y la actividad concreta que ha llevado a cabo.

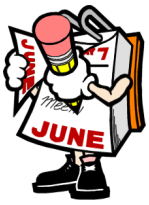
Nuestro amigo se dispone a salir de su casa tras haber cogido el dispositivo móvil que dejó sobre la mesa minutos antes. El sistema detecta que es sábado, justo el día en el cual Daniel, junto a su novia Mar suelen ir al cine para ver las películas estrenadas el día anterior. Por ello, tras subirse al coche, el módulo de detección de destinos frecuentes determina que el destino más probable de Daniel es la casa de su novia Mar, así que envía una notificación al sistema de navegación para que guíe a Daniel hacia el destino determinado. Una vez juntos, Mar y Daniel se dirigen al cine. De nuevo, el sistema de detección de destinos lanza el sistema de navegación, esta vez teniendo el centro comercial favorito de la pareja como destino.

Ya en el cine, Mar y Daniel se encuentran con que justo ese día se estrena una de las películas más esperadas de los últimos años, por lo que las colas en la taquilla son demasiado largas. El sistema de posicionamiento y el etiquetado de lugares, detectan que el usuario, en este caso Daniel, se encuentra en el cine, por lo que hace que se lance la aplicación de visualización y compra de entradas. Gracias a la conexión a internet podremos visualizar la cartelera correspondiente a este cine sin que esto suponga una interacción compleja para Daniel, ya que basándose en el contexto, el dispositivo es capaz de predecir las necesidades del usuario. Sin embargo, nuestros amigos no tienen claro aún la película que desean ver, ya que les han recomendado dos largometrajes que aún están en cartelera. Para ello, Daniel consulta a través de su dispositivo móvil una sencilla interfaz que recoge las opiniones de las distintas películas presentes en el cine donde se encuentran. Una vez que se han decantado por la película con mejores críticas por parte de los usuarios, Mar y Daniel se ahorran la espera de la taquilla tradicional, ya que el sistema de compra on-line les ha evitado pasar por taquilla.

Gracias al sistema Bluetooth, el proveedor de contexto de proximidad de usuarios detecta que cerca de la posición de Daniel se encuentra su amigo Juan. Automáticamente el sistema alerta a Daniel sobre este evento y muestra los datos del Juan en la pantalla de su móvil, ofreciendo la

posibilidad de realizar una llamada o escribir un correo para que nuestro amigo conozca nuestra posición y podamos encontrarnos con él. Gracias al sistema de detección de personas Daniel y Juan se encuentran y Juan conoce a Mar, la novia de su amigo Daniel.

Una vez compradas las entradas, justo antes de entrar en la sala, Mar se encuentra con su hermana Rosa pero, dado que no emplea de manera habitual el dispositivo Bluetooth, lo tiene desactivado. Esto hace que el subsistema encargado de detectar proximidad mediante el sistema Bluetooth no encuentre ninguna coincidencia. Sin embargo, Mar comienza a hablar con su hermana de lo que harán el próximo fin de semana. Gracias al módulo de detección de usuarios mediantes huellas de sonido, el dispositivo de Daniel reconoce la voz de su cuñada Rosa y detecta su proximidad.



Mediante un proceso de búsqueda en la agenda, el dispositivo detecta que Daniel tiene una tarea pendiente con ella Rosa, así que al detectar que está junto a ella, le muestra una notificación con esta tarea. Debido a que la película no empieza hasta dentro de media hora, Daniel y Rosa deciden hablar del tema que tenían pendiente, por lo que la agenda de Daniel se libera de la cita con Rosa que más tarde decidirá para qué emplearlo.

Por último, Daniel y Mar entran en la sala para comenzar a ver la película elegida, así que el sistema de detección de actividades reconoce que Daniel está sentado, por lo que detiene inmediatamente el sistema de posicionamiento GPS, ya que no será necesario para determinar la posición.

Tras finalizar la película, Daniel y Mar vuelven a casa y deciden poner a cargar sus dispositivos móviles. La batería del dispositivo de Mar se agotó hace unos minutos, mientras el de Daniel aún tiene un 30% de la carga, debido al ahorro producido por el uso del sistema de detección de salidas y desconexión GPS.

En el escenario de aplicación expuesto se pone de manifiesto la necesidad de interconectar aplicaciones, datos obtenidos por distintos sistemas así como lanzar nuevas aplicaciones que no estaban en ejecución. Sin embargo, estas acciones requeridas no pueden ser realizadas usando los sistemas actuales y las arquitecturas empleadas hoy en día, ya que el hecho de compartir información entre aplicaciones no es una labor sencilla. En gran parte, la dificultad de la compartición de información se puede entender como un problema de integración de información (Information integration, 1998), (The design and implementation of information integration framework on equipment domain based on ontology, 2010). Sin embargo el problema va más allá, debido a que la información debe estar disponible de manera inmediata a las diferentes aplicaciones, en el mismo instante en el que la información ha sido generada por la aplicación fuente. Además, la integración de toda esta lógica en un dispositivo móvil, hace que el sistema sea poco intrusivo y que la cantidad de información susceptible de ser recopilada por el sistema sea más elevada que si se realizara en otro tipo de dispositivos. Además, el hecho de que se trate de un sistema ubicuo hace indispensable que el sistema de obtención de la información esté lo más próximo al usuario y durante el mayor tiempo posible.

Dado que todo el sistema se ejecutará en un dispositivo energéticamente autónomo y con una duración de baterías limitada, es imprescindible que el consumo del núcleo del sistema sea lo menor posible. Además, el funcionamiento de los proveedores de contexto debe ser preciso, ya que un fallo en este sistema podría provocar que una aplicación se disparara sin tener que hacerlo o que la

información compartida con las aplicaciones sea incorrecta. Por tanto, debemos llegar a un equilibrio entre eficiencia y eficacia que, como se muestra en este documento, se ha llevado a cabo mediante el desarrollo del sistema de detección de actividad física.

JUSTIFICACIÓN

Por todo lo anterior, el objetivo de esta tesis es resolver el problema que los desarrolladores de aplicaciones basadas en contexto se encuentran a la hora de estructurar y comunicar sus aplicaciones, así como la constante necesidad de obtener el contexto mediante el procesamiento de las señales provenientes de los diferentes sensores presentes en el dispositivo. La arquitectura que puede presentar una aplicación *tradicional* dista mucho de las arquitecturas empleadas para desarrollar aplicaciones ubicuas. En el primer caso, la fuente de información suele provenir del propio usuario, de una base de datos o de un servidor que provee dichos datos. Sin embargo, cuando nos enfrentamos al desarrollo y al diseño de una aplicación contextual, nos damos cuenta de que la información proviene de una serie de sensores y procesos que adaptan la información que obtienen del exterior. Aunque a priori podría parecer que las diferencias son mínimas, nada más lejos de la realidad.

A continuación son enumeradas una serie de características que hacen de los sistemas contextuales un conjunto característico de aplicaciones:

- *Dependencia del hardware concreto de los sensores.* Dado que la mayoría de la información de la aplicación debe ser obtenida a partir de los sensores del dispositivo, la aplicación debe soportar el uso de diferentes sensores. Dichos sensores pueden tener un diferente grado de calibración o incluso un sistema de medida totalmente distinto, por lo que debe ser la aplicación la encargada de adaptar estos datos para poder trabajar con ellos en todos los casos.

- *Necesidad de un procesamiento inmediato de la información.* La velocidad con la que se procesa un determinado tipo de información en Sistemas Ubicuos puede llegar a ser de vital importancia. Imaginemos que debemos detectar caídas en un anciano sometido a una monitorización a través de un sistema de reconocimiento de actividades. En este caso, la velocidad con que se detecte la caída puede llegar a suponer la diferencia entre la vida y la muerte del usuario, por lo que es inviable el hecho de procesar los datos de manera manual.

- *Sistemas con alto grado de autonomía.* Precisamente es una consecuencia del punto anterior. Si el procesamiento de la información debe ser lo más rápido posible, el sistema debe poder ser autónomo, dicho de otra forma, el usuario no debe participar en ningún momento en la ejecución del sistema, a no ser que éste, de manera justificada, necesite información por parte del propio usuario. En general, cualquier acción que requiera intervención por parte del usuario, inevitablemente conllevará un retraso en el procesamiento.

- *Cualquier dato debe ser procesado previamente a su presentación.* En general, la información obtenida por los sensores no es suficiente para poder ser usada en un contexto real de aplicación, por lo que dicha información debe ser procesada por la propia aplicación o por otro sistema con este fin. De una manera u otra, el sistema debe transformar los datos de los sensores a una información entendible tanto por la aplicación como por el usuario, y esto en la mayoría de los casos no es una tarea sencilla.

- *Gran cantidad de datos anómalos debido al ruido o al margen de error de los sensores.* Los sistemas ubicuos emplean normalmente diferentes tipos de información que, en la mayoría de los casos, son idénticas entre sí. Es más, en muchos casos las aplicaciones realizan el mismo procesamiento de los datos y posteriormente emplean los datos para fines distintos. Este proceso de filtrado de datos anómalos y detección de errores en la información de los sensores suele ser común a la mayoría de las aplicaciones, así que una posible aproximación para la reducción de la complejidad del problema sería posibilitar el envío de información post-procesada entre aplicaciones.

- *Necesidad de comunicación entre sistemas ubicuos.* En una aplicación *tradicional*, sería impensable empezar desde la base y definir un nuevo sistema de gestión de ventanas y un nuevo protocolo de envío de información por internet, a no ser que las necesidades del sistema sean extremadamente específicas. Sin embargo, la reutilización y la comunicación entre aplicaciones contextuales no están regidas por ningún estándar ni por ninguna arquitectura. Dicho de otro modo, cada desarrollador fabrica la aplicación a medida implementando toda la funcionalidad desde cero, sin apoyo en ningún servicio externo al propio sistema operativo. Tomando como partida el punto anterior, la evolución lógica sería diseñar una arquitectura en forma de middleware de ejecución, que permita la comunicación, el envío de información y la gestión automática de la misma con el fin de permitir la interconexión entre ellas.

- *Ejecución en una plataforma móvil con recursos limitados.* Prácticamente la totalidad de sistemas ubicuos se ejecutan en dispositivos móviles debido a la propia definición de sistema ubicuo de portabilidad, transparencia y ubicuidad. Por ello, debemos desarrollar cualquier sistema ubicuo teniendo en cuenta las restricciones físicas de estas plataformas. En especial, se debe evitar a cualquier precio el procesamiento duplicado de la información, dado que los recursos consumidos aumentarán linealmente mientras que la productividad permanece inalterada.

- *Problemas de ahorro energético.* Como consecuencia del punto anterior, una mayor necesidad de procesamiento implica inevitablemente un aumento del consumo energético. Esto podría llegar a ser inadmisible debido a que impediría la ejecución de un conjunto de aplicaciones en ejecución constante, ya que la batería del dispositivo podría llegar a durar tan sólo unas horas. Por este motivo, siempre que se realiza una aplicación de perfil contextual y ubicuo, debe ser pensada y elaborada teniendo presente en todo momento el ahorro computacional y con el menor número de conexiones inalámbricas posible, puesto que es otro de los motivos de consumo elevado de baterías.

- *Alta probabilidad de fallos debido al elevado tiempo de ejecución constante de los sistemas*. Las aplicaciones ubicuas son ejecutadas generalmente en segundo plano y durante un tiempo elevado, ya que el objetivo en la mayoría de los casos es obtener información sobre el perfil del usuario basándose en el medio que lo rodea. Esto hace que un pequeño error en la aplicación, como por ejemplo un fallo en la gestión de la memoria, cause un fallo irreparable en el sistema, por lo que provocaría el cierre de la aplicación. Mediante una elaboración ordenada y estructurada del software no nos podemos asegurar de que no existan fallos, aunque sí podemos disminuir el número y el impacto de los mismos. Además, si unimos esto a la conectividad entre aplicaciones y a la compartición de la información, podríamos emplear sistemas altamente testeados para realizar operaciones que se necesiten en el sistema a

desarrollar. Esto reduciría de manera apreciable el coste y el tiempo de desarrollo, así como un aumento de la calidad de las aplicaciones generadas.

- *Seguridad de los datos personales.* Los datos que se manejan en las aplicaciones ubicuas son generalmente de perfil personal, ya que son obtenidos directamente de conductas, patrones o a partir del entorno del usuario que porta el dispositivo. Por este motivo es rigurosamente necesario asegurar la confidencialidad de la información manejada por las aplicaciones. Obviamente no es posible asegurar esta confidencialidad totalmente, ya que serán las aplicaciones en última instancia las que deban proteger la información manejada. Sin embargo, es posible incrementar la seguridad del sistema si la información manejada por las aplicaciones se transmite de manera encriptada.

Como podemos comprobar, son varios los problemas que nos encontramos asociados a las características de los sistemas ubicuos basados en contexto. Sin embargo, a lo largo de este trabajo se darán soluciones a algunos de estos problemas, haciendo así el desarrollo de aplicaciones y la comunicación entre ellas una tarea más sencilla y eficaz.

OBJETIVOS

Tras haber expuesto los problemas a los que un desarrollador se enfrenta cuando comienza a desarrollar una aplicación que hace uso de información contextual, es hora de presentar cómo este trabajo puede ayudar en este aspecto y cuáles serán los objetivos fundamentales que serán perseguidos a lo largo del mismo.

Puesto que el trabajo está dividido en dos grandes bloques, por un lado el análisis y diseño de una arquitectura para el desarrollo de aplicaciones basadas en contexto y por otro lado la generación de un nuevo método para el reconocimiento de actividades físicas, los objetivos estarán por tanto divididos también en ambos bloques. Los primeros objetivos que se exponen a continuación serán alcanzados en el bloque de la arquitectura, mientras que los restantes serán cubiertos en el bloque de reconocimiento de actividades y detección de salidas.

- Estudiar y analizar las necesidades de las aplicaciones ubicuas cuyos orígenes de datos provienen de la información contextual del usuario.

- Establecer una arquitectura de desarrollo de aplicaciones basada en capas que permita un desarrollo ordenado, eficaz y estándar para la implementación de sistemas basados en contexto.

- Definir un núcleo del sistema así como un protocolo de mensajes que permita la comunicación entre aplicaciones que sigan la arquitectura definida en este trabajo, haciendo posible el envío y recepción de información procesada o generada por otras aplicaciones distintas a la que se está desarrollando. Además, dicho protocolo de comunicación permitirá disminuir el tiempo de desarrollo de aplicaciones empleando la reutilización de procesos.

- Proponer un método para el registro y descubrimiento de aplicaciones basado en funcionalidad contextual, capaz de conseguir la conciencia del sistema a cerca de la funcionalidad que pueden ofrecer las aplicaciones instaladas y en ejecución.

- Definir una técnica que permita el lanzamiento automático de aplicaciones en base a determinadas condiciones del entorno del usuario.

- Definir un nuevo método para el reconocimiento de actividad basado en datos procedentes del sensor de acelerometría en un dispositivo móvil de última generación basado en el estudio cualitativo de las variables involucradas.

- Establecer un método poco costoso computacionalmente para la eliminación de variables irrelevantes de un conjunto de variables cuantitativas.

- Establecer un método de discretización y selección de características a partir de datos de acelerometría agrupados en ventanas temporales.

- Realizar un modelado del problema del reconocimiento de actividades físicas para el filtrado a posteriori de errores de clasificación.

- Llevar a cabo un sistema que permita reducir el coste energético del posicionamiento ininterrumpido a través de GPS mediante la detección de salidas a exteriores.

METODOLOGÍA Y PLAN DE TRABAJO

La ciencia es producto de acciones razonadas y sistemáticas que permiten descubrir nuevos elementos esclarecedores y significativos en la realidad

La metodología seguida durante todo el proceso de investigación y creación de esta tesis, ha sido la habitual del método científico con algunas excepciones que comentaremos a continuación. En primer lugar podríamos resumir los pasos en los siguientes puntos:

- *Formulación formal del problema.* En esta fase se definirán los objetivos de la investigación y se analizarán los problemas que se encuentran en la actualidad relacionados con la temática, explicándolos con total claridad y precisión. En este caso el problema está relacionado con el desarrollo de sistemas ubicuos en dispositivos móviles, la estructura de las aplicaciones basadas en contexto y la problemática del reconocimiento de actividades.

- *Análisis del estado del arte.* El objetivo de esta fase es obtener una base de información sobre la temática a tratar, lo suficientemente amplia como para conocer todas las propuestas existentes en la investigación realizada por la comunidad científica. Con ello se obtendrá un conocimiento de base que permitirá identificar carencias y problemas en las soluciones obtenidas en otros trabajos, lo cual servirá de base para el comienzo de la investigación.

- *Diseño de la investigación.* Aunque existe suficiente literatura relacionada con el tema que se ha elegido para la tesis, no deja de ser un campo muy novedoso y en el que aún queda mucho por explorar. Por este motivo, se realizara en primer lugar un estudio exploratorio con el fin de familiarizarnos con el tema de la investigación, lo que más tarde dará lugar a un estudio más detallado sobre algunos problemas concretos detectados. Tras esto, se realizará un estudio explicativo, cuyo objetivo es dar una solución a los problemas detectados durante la fase anterior. Sin embargo, dada la peculiar distribución de este proyecto, durante los

primeros capítulos serán estudiados aspectos más tecnológicos y arquitectónicos, por lo que no tiene sentido hablar de un análisis explicativo que permita conocer la relación entre variables involucradas en el sistema. Sin embargo, durante los últimos capítulos, se realizará un proceso más matemático que tecnológico para el reconocimiento de actividades. En tal caso sí que hablaremos de estudios correlaciónales y descriptivos.

- *Desarrollo de la propuesta.* Una vez planificada la etapa de diseño de la investigación y obtenido el conocimiento suficiente sobre el campo que se tratará en el proyecto, es hora de desarrollar una propuesta a los problemas detectados. Esta es sin duda la etapa en la que se invertirá más tiempo y esfuerzo, dado que será necesario el estudio de varias técnicas de diferentes disciplinas para dar solución a los problemas identificados.

- *Análisis y validación.* Una vez propuesta una solución a los problemas que se marcaron como objetivos de la investigación, es necesario analizar dicha solución así como someterla a un proceso de validación que asegure que la solución propuesta es adecuada y los resultados son los adecuados.

CONTRIBUCIONES

Este trabajo ha dado como resultado una serie de técnicas novedosas en el campo de la computación ubicua y del disparo de aplicaciones por contexto. En concreto, se ha centrado en la obtención de un middleware de comunicación entre sistemas basados en contexto, así como en una técnica innovadora para el reconocimiento de actividades físicas y la detección de salidas. Aunque existen varias contribuciones originales, cabe destacar las siguientes:

- Se ha desarrollado un middleware capaz de comunicar diferentes aplicaciones basadas en contexto de una manera completa y eficiente.

- Una descripción detallada de las capacidades y características que debe tener un determinado sistema para que pueda ser disparado en base a unas determinadas condiciones del entorno impuestas por el propio desarrollador.

- Se ha desarrollado una técnica capaz de determinar el nivel de dependencia estadística de varias variables entre sí, con el fin de eliminar aquellas que menos cantidad de información aporten al sistema, con la consiguiente reducción de la complejidad del problema a resolver.

- Un método eficaz y autónomo, capaz de ser ejecutado en un dispositivo móvil, que permite monitorizar la actividad física que está realizando una determinada persona.

- Un abanico de métodos de reconocimiento de actividades físicas que pueden ser seleccionados en base a las necesidades y características del sistema donde vayan a ser empleados.

- Se ha desarrollado un método para la detección de salidas y entradas en lugares cerrados con el fin de ahorrar energía en aplicaciones que requieren un posicionamiento continuo del usuario basado en GPS.

La mejor estructura no garantizará los resultados ni el rendimiento. Pero la estructura equivocada es una garantía de fracaso.

- Peter Drucker -

El documento actual presenta la estructura descrita a continuación:

- **Capítulo 2: Introducción a los sistemas** ubicuos. En este capítulo serán expuestos los conceptos básicos que se manejan en la computación ubicua y que serán tratados a lo largo de este documento. Además se realizará un estudio sobre los sistemas basados en contexto en general, desde sus orígenes en la historia hasta la aparición de los dispositivos móviles inteligentes y la revolución del mundo de los sistemas ubicuos. Por último, en el segundo capítulo se introducirá el concepto de *aplicaciones lanzadas por contexto*, sobre el cual se fundamenta la justificación de este trabajo.

- **Capítulo 3: Middleware para aplicaciones basadas en contexto**. Debido al auge de las aplicaciones con información de origen contextual y a la expansión masiva esperada en los próximos años, se hace imprescindible el uso de una arquitectura eficaz para el desarrollo de dichas aplicaciones. En este capítulo se expondrá la solución que se da a este problema y se presentará una arquitectura diseñada para este propósito. Además de esto se hará un estudio pormenorizado de la representación y almacenamiento de datos contextuales con el fin de unificar la sintaxis mediante el uso de una clasificación específica. Por otro lado las aplicaciones actuales y muy probablemente las aplicaciones futuras, obtienen los datos directamente del contexto del usuario, es decir, el usuario cada vez necesita interactuar en menor medida con el dispositivo. Esto provoca que el dispositivo deba procesar una gran cantidad de datos procedentes de los sensores para cada una de las aplicaciones que los necesitan. En este capítulo se presentará un middleware que permite la comunicación entre aplicaciones, de manera que los datos puedan ser generados por una aplicación y consultados por otra sin necesidad de conexión física entre las aplicaciones. Por último se describirán en detalle cada una de las capas lógicas que componen el middleware y la interconexión entre las mismas.

- **Capítulo 4: Reconocimiento de actividades.** En este capítulo se presentará un método para el reconocimiento de actividades físicas llevado a cabo por un usuario. A lo largo del mismo se presentarán diferentes técnicas de selección de características singulares, generación de intervalos, discretización de datos y métodos de aprendizaje necesarios para llevar a cabo la solución propuesta. Además, se estudiarán diferentes métodos para llevar a cabo el reconocimiento, ofreciendo una comparativa para determinar la eficiencia y eficacia de cada uno de ellos.

- **Capítulo 5: Sistema de detección de salidas.** A lo largo del quinto capítulo se abordarán todos los conceptos necesarios para el desarrollo de un sistema de detección de salidas a exteriores. Además serán expuestos una serie de métodos innovadores desarrollados en esta tesis con el fin de minimizar el coste computacional y aumentar la precisión del método de detección de salidas, mejorando de esta manera los diferentes métodos presentes en la bibliografía.

- **Capítulo 6: Conclusiones y trabajos futuros.** En este capítulo se muestran las conclusiones de la tesis expuesta en este documento y los diferentes trabajos que serán realizado en los próximos años para la realización de la tesis.

- **Anexo I.** En el primer anexo se expone el contenido de los diferentes ficheros *XML* que contienen la información contextual manejada por el middleware desarrollado.

- **Anexo II**. En este anexo se describirán las diferentes interfaces empleadas para la definición de la funcionalidad de la arquitectura así como para el intercambio de información entre los diferentes módulos que componen el sistema.

- **Anexo III**. En el tercer anexo se presentan los trabajos de investigación llevados a cabo por el autor durante la elaboración de la tesis que se presenta en este documento.

CAPÍTULO 2: INTRODUCCIÓN A LOS SISTEMAS UBICUOS

Algún día en cualquier parte, en cualquier lugar indefectiblemente te encontrarás a ti mismo, y ésa, sólo ésa, puede ser la más feliz o la más amarga de tus horas.

- Pablo Neruda -

A medida que ha transcurrido el tiempo, la manera de interactuar entre los humanos y las computadoras ha cambiado de una manera trascendental. Cada vez más, los usuarios pasan de estar frente a una pantalla de ordenador, con el teclado, ratón y demás periféricos, a manejar dispositivos móviles, tecnología embebida, televisores de última generación, reproductores de MP3 e incluso elementos que ni siquiera saben que están presentes, como por ejemplo las etiquetas RFID (Smart Parking Applications Using RFID Technology, 2007). Todo esto no ha sido más que un paso adelante a lo que Mark Weiser predijo años atrás, en 1991, cuando afirmó que en unos años, la tecnología estaría tan cerca del propio usuario, que pasaría inadvertida. Por tanto, podríamos decir que el siglo actual podría convertirse sin lugar a duda en el siglo de la computación pervasiva y los sistemas ubicuos (Pervasive Computing: Implications, Opportunities and Challenges for the Society, 2006)

Según la Wikipedia®, la enciclopedia on-line más importante de todos los tiempos, se entiende por computación ubicua *la integración de la informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados*[1]. Como se puede observar en la definición anterior, se trata de un concepto muy general y por tanto, aplicable a multitud de campos de investigación actuales. Ejemplos de dichos campos de investigación son el reconocimiento de voz (Pervasive speech recognition, 2004), reconocimiento de imágenes (Recognizable-Image Selection for Fingerprint Recognition With a Mobile-Device Camera, 2008) o el posicionamiento en interiores (Modeling of indoor positioning systems based on location fingerprinting, 2004).

Además, la computación ubicua no es tan sólo aplicable a los campos citados anteriormente, sino que su evolución se nutre de otras ramas de la ciencia y la tecnología, lo que la hace convertirse en un campo de investigación transversal. Esta visión es compartida por multitud de autores, como así se demostró en el año 2000 en Bristol, donde en el simposio internacional *Handheld and Ubiquitous Computing* se reunieron por primera vez investigadores de una gran cantidad de campos, para discutir exclusivamente sobre la computación ubicua (Handheld and Ubiquitous Computing, 2000). Más tarde se celebrarían nuevos simposios y congresos, como las conferencias internacionales del campo de la computación ubicua por excelencia, Ubicomp[2], que en 2010 celebró su duodécima edición.

Tampoco cabe duda de que la computación ubicua ha revolucionado el mundo de las comunicaciones, como es el caso del *blogging*, las redes sociales y los sistemas colaborativos en tiempo real. En concreto, las redes sociales geoposicionadas[3] han sido posibles gracias a la inclusión de dispositivos receptores de señal GPS en los propios dispositivos móviles.

[1] http://es.wikipedia.org/wiki/Computación_ubicua
[2] http://www.ubicomp2010.org
[3] http://www.google.com/latitude

Sin embargo, el campo de los sistemas ubicuos y la computación basada en contexto no ha hecho más que empezar, y aún es necesario un proceso de adaptación para conseguir una madurez en el desarrollo de este tipo de aplicaciones semejante a la que han alcanzado las aplicaciones tradicionales de escritorio, web o móviles. Precisamente para detectar las necesidades de las que carece la computación ubicua actual, es necesario echar la vista atrás y entender la evolución que han sufrido este tipo de sistemas y la idea de ubicuidad en general. Sólo una vez realizado esto, podremos ser capaces de mirar hacia el futuro y comprender, estudiar y analizar las necesidades de los sistemas ubicuos y lo que es aún más importante, de los propios usuarios que harán uso de ellos.

EVOLUCIÓN HISTÓRICA

En la historia de los computadores y los sistemas de información, la mayoría de la literatura y los expertos en computación diferencian cuatro grandes etapas. En primer lugar se encuentra la época comprendida entre los años 1943 y 1980, aunque previamente se puede identificar otra época correspondiente a la creación de las primeras computadoras. La segunda generación de computadoras se caracterizaba por poseer unos tamaños gigantescos y dar servicio a varios terminales a partir de una solo unidad de procesamiento. Podríamos decir que se trataban de grandes servidores, a los que se conectaban multitud de teclados y pantallas, a través de los cuales los usuarios de la computadora podían realizar los cálculos necesarios.

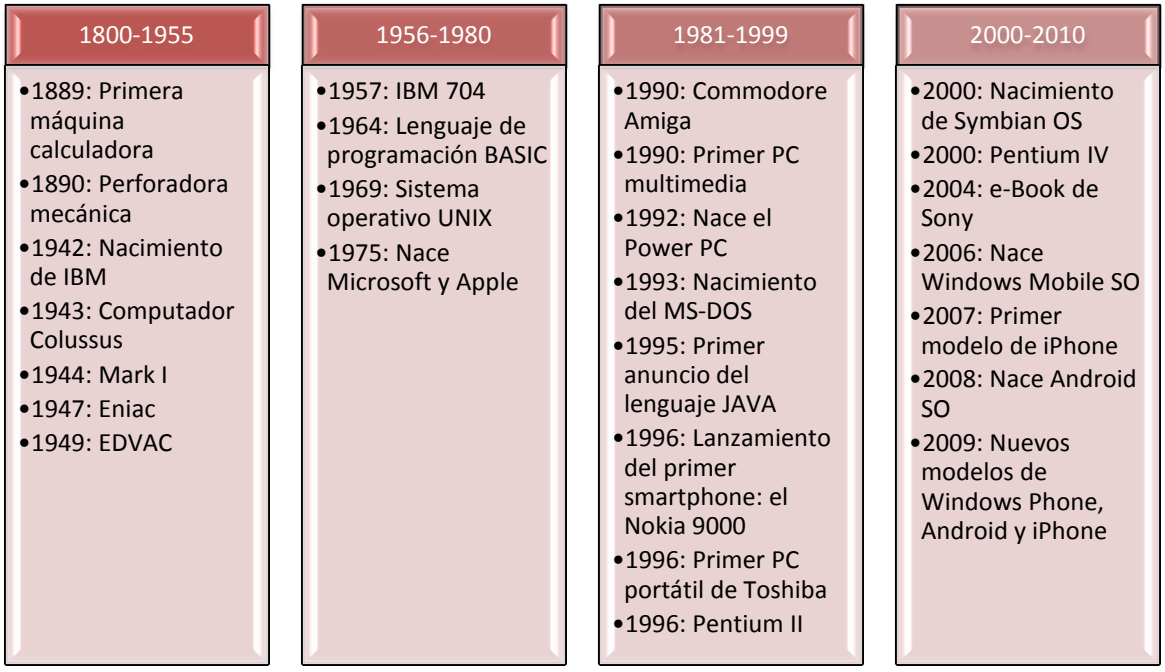


Figura 2: Evolución histórica de los computadores

Tras esta primera era, en 1980 comienza la segunda era de la computación, marcada por el nacimiento del primer ordenador personal (PC) de la historia. Este hito permitió acercar la computación a la vida de las personas, ya que se trataban de máquinas autónomas, con memoria propia y capacidad de almacenamiento y cálculo sin precedentes hasta la fecha. Además, su relativa facilidad para ser transportada y su pequeño tamaño comparado con las computadoras de la generación anterior, permitió que se pudiera convertir en un artilugio doméstico y no sólo empresarial. Es decir, pasamos de tener un supercomputador en una sala específica para él con unas condiciones ambientales muy controladas, a tener un PC en un hogar donde las condiciones no tendrían por qué ser óptimas. Aunque lo más importante de toda esta evolución es la relación que se empezaría a crear a partir de los años 90

entre el par persona/computador. En este momento se dejaría entrever la siguiente generación de la computación, en la que persona y computador estarían estrechamente relacionados.

Sin embargo, en la tercera era de los sistemas computacionales, la relación que se daba en la primera era entre varias personas y un ordenador se invierte. A partir de 1991 y hasta nuestros días, cada vez son más los usuarios que no sólo tienen un dispositivo, sino varios computadores personales. Además, inevitablemente el concepto de computador ya no es simplemente un gran mainframe en una sala de un sistema de clústeres de una gran empresa, sino un pequeño dispositivo móvil que el usuario lleva en su bolsillo, un reproductor de música capaz de conectarse a un servicio a través de tecnología 3G o un ordenador ultra portátil con conexión a alta velocidad a Internet.

Sin embargo, hay un personaje ampliamente conocido en el mundo de la computación ubicua y la informática en general que marca el inicio de la tercera era: Mark Weiser. Weiser, mientras dirigía el Laboratorio de Ciencia Computacional (CSL) de Xerox PARC en 1988, fue el fundador del término *computación ubicua*. Sin duda, Weiser fue un visionario y fue capaz de predecir casi con una década de antelación, cómo sería la tecnología del futuro. La principal característica que tendría, sería su ubicuidad, es decir, la tecnología estaría en todos los lugares, embebida en objetos que forman parte de la vida cotidiana de los usuarios que los usan (The Computer for the 21st Century, 1991). Además del concepto, Weiser en su laboratorio de Xerox PARC y su grupo, se dedicaron a realizar distintos objetos de tamaños muy diferentes, los cuales tenían una característica en común: eran *inteligentes*. Dicho de otra forma, en éstos objetos se encontraban localizados una serie de sensores que permitían obtener diferentes características del estado del propio objeto y del usuario que los manipulaba. Obviamente, los objetos desarrollados por Weiser y su equipo no eran ninguna revolución tecnológica, pero la filosofía parecía clara, el mejor computador es aquel que no se ve, que forma parte del usuario sin ser percibido.

La computación ubicua[4] por tanto, se basa en la capacidad de un dispositivo de detectar acciones que un usuario realiza o identificar cambios en el contexto de ejecución del sistema, actuando en consecuencia para realizar una determinada acción de manera autónoma o adaptar el medio que rodea al usuario en base a sus propios beneficios.

El siguiente hito histórico que marca un avance computacional es en 1992. En este año, IBM lanza su primer teléfono móvil inteligente, *Simón*. Fue comercializado en 1993 y entre sus características se encontraban la capacidad de realizar llamadas telefónicas, libreta de direcciones, calculadora, correo electrónico, juegos y una pantalla táctil. En este avanzado dispositivo para la época, que hoy en día podríamos catalogar como un teléfono de gama baja, se empezaba a vislumbrar la idea de Weiser. Un objeto pequeño, llevable, sensorizado y comunicado con el resto de dispositivos que permitía al usuario hacer uso de él de una manera sencilla.

A partir de este momento, el lanzamiento de los primeros teléfonos móviles inteligentes de Nokia revolucionó el campo de la computación ubicua, debido a la posibilidad de instalar aplicaciones en un dispositivo de tamaño reducido que podía ser transportado cómodamente por un usuario. En este momento surgió lo que hoy se conoce como computación móvil, que fue la antecesora de lo que hoy conocemos como UbiComp.

En 1997 IEEE crea el estándar 802.11, el cual define el uso de los dos niveles inferiores de la arquitectura OSI (capas física y de enlace de datos), especificando sus normas de funcionamiento en

[4] También denotada como ubicomp en gran parte de la literatura especializada

una WLAN. Esto hizo que los dispositivos pudieran comenzar a comunicarse sin ningún tipo de cables de una forma masiva, ya que la mayoría de fabricantes en torno al año 1999 adoptó este estándar como base para el desarrollo de sus sistemas de comunicación inalámbrica. De esta forma tan sutil se produjo el cambio entre computación móvil y computación ubicua, en la cual el usuario pasaba de tener la tecnología como una herramienta a estar inmersa en ella.

Con la llegada del siglo XXI se sucedieron los dispositivos inalámbricos que compartían información automáticamente entre sí, como los equipos portátiles, las PDA y los móviles de última generación. Gracias a ellos los usuarios se sentían cada vez más conectados a la sociedad de la información y compartían sus propias experiencias con el resto de usuarios de la comunidad. Este hecho denominado sistemas sociales se integró a la perfección en la definición de computación ubicua, pasando a ser una tecnología unipersonal para convertirse en una tecnología comunitaria. Ahora la información de un usuario podía ser compartida por miles y miles de individuos, lo que dio origen a un problema que los sistemas desarrollados hoy día siguen intentando combatir: la seguridad y la confidencialidad de la información.

Sin ir más lejos, según un estudio realizado en junio de 2010 por la *ITU*[5] *(International Telecommunication Union)*, el 90% de la población mundial posee cobertura a una red móvil. Esto se traduce en que el número de dispositivos móviles ubicuos, es decir, conectados a la red y obteniendo datos de manera automática en base a ciertos criterios, ha crecido en casi un 100% entre los años 2006 y 2010. Más concretamente, en 2006 el número de dispositivos ubicuos a nivel mundial era de 2.7 billones, mientras que en el año 2010 este número ascendió a 5.9 billones. En la *Figura 3* se puede observar la evolución experimentada por la computación móvil ubicua en los últimos años. Es especialmente interesante que en el año 2006, el número de usuarios con plataformas móviles ubicuas (con conexión a la red) desde países en vías de desarrollo representaba el 40% del total de los accesos, mientras que en 2010 este número se vio decrementado hasta llegar al 27%.

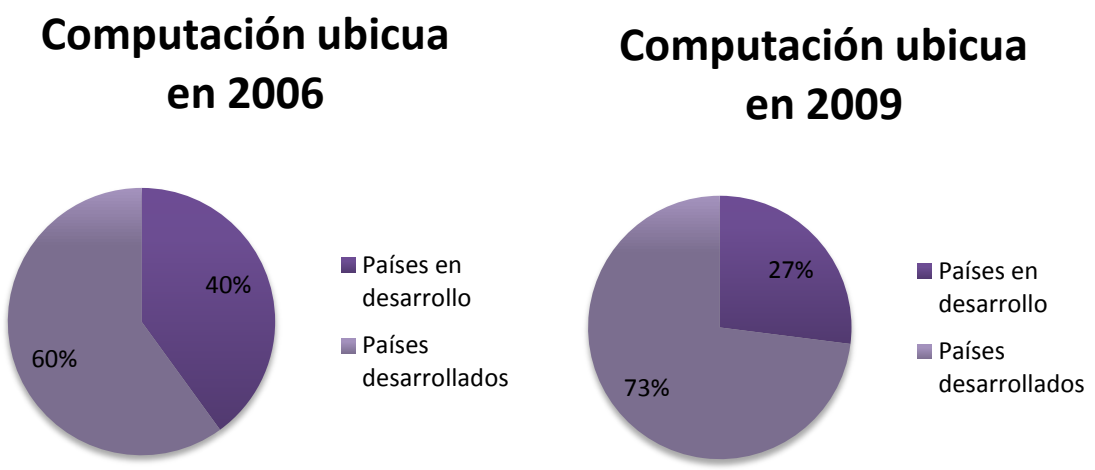


Figura 3: Evolución de la computación ubicua entre 2006 (2.7 billones de usuarios) y 2010 (5.9 billones de usuarios)

[5] Publicado en el World Telecommunication Development Conference 2010

Es indiscutible que la tecnología forma actualmente parte de la vida de millones de personas repartidas entre la totalidad de países desarrollados del planeta. Además, en cierta manera, todos dependemos de ella para sobrevivir y aumentar nuestra calidad de vida, ya que gracias a esta tecnología somos capaces de tener calefacción en nuestros hogares, de ver la televisión, de enviar correos electrónicos a nuestros contactos o de realizar llamadas telefónicas cuando deseamos hablar con algún familiar. Sin embargo, existe una tecnología que cada día se hace más y más imprescindible en la sociedad del siglo XXI, y son precisamente los sistemas ubicuos. No es necesario esperar para observar como las posibilidades de la tecnología actual han cautivado a la sociedad actual y la han hecho convertirse en una sociedad tecnológicamente dependiente.

Durante varios años, los ordenadores personales supusieron una revolución social, ya que unido a la aparición de internet, cualquier usuario era capaz de tener accesible cualquier tipo de información que buscara. No obstante, el uso del PC no es trivial e incluso actualmente, multitud de individuos no usan dicho aparato debido a los problemas y dificultades que se encuentran al trabajar con él. El objetivo que se consigue gracias a la computación ubicua es permitir al usuario que preste total atención a la tarea que debe realizar, en lugar de tener que prestar atención al manejo de la tecnología que debe ayudarlo a realizar dicha tarea. Mediante el uso de sistemas ubicuos, la sociedad en general se ve ayudada por tecnologías que ni siquiera sabe que existen, ya que pasan totalmente desapercibidas entre los usuarios.

El impacto de los sistemas basados en contexto en la sociedad podría asemejarse perfectamente al efecto que produjo la electricidad. Su integración con la vida cotidiana está tan lograda que actualmente se encuentra en todos los lugares que podamos imaginar: oculta bajo el suelo, por las paredes de una vivienda, en los automóviles e incluso encerradas en baterías más pequeñas que un botón. De igual forma, los sistemas ubicuos han ido entrando en la sociedad y gracias a ellos se han producido numerosos avances en áreas como la docencia con los sistemas de e-Learning (Game-Based Learning with Ubiquitous Technologies, 2009), en la salud con los sistemas de e-Health (Toward a Personal Health Society in Cardiology, 2010) y en otros muchos campos.

Mención aparte merece la forma en la que la sociedad actual se comunica y se relaciona. Aplicaciones como Facebook®, Twiter®, LinkedIn® y otras redes sociales, han revolucionado el mundo de la comunicación entre usuarios. Además, el desarrollo de aplicaciones basadas en dichas redes sociales para dispositivos móviles, ha llevado a que se produzca una comunicación constante y distribuida entre usuarios, creando nuevas relaciones entre miembros de la comunidad. Sin duda, el impulso que ha dado los sistema ubicuos a la sociedad de la información ha marcado un antes y un después en el concepto de *relación entre usuarios*.

De lo que no cabe duda es que en la actualidad, la investigación y el desarrollo de nuevos sistemas de computación ubicua es una realidad. Cada vez más usuarios poseen un dispositivo móvil de última generación con potencial suficiente para realizar las acciones que anteriormente comentábamos y las que aún están por venir. Además, la maduración de las tecnologías inalámbricas y los sistemas de comunicación entre plataformas han hecho que vivamos en una sociedad hiper-conectada (Marinho, 2009) donde la inmensa mayoría de sistemas electrónicos es capaz de compartir información relativa a la persona que los usa. De esta forma, en los últimos años el usuario no percibe los sistemas ubicuos que lo rodean como algo extraño ajeno a él, sino como un elemento más del mundo que lo rodea.

SISTEMAS UBÍCUOS EN EL FUTURO

> *A vision of the future is one in which our world of everyday objects and places becomes infused and augmented with information processing and exchange. In this vision, the technology providing these capabilities is unobtrusively merged with real world objects and places, so that in a sense it disappears into the background, taking on a role more similar to electricity - an invisible pervasive medium*
>
> (Wejchert, 2000)

El ordenador persona es, tal vez, el aparato tecnológico que más decepciones provoca en la sociedad actual, debido en gran parte a su gran capacidad y versatilidad. Aún para los usuarios de la última década, sigue siendo tremendamente complicado y difícil de utilizar. Sin embargo, una inmensa mayoría estos usuarios disfrutan sin ningún problema de otros elementos como el teléfono, el coche, el televisor o los reproductores musicales a pesar de que, en cierta medida, ignoran su funcionamiento interno. Numerosos avances históricos ponen de manifiesto que, a medida que un elemento se hace más complejo, la tecnología mejora y se hace más accesible, más "orientada al ser humano". Esta orientación a la persona hace que sea tan fácil de usar que en ciertos casos se vuelve virtualmente invisible hasta que parece desaparecer. En este momento decimos que el dispositivo ha entrado en la vida del usuario.

El objetivo de la tecnología es conseguir que los usuarios presten toda su atención en la actividad que deben llevar a cabo, en lugar de tener que concentrarse en el uso del dispositivo como un elemento central, perdiendo de esta forma el propósito de ayuda con el que surgen. El principal objetivo es que el usuario obtenga una cierta comodidad a la hora de emplear estas herramientas, en lugar de generar problemas a la hora de su uso. La estrategia más adecuada para lograr esta abstracción de la tecnología como elemento central del desarrollo sería que no precisen de la atención consciente del usuario, que posean una discreción tan elevada que parezcan desaparecer de la conciencia humana. Deberían estar disponibles en todo momento y cualquier lugar, probablemente deban estar completamente integradas en los objetos de uso común que pasaría a formar parte de una red más amplia de elementos interconectados entre sí, la cual no sería posible observar porque sus componentes estarían totalmente disueltos en el entorno y serían usados sin necesidad de pensarlo, intuitivamente.

Hace unas décadas se comentaba la posibilidad de que dentro de un par de décadas, el poder computacional será tal que podrá incluir fácilmente una cierta "inteligencia" en los artefactos comunes, pero hoy día esto se ha vuelto realidad. Muchos de los objetos que son empleados habitualmente por cientos de miles de personas incorporan procesadores, memorias, sensores, conexión a Internet de gran capacidad y muchas más características. De este modo, los elementos se vuelven cada vez más y más "amigables", fáciles de utilizar y sobre todo, inteligentes. El concepto que posiblemente caracterizará a la mayoría de sistema ubicuos del futuro será que no necesitarán encenderse ni apagarse, ni siquiera que sean programados rígidamente por el usuario, sino simplemente se activarán de manera autónoma cuando así estime necesario el propio dispositivo.

Todo apunta a que la computación cada vez será más y más minúscula, para cumplir la premisa de que debe estar en todas partes. Sin embargo, esto no implica que los PCs vayan a desaparecer de manera fulminante, pero es obvia que deberán convivir e incluso rivalizar con una gran variedad de dispositivos electrónicos, como es el caso de smartphones, PDAs, tablets, etc. En lugar de fijar el objetivo de la computación actual en crear dispositivos cada vez más rápido y potentes, grandes y sobrecogedores, el paradigma de desarrollo de hardware actual se centra en conseguir dispositivos pequeños y baratos. La potencia ya no es un hándicap para los dispositivos, sino que este objetivo se

centra ahora en la interconexión de los miles y miles de elementos que rodean al usuario o que, en los próximos años, terminarán por hacerlo. De esta forma, se trata de una gran red organizada donde cada nodo, procesará la información relevante y la compartirá con el resto de elementos del ecosistema tecnológico. Sin ir más lejos, esto se aplica actualmente a entornos asistidos entre los cuales se encuentran las viviendas inteligentes. Para su correcto funcionamiento, el entorno de estas viviendas se encuentra repleto de pequeñas computadoras, los sensores, encargados de obtener cierta información del usuario y del propio entorno que lo rodea.

Más allá de la obtención de datos y de su procesamiento, el verdadero avance de la electrónica actual reside en la posibilidad de que los elementos que rodean al usuario establezcan una comunicación. Dicha comunicación ayudará por un lado al propio funcionamiento del ecosistema tecnológico y por otro, hará que el usuario se abstraiga casi en su totalidad del funcionamiento de dicho ecosistema. Así, estamos viajando por un camino en el que la independencia y la autosuficiencia de los sistemas informáticos se están convirtiendo en una realidad. De esta forma, el frigorífico sabrá a la perfección qué alimentos debe comprar el usuario y de esta forma, realizar automáticamente la compra a través de una tienda on-line. Además, gracias a la determinación del clima actual en la zona donde se encuentra, podrá elegir el tipo de alimentos que el usuario probablemente deseará.

Actualmente los sistemas integrados en el hogar del usuario, entre los que se encuentran los propios electrodomésticos, satisfacen una serie de objetivos de manera independiente. Sin embargo, a pesar de esta independencia, tienen la capacidad de comunicarse los unos con los otros. De esta manera, a través de una red de interna comunicación, será posible su interconexión con el fin de obtener información a partir de otros elementos. En esta configuración, cada sistema, cada electrodoméstico o cada dispositivo móvil se comportarán como un nodo dentro de una red superior de datos, la cual comunica y distribuye la información entre el resto de elementos.

Actualmente la red de comunicación empleada para esta finalidad es Internet o redes de área local. Este tipo de infraestructuras permiten garantizar la visibilidad de los nodos de la red y además, la transferencia de información entre dichos nodos.

Un claro ejemplo de esta situación es el caso de una aspiradora regule su potencia de manera automática en base a la cantidad de polvo que está absorbiendo o del tipo de superficie que esté aspirando. Esto último podrá determinarse gracias a la instalación de etiquetas de información en los productos, las cuales brindan un enlace a la red en el cual se encuentra toda la información específica relacionada con el objeto.

SENSORIZACIÓN Y ORÍGENES DE DATOS

En los sistemas de información actuales, la información puede proceder de una gran cantidad de fuentes, desde archivos locales hasta grandes bases de datos distribuidas. Pero esto no siempre ha sido así. A principios de la década de los 60, la única vía para almacenar información era la generación de archivos en los propios equipos en los que se guardaba toda la información deseada. De esta forma, para acceder desde otro equipo a dicha información, era necesario grabarla en cintas magnéticas y transportarlas hasta el destino. Además de este problema, los datos se guardaban de forma independientes en un formato plano, es decir, no existía ningún tipo de relación entre ellos.

La evolución a esta forma de almacenar la información llegó en el año 1970 de la mano de Edgar Frank Codd, quien en un artículo titulado *"A Relational Model of Data for Large Shared Data Banks"*

definió el modelo relacional y publicó una serie de reglas para la evaluación de administradores de sistemas de datos relacionales y así nacieron las bases de datos relacionales. A partir de este descubrimiento, Larry Ellison desarrolló la base de datos Oracle, cuyas evoluciones llegan a nuestros días.

A mediados de los 80, con el desarrollo de Internet se produce un cambio trascendental en los sistemas de información: las bases de datos centralizadas. Gracias a ellas, varios equipos que trabajaban a miles de kilómetros de distancia, podían acceder a la misma información e incluso actualizarla y visualizar los cambios de manera inmediata. Poco años más tarde se crea el lenguaje SQL específico para consultas, lo cual supuso que la información almacenada en las bases de datos fuera accesible de una manera sencilla y con un lenguaje totalmente adaptado a este fin.

El siguiente paso en términos de origen de información podría ser la aparición de los sistemas distribuidos como RPC, EDI, CORBA, COM o APPC. Sin embargo, la verdadera evolución llega cuando se alcanza una independencia de las aplicaciones, es decir, una aplicación podría poner a disposición de otros sistemas una gran cantidad de información, y otro sistema acceder a ella sin tener ninguna relación entre ellos. Esto se consigue gracias a los servicios web y sobre todo a la estandarización de los mismos gracias a los lenguajes de descripción WSDL y el estándar SOAP.

Los sistemas de información y especialmente los sistemas ubicuos, necesitan compartir una gran cantidad de información, ya que la gran mayoría de dichos sistemas se encuentran distribuidos alrededor del propio usuario. De esta forma y gracias a los servicios web y las bases de datos centralizadas, la distribución de la información a lo largo de la red de dispositivos que conforman el sistema ubicuo se realiza de una manera eficaz y logrando una mayor independencia entre las plataformas.

Este envío de información de un lado hacia otro es especialmente importante cuando es necesario trabajar con datos procedentes del propio usuario, dicho de otro modo, con datos que provienen de sensores instalados en el dispositivo móvil del usuario o en cualquier otro lugar. Dicha información es generada en el propio sensor, sin embargo debe ser distribuida a lo largo del sistema de información para que otros elementos de dicho sistema puedan procesarla con el fin de obtener nueva información derivada de la primera. En el middleware y el sistema de reconocimiento de actividades que expondremos a lo largo de este documento, se hace uso intensivo de la transferencia de información de la que hablábamos, de manera que todos los sistemas pueden tener constancia del estado y de la información generada por otro sistema independiente a éste.

DISPOSITIVOS MÓVILES Y SISTEMAS DE COMPUTACIÓN UBÍCUA

Anteriormente hemos puesto de manifiesto la clara relación existente entre dispositivos móviles y computación ubicua. Si bien no sólo los terminales móviles inteligentes forman la red hardware de sistemas ubicuos actuales, sí que son los más numerosos y los más empleados por los usuarios, ya sus dispositivos móviles forman parte de su vida cotidiana.

Como se comprobará más adelante, el uso de aplicaciones lanzadas por contexto supone un paso adelante en la computación orientada a dispositivos de bolsillo. Esto se debe a que es el propio dispositivo el que lanza las aplicaciones en base a las características de su entorno y al estado del usuario que lo porta. Mediante la obtención, análisis y procesamiento de los datos procedentes de los

sensores, se reduce al mínimo la interacción entre el usuario y el propio dispositivo, agilizando así el uso de éste último.

Además, el desarrollo de aplicaciones capaces de lanzar y ser lanzadas en base a unas determinadas condiciones del contexto que rodea al usuario y a los dispositivos que lo rodean, permite que las aplicaciones intercambien información y compartan la funcionalidad.

Sin embargo, existen varias limitaciones y problemas que deberán ser resueltos durante la investigación en este campo. Un ejemplo de estas limitaciones y que será tratado más adelante, es la sintaxis para el intercambio de información, puesto que todas las aplicaciones deberán comprender la información de contexto generada por otra para poder adaptar así su funcionamiento y además, para poder enviar información adicional una vez las aplicaciones sean disparadas desde otras aplicaciones, indicando los parámetros de inicialización de la aplicación.

Paralelamente al concepto de *ubicomp*, los dispositivos móviles han presentado una evolución extraordinaria en la última década. Desde que saliera al mercado el primer dispositivo móvil inteligente en 1993, cientos de marcas han comercializado dispositivos móviles desde entonces. El primer gran fabricante que durante años ha estado al frente de la producción y distribución de dispositivos fue Nokia. La compañía finlandesa aportó multitud de ideas que hoy en día aún se mantienen en los dispositivos móviles de este y otros fabricantes. Sin embargo, sus teléfonos estaban diseñados para un fin: realizar y recibir llamadas. Aunque a medida que pasaba el tiempo los terminales se fueron haciendo más y más sofisticados, su objetivo seguía siendo el mismo.

Sin embargo, en 2007 se produjo una revolución tecnológica sin precedentes: el lanzamiento al mercado del primer teléfono móvil de la empresa *Apple*®, el *iPhone*®. Este dispositivo marcó un antes y un después en el mundo de la telefonía móvil, ya que por primera vez se creaba un dispositivo que accedía de manera habitual y automática a Internet para la sincronización, actualización y comunicación con otros dispositivos. Por primera vez el teléfono móvil no estaba diseñado específicamente para realizar llamadas telefónicas, sino para explotar al máximo las posibilidades de la comunicación en su totalidad. Esta unificación entre dispositivo e Internet permitió acercar al usuario el sistema de computación ubicua, ya que una de las principales características de ésta es el intercambio de información con otros sistemas.

Dos años más tarde, en 2009, vió la luz la primera versión del sistema operativo de *Google*®, el Android®. Al igual que el *iPhone*, los terminales de *Google* accedían a Internet de manera habitual y con total transparencia para el usuario, lo que permitía sincronizar y enviar información de un sistema a otro dando una sensación de unificación entre todos los sistemas de información. Precisamente esto es lo que se busca en la computación ubicua, que todos los dispositivos puedan compartir información y además, cada uno de ellos pueda añadir nueva información al catálogo existente.

Por algún motivo, quizás por planes de mercado o por tácticas comerciales, los dispositivos cuyo sistema operativo estaba desarrollado por *Apple* o *Google*, han ido adquiriendo una nueva dimensión. Cada vez son más los sensores que poseen, más precisos y permiten conocer con una mayor fiabilidad determinados contexto de ejecución de las aplicaciones. Sin ir más lejos podemos pensar en sensores tan comunes como el acelerómetro o la brújula, que más adelante se comentarán en qué consisten con más detalle.

En la Figura 4 podemos observar la evolución del número de ventas de dispositivos móviles a nivel mundial en función del sistema operativo empleado. Además de la clara diferencia entre los

sistemas basados en *iOS*, *Android* y *Symbian OS* respecto al resto de sistemas, podemos ver la evolución que estos tres sistemas operativos han sufrido a lo largo del tiempo, desde mayo de 2009 hasta mayor del año 2010. Como se puede observar, la tendencia al alza de los sistemas basados en Android ha sido claramente mucho mayor que el resto de sus competidores comerciales. Es más, en los últimos meses de la comparativa realizada y publicada por *adMob* como *Mobile Metrics Report* en Junio de 2010, se puede ver como el número de dispositivos con el sistema operativo Android ha sobrepasado a los dispositivos con Symbian OS, el sistema operativo por excelencia de Nokia.

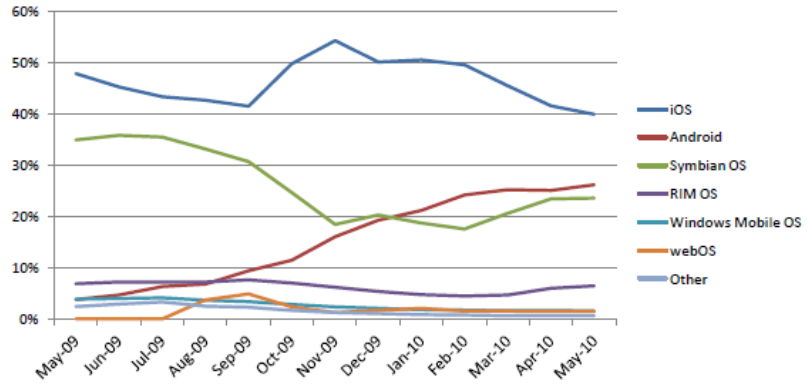


Figura 4: Evolución de sistemas operativos móviles

El objetivo de este documento no es realizar una comparativa entre sistemas operativos, sino introducir en cierta manera la computación ubicua. Por tanto, una primera pregunta que podría ser planteada es el porqué de esta comparación. La respuesta es inmediata, un análisis de los perfiles de los dispositivos más vendidos puede dar una clara idea de cómo está evolucionando la sociedad y la computación ubicua en general.

Tal y como decíamos anteriormente, salvando las distancias en cuanto a software entre los distintos modelos de dispositivos, los sistemas sensoriales son una clara diferencia que en gran parte hace que la balanza se incline hacia el lado de *iOS* y de *Android*. Obviamente los sensores por sí mismos no constituyen una experiencia enriquecedora para el usuario, sino que son las aplicaciones que hacen uso de ellos las que consiguen dotar al sistema de un conjunto de funcionalidades que años atrás eran inimaginables.

Otro gran avance que los sistemas operativos destacados tienen en común es su funcionamiento, el cual se lleva a cabo de una manera altamente comunicada con otros sistemas. Como se citaba anteriormente, los dispositivos móviles han dejado de ser unos instrumentos electrónicos independientes para convertirse en uno de los sistemas computacionales con un mayor número de comunicaciones e intercambio de información del panorama de la tecnología actual. En la Figura 5 podemos observar como el tráfico de internet desde dispositivos móviles crece mes a mes de forma continua. Este incremento de accesos a internet en los dispositivos móviles se traduce en una mayor ubicuidad en los sistemas de comunicación actuales. De nuevo Weiser en otra de sus obras (The coming age of calm technology, 1996), tenía razón cuando decía que la *edad de la calma tecnológica*[6] llegaría cuando los dispositivos se conectaran unos a otros a través de internet, usando tecnología multicast que le permitiera estar informados de todo lo que les pasaba a los demás dispositivos de su entorno.

[6] Manera en la que Mark Weiser se refería a la tercera era de la computación

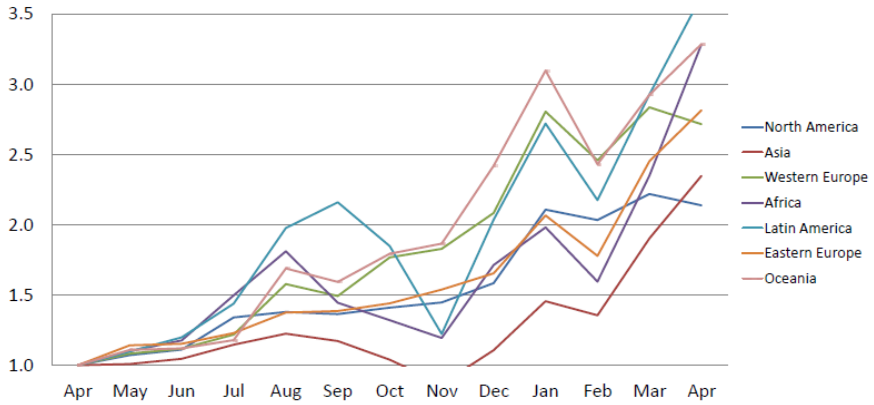


Figura 5: Uso de internet en dispositivos móviles desde abril de 2009 tomando como valor base los empleados en esta fecha

Según un informe publicado en abril de 2010 por los analistas de Morgan Stanley[7], se estima que en el año 2015, el 54% de los dispositivos móviles europeos tengan acceso a internet a través de redes 3G. Además, el número estimado de dispositivos móviles en el mundo se multiplicará por 10, entendiendo por dispositivos móviles cualquier Smartphone, mp3, Tablet, GPS, teléfonos móviles y sistemas de entretenimiento portátiles.

Sin embargo, aunque la conectividad entre distintos sistemas ha aumentado en los últimos años, existen limitaciones a la hora de intercambiar información con aplicaciones que se ejecutan en el propio dispositivo. Esto ocurre debido a la falta de una arquitectura extendida en los dispositivos que permita establecer unos puntos de entrada al sistema, a través de los cuales leer y enviar información. De esta manera, la información enviada podría ser consultada por otras aplicaciones, e incluso administrada por el núcleo del sistema con el fin de certificar la seguridad del sistema.

En el próximo capítulo estudiaremos la estructura del middleware propuesto y la arquitectura que debería tener cualquier sistema basado en contexto para asegurar la posibilidad de conexión entre aplicaciones así como la eficiencia de la propia aplicación. Con esto entraremos de lleno en el mundo de la computación ubicua, trabajando con los diferentes sensores que los dispositivos nos proporcionan en la actualidad y nos proporcionarán en un futuro para identificar diferentes aspectos del contexto que rodea al dispositivo y al propio usuario.

LIMITACIONES EN LOS DISPOSITIVOS MÓVILES

Durante los últimos diez años, la evolución realizada en el hardware de los dispositivos móviles, ha permitido reducir el tamaño y aumentar la capacidad de computación de tal manera que se han convertido en auténticos ordenadores de bolsillo. Sin embargo, a pesar de esta evolución, los dispositivos móviles siguen teniendo ciertas limitaciones que hacen que el desarrollo de cualquier tipo de aplicación deba ser mucho más cuidadoso y complejo. Los principales problemas con los que nos encontramos a la hora de generar sistemas complejos para estos dispositivos son los siguientes.

LIMITACIONES ENERGÉTICAS

[7] http://www.morganstanley.com/

En la mayoría de las ocasiones, las aplicaciones contextuales obtienen la información a partir de los sensores instalados en el propio dispositivo móvil. Sin embargo, los sensores presentes en estos dispositivos están limitados debido al consumo energético necesario para incluir elementos más precisos, con una mayor frecuencia de muestreo o de mayor tamaño. Esto hace que la experiencia a la hora de obtener datos a partir de los sensores es mucho más limitada que empleando sensores específicos con una fuente de alimentación independiente.

Otro problema relacionado con el consumo es la precaución que se debe tener a la hora de desarrollar procesos con un elevado coste computacional. Se debe ser cauto con el uso del procesador del dispositivo con el fin de alargar al máximo el tiempo de uso de la batería sin necesidad de recarga. El aumento del tiempo de uso producirá una mejor experiencia del usuario sobre el sistema desarrollado.

Una primera propuesta para solucionar este problema sería realizar el procesamiento en dispositivos más potentes como por ejemplo ordenadores portátiles, y dotar a estos de una red de sensores que permitan obtener parámetros del contexto (OPPORTUNITY: Towards opportunistic activity and context recognition systems, 2009). Sin embargo esta solución posee un inconveniente, dado que el uso de equipos más pesados y voluminosos haría perder al sistema el concepto de ubicuidad. El usuario no sería capaz de llevar siempre consigo el pesado equipo y probablemente optaría por no emplearlo. Sin embargo, el verdadero potencial de los dispositivos móviles es que, como comentábamos anteriormente, su peso y volumen permiten que sea llevado de manera continua por el usuario durante la mayor parte del día. Esto convierte a los dispositivos móviles en uno de los mejores equipos de computación ubicua.

LIMITACIONES DE ALMACENAMIENTO

Las aplicaciones necesitan cada vez más un elevado espacio de almacenamiento para ser instaladas y para almacenar los datos que posteriormente serán procesados. Si esto se une a la gran cantidad de datos que pueden ser obtenidos del contexto de ejecución de la aplicación y al almacenamiento del histórico de datos pasados, se puede observar las necesidades de almacenamiento.

Sin embargo, los dispositivos móviles poseen una capacidad limitada de almacenamiento, por lo que muchos sistemas operativos limitan el espacio dedicado a cada aplicación. A pesar de esto, los dispositivos de última generación poseen una memoria física muy superior a la de sus antecesores, pero aun así están muy lejos de las capacidades de sus hermanos mayores, los ordenadores personales.

Esta limitación obliga a las aplicaciones contextuales a ser cuidadosas con los datos que almacenan y con la periodicidad a la que lo hacen. Además deben tener presente en todo momento el hecho de eliminar los datos que no volverán a ser usados.

LIMITACIONES COMPUTACIONALES

La obtención de los datos RAW de los sensores de los dispositivos no suele presentar un problema computacional para dispositivo. Sin embargo, el procesamiento de estos datos para la generación de información útil para las diferentes aplicaciones, necesita un tratamiento intensivo de dichos datos, lo que se traduce en capacidad de cómputo.

Tradicionalmente, el contexto obtenido mediante los dispositivos móviles se limitaba a unos datos poco tratados y de escaso interés para las aplicaciones reales (Kotz, 2000). Sin embargo poco a poco esta limitación ha ido haciéndose más sutil debido al avance en la capacidad de cálculo de los dispositivos de última generación, aunque sigue siendo aún una limitación real.

Una solución a este problema es externalizar en un servidor todo el procesamiento costoso computacionalmente (Context-aware computing applications, 1994). Sin embargo, como se comentaba anteriormente, esto podría conllevar un uso excesivo de los canales de comunicación del dispositivo.

LIMITACIONES DE CONECTIVIDAD

En un mundo interconectado, la comunicación entre dispositivos móviles es esencial. Los agentes contextuales deben intercambiar continuamente información de contexto sobre el entorno del usuario para incrementar el potencial de las aplicaciones. Sin embargo, no todos los dispositivos gozan de un sistema de conexión de banda ancha móvil a internet o el volumen de datos está restringido por el operador. Estos dos problemas resultan muy comunes en los sistemas actuales, por lo que las aplicaciones deben tenerlo en cuenta a la hora de enviar y recibir los datos necesarios.

Es realmente difícil solucionar las limitaciones de conectividad de los dispositivos móviles, aunque afortunadamente la conectividad móvil mejora constantemente. Según una comparativa realizada en septiembre de 2010, la velocidad de las redes móviles de datos en Europa es de 3.7Mb/s de media[8], lo cual es muy superior al 1.8Mb/s que se registró en abril de 2007.

[8] http://www.broadband-expert.co.uk/

CAPÍTULO 3: MIDDLEWARE PARA APLICACIONES BASADAS EN CONTEXTO

Metadata are data about data. Middleware is software about software.

- Nick Gall -

A lo largo de este capítulo será presentado un middleware para la gestión del contexto de las aplicaciones en dispositivos móviles. Éste permitirá aislar completamente la aplicación de la generación del contexto, de manera que la lógica necesaria en las aplicaciones que emplean información contextual se reduce significativamente. Además de esto, se elimina gran parte de las necesidades computacionales requeridas para la obtención de contextos derivados, los cuales necesitan una integración de la información procedente de los sensores que habitualmente, consiste en el uso de operadores lógicos para la agregación de contextos. Como se verá a lo largo del capítulo, esto se consigue mediante la reutilización de los módulos proveedores de contexto. Por último, la arquitectura propuesta recoge de manera global todas las necesidades identificadas en la computación ubicua actual, lo cual hace que diferentes aplicaciones puedan tener una estructura común gracias a la definición de una serie de puntos de acceso comunes al middleware. De esta forma es posible que la reusabilidad del código sea mucho mayor que en arquitecturas de desarrollo específicas.[9]

APLICACIONES BASADAS EN CONTEXTO

En los últimos años, la computación basada en contexto ha experimentado una gran revolución debido especialmente a tres motivos: la integración de sensores en dispositivos móviles, aumento de la presencia de móviles de última generación en la vida cotidiana de los usuarios y la conectividad masiva entre dispositivos y con Internet.

Por un lado, el avance tecnológico ha permitido integrar en los dispositivos móviles de última generación una gran cantidad de sensores (Some computer science issues in ubiquitous computing, 1993), que años atrás necesitaban de grandes infraestructuras para su montaje. Ejemplo de estos sensores son los acelerómetros triaxiales, chips de posicionamiento basados en GPS, elementos de conectividad inalámbrica como Wifi, RFID o Bluetooth así como sensores de luminosidad entre otros.

Por otro lado, la integración de los dispositivos móviles en la vida diaria de los individuos de una población, permite el desarrollo de aplicaciones de utilidad para los usuarios que puedan usarse en cualquier lugar, gracias a la portabilidad de los dispositivos donde se ejecutan. Debemos tener en cuenta que al tratarse de elementos portátiles, el ahorro energético debe estar presente en cualquier desarrollo que se realice, dado que será necesario maximizar el tiempo de uso del dispositivo, evitando así que las baterías supongan un problema. Además de esto, es necesario maximizar el tiempo de baterías para la comodidad del usuario del dispositivo, ya que una frecuencia de recargas elevada puede llegar a disgustar al usuario.

[9] Debido al uso de la nomenclatura extendida a lo largo de la bibliografía, en este capítulo serán numerosas las referencias a elementos que para su notación se emplea la lengua inglesa.

En último lugar y, en gran medida el detonante del desarrollo masivo de aplicaciones ubicuas, se encuentra el aumento de elementos de conectividad en los dispositivos portátiles, especialmente aquellos que permiten la conexión a Internet. A lo largo del año 2009, el volumen de tráfico de Internet llevado a cabo a través de dispositivos móviles creció en un 193% y en marzo de 2010 el 38% de los dispositivos en Europa poseían una conexión de acceso a internet, la cual puede emplearse de manera ininterrumpida. En el estudio "*European Mobile Forecast: 2008 To 2013*" se afirma que en 2013, 125 millones de europeos tendrán acceso a Internet en su dispositivo móvil de forma regular. Esto supone que en cinco años, el número de accesos a internet fuera del hogar crecerá en un 315%, o lo que es lo mismo, una de cada siete personas en Europa podrán disfrutar de esta tecnología.

Todas las tecnologías descritas anteriormente hacen que el dispositivo móvil pueda obtener más información del entorno que lo rodea y del propio usuario que porta el dispositivo. Toda esta información puede ser empleada por diversas aplicaciones de manera que la interacción entre el dispositivo y el entorno haga más fácil el uso del dispositivo por parte del usuario.

Otro aspecto a tener en cuenta en el ámbito de aplicaciones basadas en contexto son las preferencias del usuario. Dichas preferencias pueden ser obtenidas a partir de experiencias pasadas y, tras un proceso de aprendizaje, generar una base de conocimiento a partir de la cual realizar una adaptación de la aplicación en función de las preferencias almacenadas.

El verdadero problema de los sistemas basados en el conocimiento del contexto del usuario, se encuentra en la ingente información de datos que se recuperan tanto del propio usuario como del entorno que lo rodea. Además, es necesaria la interacción con otros dispositivos y con otros usuarios, lo cual aumenta aún más la cantidad de información a procesar y, sobre todo a clasificar. Por ello la primera labor que debemos emprender es la búsqueda de un sistema de almacenamiento y una estructura que almacene toda la información requerida por el sistema de una manera ordenada y fácil de acceder.

Más adelante veremos una primera aproximación del conjunto de valores que el sistema debe almacenar para describir el estado del usuario en un momento determinado pero, de una forma muy general, podemos decir que los datos mínimos que debemos conocer del usuario a partir de los datos de contexto son: Quién, Qué, Cuándo, Dónde y Cómo. Estas cinco preguntas, también conocidas como 4W1H (Who, What, When, Where and How), permiten conocer toda la información necesaria sobre el usuario y su entorno.

Durante los primeros años de investigación en el campo de la computación ubicua se pretendió definir en qué consistía un sistema "wearable", y la mayoría de los autores llegaron a la conclusión de que se definían como aquellos sistemas que acompañan al usuario allá donde vaya y que su ejecución se realiza de manera ininterrumpida. Esta primera definición se rodeó de una connotación de posicionamiento, ya que la aplicación más evidente fue la determinación de la posición de un usuario en todo momento con el fin de conocer en qué lugar se encontraba (Context-Aware Computing Applications, 2003), (Providing location information in a ubiquitous computing environment, 1993), (An Architecture for Location Aware Applications, 2002). Durante los cinco años siguientes aproximadamente, la tendencia de los sistemas ubicuos se encaminó de manera casi exclusiva al cálculo constante de la posición y a sus aplicaciones. Esto se vio favorecido en gran medida por el lanzamiento del sistema GPS, el cual alcanzó capacidad operativa total en el año 1995, y a la creación de sistemas portátiles de receptores para esta tecnología.

Ya en el año 1998 el campo de investigación de los sistemas ubicuos abandonó su tendencia centrada en el posicionamiento y amplió el horizonte hacia el estudio más detallado del contexto

general que rodea al usuario. En realidad, un sistema ubicuo debe estar muy relacionado con el contexto, y este a su vez se define como "*that which surrounds, and gives meaning to something else"* (aquello que rodea y aporta significado a alguna otra cosa) por lo que parece algo primitivo el hecho de centrarse en algo tan concreto como la posición para generalizar todo un contexto de situación (There is more to Context than Location, 1999), (What GPS doesn't tell you: determining one's context with low-level sensors, 1999). En el trabajo de *Mike Spreitzer y Marvin Theimer* (Providing location information in a ubiquitous computing environment, 1993) podemos ver que el campo de investigación en sistemas ubicuos se extiende a factores tan diversos como la presión ambiental, luminosidad, temperatura, audio, etc. Sin embargo, en el trabajo de Albrecht Schmidt y Jessica Forbess, tan sólo es posible diferenciar dos contextos distintos: interior y exterior.

Durante los años 2000 y 2001 se desarrollaron una serie de frameworks para la gestión de sistemas ubicuos (The Context Toolkit: Aiding the Development of Context-Aware Applications, 2000), (CybreMinder: A Context- Aware System for Supporting Reminders, 2000) y (Understanding and Using Context, 2001). En concreto, la arquitectura más extendida entre los sistemas de la época fue *ContextToolkit*. Esta herramienta permitía integrar varios sensores en un mismo sistema de manera que la adición de un nuevo sensor se pudiera realizar de una manera rápida y eficaz sin ser necesario reprogramar el sistema completo. Esto se realizaba a través de los denominados *widget*, los cuales se entendían como elementos independientes del sistema que al ser integrados en él pueden convivir sin ningún problema, siendo el núcleo del sistema el responsable de obtener y utilizar los datos procedentes de ellos en base a las necesidades. En cambio, la principal limitación del sistema *ContextToolkit* era que no permitía la obtención de datos de contexto por parte de las aplicaciones, lo cual se enfrentaba con la filosofía de aplicaciones basadas en contexto que la comunidad científica seguiría años más tarde.

Por otro lado apareció *GeoNotes* (GeoNotes: Social and Navigational Aspects of Location-Based Information Systems, 2001), un sistema ubicuo "tradicional" que permitía obtener información de contexto centrada en el posicionamiento del usuario. Decimos que se trata de un sistema "tradicional" porque no permite la integración de más información de contexto además de la propia posición del usuario. Esto supone una gran limitación, ya que la información relacionada con las características del entorno como, por ejemplo, la luminosidad, ambiente de sonido, y temperatura no podía ser tratado con el sistema *GeoNotes*. Sin embargo, el sistema desarrollado por *Espinoza et al* no sólo se trataba de un framework para el desarrollo de sistemas ubicuos, sino que contenía una aplicación de notas geoposicionadas, aunque debido a las limitaciones comentadas anteriormente, el sistema no gozó de demasiado éxito, ya que no aportaba ninguna características de interés respecto a trabajos anteriores. No queda claro que aporta *GeoNotes* con respecto a trabajos anteriores.

En 2003 se da un paso más en la investigación y el desarrollo de aplicaciones basadas en sistemas ubicuos: la interconectividad de sistemas (Context-awareness on mobile devices - the hydrogen approach, 2003). Está claro que algo tan estrechamente relacionado con el usuario, necesita relacionarse con otros sistemas igualmente relacionados con sus usuarios. Esto amplía las posibilidades del sistema y permite relaciones entre varios sistemas con el fin de enriquecer la funcionalidad de la red ubicua. La arquitectura propuesta por *Thomas Hofer et al* asienta las bases del primer framework para el desarrollo de aplicaciones ubicuas multisensoriales de la bibliografía. Gracias a la subdivisión en capas del sistema, hacía posible que un nuevo sensor pudiera ser incorporado al sistema mediante la inclusión de un nuevo driver en la capa de adaptación (Adaptor Layer). Todo el conjunto de sensores de la capa inferior se controlaban mediante la capa intermedia de control (Management Layer). Sin embargo, lo más importante de todo el sistema fue la generación de una última capa por encima de la de control denominada capa de aplicación (Application Layer). Esta

capa permitía a las aplicaciones consultar los datos provenientes de los sensores y basarse en ellos para generar una aplicación basada en el contexto donde el usuario se encuentra. Por tanto, el principal beneficio del framework fue la facilidad de adaptación y la usabilidad que tenía. Sin embargo, el punto débil del sistema fue la interconectividad entre usuarios. Cada usuario poseía su propio conjunto de datos y estos no eran visibles por el resto de aplicaciones del sistema. Esto aseguraba la confidencialidad de los datos, pero mermaba la proyección del sistema.

A partir de 2003, surgieron dudas sobre la seguridad de los sistemas ubicuos. Esto se debió en gran medida al inicio de la compartición de datos por parte de la comunidad de usuarios. La mayoría de los temas de discusión a lo largo de los últimos años se han centrado en la seguridad en datos de posicionamiento (Location Privacy in Pervasive Computing, 2003), (Security for Ubiquitous Computing, 2002) y (Context Awareness by Case-Based Reasoning in a Music Recommendation System, 2008) ya que estos datos permiten a individuos malintencionados conocer la posición exacta de un determinado usuario de manera directa, puesto que la mayoría de sistemas emplean datos objetivos como latitud y longitud para almacenar la posición del usuario.

Sin embargo, la verdadera revolución de la computación ubicua se ha producido en los últimos tres años, debido al aumento de los sensores integrados en dispositivos móviles comerciales. Ya comentábamos anteriormente en el apartado de *introducción* que el uso masivo de teléfonos móviles ha favorecido el desarrollo de aplicaciones basadas en contexto. Un claro ejemplo de este tipo de aplicaciones es el trabajo de *Jae Sik Lee et al* (Context Awareness by Case-Based Reasoning in a Music Recommendation System, 2008). En él se implementa un sistema de recomendación musical en función a la fecha, posición, temporada, día de la semana, temperatura media, máxima y mínima así como las condiciones climáticas en el lugar y el instante actual. De esta forma, el sistema es capaz de determinar el ámbito en el que se encuentra el usuario y reproducir la música que habitualmente el usuario reproduce en un contexto semejante al actual.

Otro ejemplo de un sistema basado en contexto es el realizado por *Gabriella Castelli et al* (Extracting High-Level Information from Location Data: The W4 Diary Example in a Music Recommendation System, 2009). En este caso se trata de un diario basado en el framework W4 para la recopilación de lugares donde el usuario ha estado. El framework W4 asegura que toda información relacionada con el contexto puede englobarse en cuatro elementos distintos: Quién, Qué, Cuándo y Dónde (Who, What, When and Where), de ahí que se denomine framework W4.

MOTIVACIÓN

La motivación troncal de este capítulo es en gran medida el elevado grado de dispersión a la hora de plantear entornos comunes de ejecución de aplicaciones basadas en contexto ya que, como analizamos en la sección de estado del arte, ningún autor hasta la fecha ha definido una arquitectura de desarrollo estructurada que se adapte completamente al desarrollo normalizado de sistemas ubicuos desde un punto de vista común para la totalidad de aplicaciones móviles.

La creación de esta arquitectura supondría una gran ventaja para que las aplicaciones explotaran las ventajas que aporta el conocimiento del contexto que rodea al usuario para aumentar la funcionalidad de las mimas. Además del aumento de la funcionalidad, lo más importante sería la adaptación total del contenido presentado al usuario y el estado en el que se encuentra tanto el propio usuario como su entorno en el momento de la ejecución.

Además de crear una arquitectura normalizada, las capacidades de dicho entorno de trabajo debe dar soporte a la integración de todas las técnicas de recuperación de contexto en el dispositivo más ubicuo que se conoce: el móvil. Esto debe realizarse en todo momento maximizando el número de contextos soportados por la arquitectura de manera que el potencial del mismo para el desarrollo de aplicaciones sea lo más elevado y flexible posible.

En los siguientes apartados se presentará la arquitectura propuesta y se explicarán en detalles todas las capas que la componen. El uso de dicha arquitectura es una ventaja en sí mismo, ya que hace posible un desarrollo ordenado, estructurado y sobre todo generalizado para las aplicaciones, por lo que el coste temporal necesario a la hora de la detección y reparación de errores es mucho menor. Sin embargo, el verdadero potencial de la arquitectura se basa en la interconexión de aplicaciones. Como veremos más adelante, el uso de un middleware para obtener información desde otras aplicaciones y contribuir a la base de información con nuevos datos obtenidos en una aplicación concreta, facilita el desarrollo de aplicaciones. Esto unido a una arquitectura óptima para el desarrollo de aplicaciones basadas en información contextual, da la posibilidad de interconexiones a distinto nivel entre aplicaciones de una manera lo más transparente y sencilla posible para el desarrollador.

ESTADO DEL ARTE

> *El objetivo de un middleware es ayudar a los sistemas desarrollados a abstraerse de ciertas tareas haciendo más fácil el desarrollo.*
>
> - *Robert P. Swiss* -

El concepto de middleware no es nuevo en el campo de la computación ubicua sino que desde 2001, año en el que surgió el primer "framework", han sido muchos los productos desarrollados dentro de esta gama de sistemas. Como hemos citado, en 2001 aparece una primer aproximación a un framework de aplicaciones contextuales (A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, 2001). En este caso tan sólo se intentó identificar las características básicas de las aplicaciones contextuales y abstraerlas de manera que pudieran ser seleccionadas aquellas funcionalidades que pudieran ser generalizadas y adaptadas por el propio framework para abstraer a la aplicación concreta de la plataforma en la que se ejecutaba finalmente. El resultado de este trabajo fue la generación de "Context Toolkit", una herramienta que permitía construir aplicaciones basadas en contexto facilitando la reusabilidad de componentes, la evolución de las aplicaciones y la adquisición y el uso de información compleja. Las herramientas que brindaba dicho framework para la obtención de información por parte de las aplicaciones eran los *Widget* y los *Aggregators*. Los *Widget* no eran más que implementaciones incluidas en el propio framework que ayudaba a obtener datos comunes a partir del dispositivo, como por ejemplo la localización, servicios de TTS (Text-To-Speech), reproducción de sonidos, etc. Sin embargo los *Aggregators* iban más allá, ya que la funcionalidad no era ofrecida por el propio framework, sino que era otro sistema interconectado al framework el encargado de proveer dicha funcionalidad. Sin embargo, la principal desventaja de este trabajo era que dos aplicaciones independientes no podían compartir información, ya que necesitaban de un *Aggregator* como intermediario para ello.

En la Figura 6 se puede observar un ejemplo de una arquitectura para una aplicación concreta que interconecta varias funcionalidades procedentes tanto de *Widget* como de *Aggregators*. En ella se

puede ver claramente cómo la aplicación de interconexión hace uso de la arquitectura descrita anteriormente para realizar el intercambio de información entre los distintos módulos.

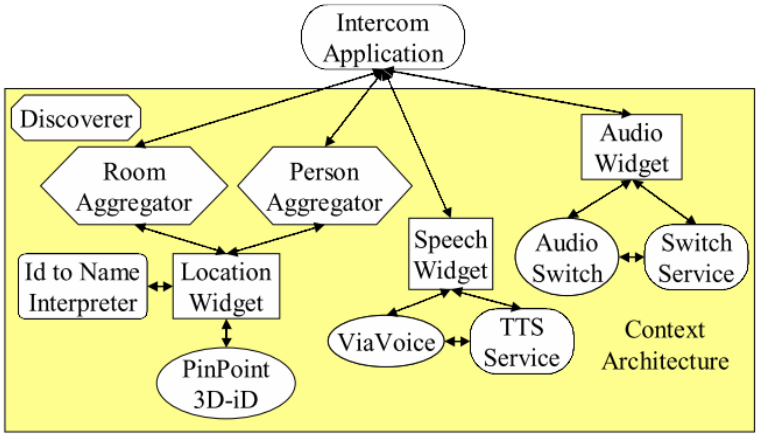


Figura 6: Arquitectura para una aplicación de interconexión contextual

Sin embargo, no sólo existen frameworks y middlewares para aplicaciones contextuales en dispositivos móviles, sino que la computación ubicua como se ha comentado en el capítulo introductorio, engloba aspectos relacionados con espacios inteligentes y redes de sensores distribuidos. En este campo, una arquitectura basada en agentes de gran impacto en los sistemas posteriores fue la arquitectura CoBrA (Using OWL in a Pervasive Computing Broker) realizada en 2003. Los espacios inteligentes, a los cuales está orientada específicamente la arquitectura CoBrA, son espacios físicos poblados con distintos tipos de sensores que proporcionan información de contexto relacionada con el usuario.

El núcleo central de CoBrA es la presencia de un agente contextual que mantiene y administra un modelo contextual compartido por una comunidad de agentes. Estos agentes pueden ser las aplicaciones alojadas en los dispositivos móviles, los servicios que se proporcionan por medio de un dispositivo ambiental o servicios web que proporcionan información sobre las personas, lugares y objetos en el mundo real. El agente contextual consta de cuatro componentes funcionales principales: la *base de conocimiento*, el *motor de inferencia contexto*, el *módulo de adquisición de contexto* y el *módulo de gestión de la privacidad*.

Además de todas las características descritas de CoBrA, este sistema usaba una ontología propia basada en OWL para la descripción de contextos, llamada COBRA-Ont (An Ontology for Context-Aware Pervasive Computing Environments, 2004). Esta ontología provee de un vocabulario especial para diferentes tipos de contexto, identificado así varios subdominios del contexto usado para representar y compartir la base de conocimiento.

Otro middleware centralizado y extensible es el propuesto en el proyecto CASS (Middleware for Mobile Context-Aware Applications, 2004), el cual es específico para sistemas basados en contexto en dispositivos móviles. El middleware contiene un intérprete, un *ContextRetriever*, un *Rule Engine* y un *SensorListener*. El *SensorListener* será el encargado de mantener la escucha para recibir actualizaciones de sensores que se encuentran distribuidos en los equipos llamados nodos sensores. A continuación, la información recopilada es almacenada en la base de datos por el propio *SensorListener*. El *ContextRetriever* es responsable de recuperar el contexto almacenado. Ambas clases pueden utilizar los servicios de un intérprete. El *ChangeListener* es un componente con capacidades de comunicación, que permite a un dispositivo recibir una notificación de eventos

basados en cambio de contexto. Las clases *Sensors* y *LocationFinder* también se han incorporado en las capacidades de comunicación. Los dispositivos móviles se conectan al servidor a través de redes inalámbricas para la recuperación de la información. Sin embargo, para reducir el impacto en las baterías del dispositivo de las continuas conexiones inalámbricas, el sistema permite un almacenamiento de dicha información en una caché local. En la Figura 7 se muestra la arquitectura general del middleware CAAS que se ha explicado anteriormente.

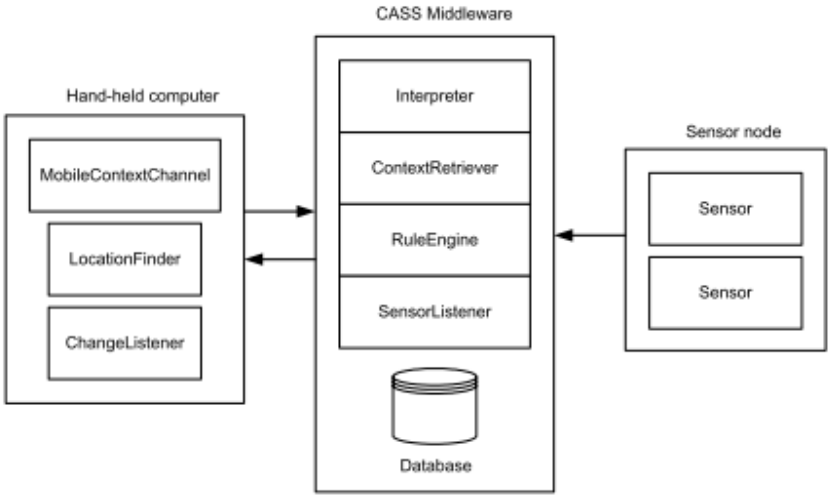


Figura 7: Arquitectura del middleware CAAS

El proyecto Hydrogen (Context-Awareness on Mobile Devices - the Hydrogen Approach, 2003) también se trata de un framework específico para el desarrollo de aplicaciones contextuales en dispositivos móviles. La novedad que este trabajo aportaba frente a otros frameworks desarrollados anteriormente era la descentralización. Mientras que en la mayoría de las arquitecturas distribuidas propuestas hasta entonces era fundamental un componente central para el manejo de toda la red de dispositivos, el sistema Hydrogen intentaba eliminar esta dependencia. Para ello se distinguían entre contexto remoto y contexto local. El primer tipo consistía en un contexto que provenía de otro sensor distribuido en la red a través de una conexión de datos, mientras que el contexto local era toda aquella información contextual recogida por el propio dispositivo. Tanto el contexto remoto como el local eran descritos mediante una clase *ContextObject*, la cual era extendida por una serie de clases más específicas en función del contexto que se desee almacenar en ella. Gracias a la estructura en árbol, permite la inclusión de nuevos contextos mediante la especialización de la entidad *ContextObject*. En la Figura 8 se muestra la estructura de objetos del framework para el manejo de contextos.

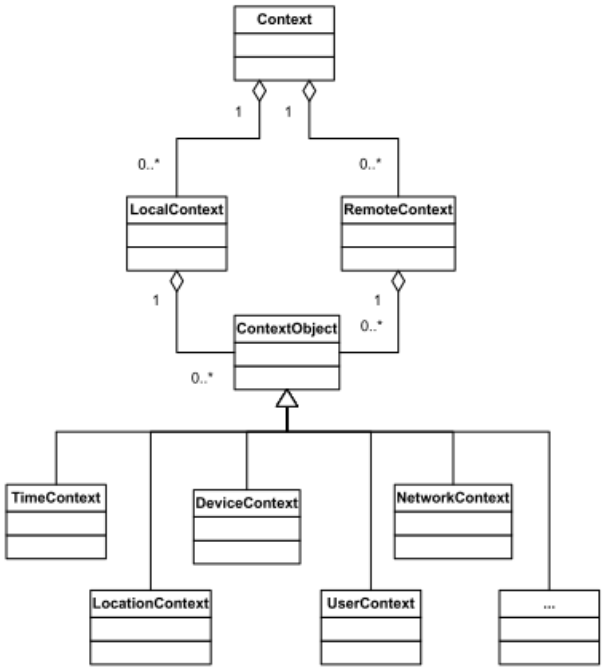


Figura 8: Estructura para el manejo de contextos de Hydrogen

La arquitectura de Hydrogen consiste en tres capas que están presentes en el propio dispositivo. La primera de ellas es la responsable de obtener los datos raw desde los sensores y procesarlos para su comprensión por los elementos de nivel superior de la arquitectura. En la segunda capa se encuentra el *Administrador de capas*, que hace de intermediario entre los sensores y la aplicación. Su objetivo es proporcionar información de contexto a partir de los datos provenientes de los adaptadores. Por último, la última capa de la arquitectura son las propias aplicaciones, las cuales implementan diferentes funcionalidades en base a la información proporcionada por el *Administrador de capas* sobre el contexto. La Figura 9 muestra la estructura en capas del framework Hydrogen, así como la comunicación existente entre los diferentes módulos del sistema.

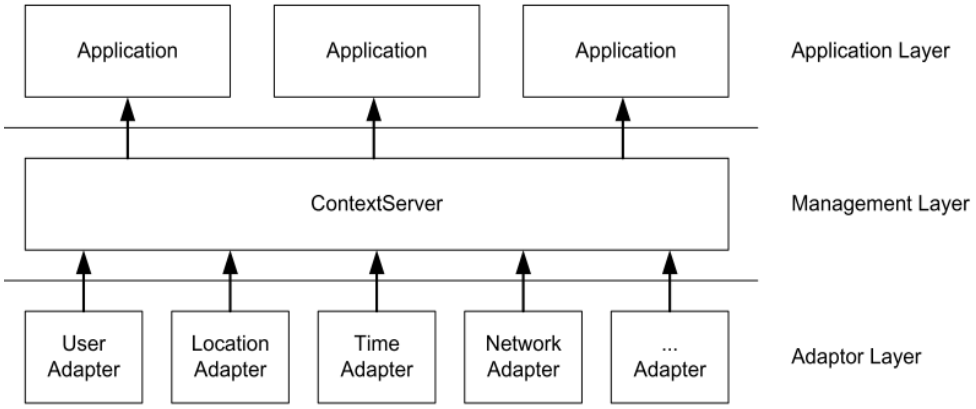


Figura 9: Estructura de capas de Hydrogen

Otro proyecto centrado en la obtención de un framework para el desarrollo de aplicaciones contextuales es Gaia (Gaia: A Middleware Infrastructure to Enable Active Spaces, 2002). Su objetivo es extender las capacidades del sistema operativo e incluir un módulo de manejo de contexto. La arquitectura de Gaia consta de cuatro capas, las cuales pueden verse en la Figura 10. El elemento más

importante sin lugar a dudas es el *Event Manager Service*, el cual es el encargado de distribuir los eventos a través de todo el espacio activo e implementar un sistema de comunicación entre dispositivos. Con la ayuda del módulo *Context Service*, las aplicaciones pueden realizar consultas e inscribirse para la obtención de información contextual específica, así como objetos de contexto de alto nivel. Y por último, el *Context File System* lleva a cabo un almacenamiento automático y centralizado de la información personal, lo cual hace posible que dicha información esté disponible en cualquier lugar donde se encuentren los usuarios.

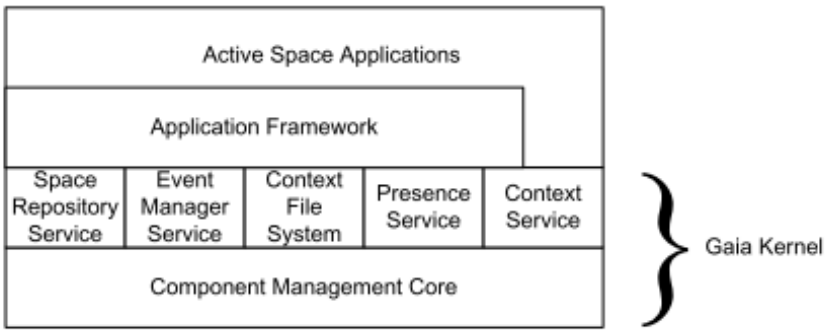


Figura 10: Estructura del framework Gaia

Por último, en 2009 aparece otro sistema llamado SharedLife (A Framework for Ubiquitous Content Sharing, 2009) que más que de un middleware se trataba de una aplicación que permitía compartir información entre varias fuentes y clientes. Dicha información era almacenada en un sistema centralizado como un conjunto de modelos semánticos de conocimiento, que generaba la llamada *memoria digital* del usuario. Además, el sistema permitía compartir dicha información con otros usuarios y aplicaciones. SharedLife maneja tres tipos de información; por un lado los eventos que describen las interacciones entre el usuario y el entorno, de manera que una determinada opinión ofrecida por el usuario, debería ser asociada a un evento y este a su vez, contendría una lista de objetos, personas y acciones involucradas en él. Por otro lado están los perfiles del usuario, los cuales almacenan información general sobre el usuario entre las que se encuentra las relaciones con otros usuarios de la red, dando al sistema un sesgo claramente orientado a redes sociales. Finalmente aparece la información multimedia anotada semánticamente, que permite al sistema obtener información de manera automática y asociarla a un determinado evento. En la Figura 11 podemos ver un diagrama que muestra la arquitectura general para una aplicación desarrollada mediante el framework ShareLife.

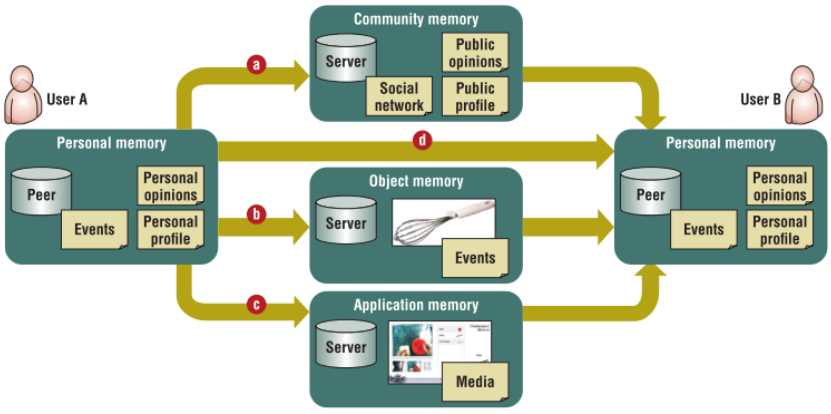


Figura 11: Modelo de memoria para el framework ShareLife

Una vez descritos algunos de los frameworks y middlewares más importantes y con mayor impacto en el campo de la computación ubicua, presentaremos en el siguiente apartado el middleware propuesta en este trabajo. La principal diferencia con el resto de sistemas expuestos es que se centra en la interconexión y lanzamiento automático de aplicaciones en el propio dispositivo móvil. Esto se hace con el fin de ahorrar cómputo y por tanto batería, mediante la reutilización de servicios. La visión que se pretender transmitir a lo largo de los sucesivos apartados es que las aplicaciones serán tratadas como un conjunto de servicios de cara a otras aplicaciones, por lo que toda la información compartida con el middleware será encapsulada y registrada. De esta forma, otras aplicaciones pueden descubrir, inscribirse o emplear los servicios ofrecidos por otras aplicaciones de una manera sencilla y totalmente transparente. Además, a diferencia de otros middlewares, toda la lógica de lanzamiento de eventos se realizará en el propio sistema, aislando así a las aplicaciones de gestionar la propia lista de sistemas suscritos a los eventos lanzados por ella misma.

PRINCIPIOS DE DISEÑO DEL MIDDLEWARE

Los sistemas ubicuos basados en contexto pueden ser implementados de varias maneras. Generalmente la implementación dependerá de las características que posean los sensores empleados para la obtención de los datos, así como las condiciones que se deseen imponer al sistema. Ejemplo de estas características son la situación de los sensores (locales o remotos), la cantidad de usuarios que pueden trabajar con la aplicación (uno o varios), la tipología de los dispositivos empleados (PC's, dispositivos móviles o tablets) o la precisión de los datos obtenidos. Basándose en estas diferencias tipologías de sistemas contextuales, se pueden distinguir las siguientes arquitecturas a la hora de desarrollar una determinada solución (Chen, 2003):

- *Acceso directo a los sensores*. Es empleada cuando los sensores están instalados en el propio dispositivo o están conectados a él físicamente. Generalmente este tipo de plataformas dependen hasta tal punto del propio hardware que en la mayoría de los casos resulta extremadamente difícil reaprovechar la lógica empleada. Por otro lado, este tipo de plataformas presentan serias limitaciones a la hora de trabajar con sensores remotos, ya que no es posible gestionar el acceso múltiple a los recursos.
- *Arquitectura basada en middleware.* El objetivo primordial de los middlewares es abstraer al desarrollador de aplicaciones contextuales del hardware específico del dispositivo. Esto se consigue mediante la inserción de una capa intermedia entre el sensor y la propia aplicación. Por tanto, el middleware será el encargado de obtener la información a partir de los sensores, procesarla, almacenarla, combinarla y distribuirla entre el resto de aplicaciones que dependen de él. Esta solución mejora notablemente la reusabilidad ya que por ejemplo, es posible ejecutar la misma aplicación en una arquitectura hardware distinta, gracias a la adaptación a un entorno común de ejecución de los sistemas.
- *Arquitectura basada en servidor.* Es este tipo de arquitectura se da un paso más en relación con la anterior, ya que toda la lógica de la generación del contexto no se ejecuta en el propio dispositivo, sino en un servidor dedicado. Mediante esta estructura se libera al dispositivo de todo el coste computacional de la adquisición del contexto. El coste puede llegar a ser elevado en función del contexto a obtener, pero en un gran número de casos de contextos derivados, el coste de la adquisición aumenta debido al uso de técnicas de aprendizaje o de clasificación. Sin embargo, esta ventaja conlleva un impacto negativo en el volumen de tráfico de información a través de las redes de datos. Dado que el contexto es obtenido en el servidor a partir de los datos procedentes del dispositivo, es necesario realizar dos operaciones de tráfico de datos. Por un lado se debe enviar los datos RAW al servidor y una vez obtenido el

contexto, devolverlo al dispositivo para que actúe en consecuencia. Otro problema derivado del anterior es la agilidad en la generación del contexto, ya que si la red de datos del dispositivo es demasiado lenta o el volumen de datos es muy elevado, el tiempo de cálculo del contexto debido al tráfico de información se puede demorar.

De igual manera que la clasificación anterior, Winograd (Architectures for Context, 2001) propone tres entornos de gestión del contexto para la integración entre diferentes procesos que hacen uso de esta información:

- *Widgets*. Estos componentes proporcionan una capa de lógica sobre el hardware del dispositivo, permitiendo aislar a las aplicaciones desarrolladas del hardware concreto instalado. Esto favorece la reutilización en las aplicaciones contextuales debido a que pueden ser ejecutadas en diferentes sistemas con hardware distinto sin necesidad de cambiar en absoluto la lógica de la aplicación, sino simplemente instalando los *Widgets* adecuados para la nueva arquitectura.
- *Modelo en red*. En este caso se externaliza la lógica de la aplicación al servidor, de manera que la única misión del cliente es enviar los datos en bruto o con un leve post-procesamiento al servidor. Éste será el encargado de manipular los datos con el fin de obtener información relacionada con el contexto. Además, este entorno basado en red facilita la integración entre la información procedente de diferentes equipos, ya que el resultado del procesamiento sigue las mismas reglas semánticas, las cuales son entendidas por el servidor. Sin embargo, el principal problema de esta arquitectura de obtención de información contextual es la reducida eficiencia si la comparamos con la arquitectura basada en *Widgets*, debido a que la información debe enviarse del cliente al servidor y viceversa. En cambio, el servidor de contexto provee al sistema de una gran robustez, dado que en caso de fallo de alguno de los componentes de procesamiento de la información del servidor, éste puede ser sustituido por otro rápidamente.
- *Modelo Blackboard.* Este modelo combina las ventajas de los dos anteriores gracias al uso de un nuevo punto de vista del contexto centrado en los datos (*data-centric view*) en lugar de en los servicios (*service-oriented model*). Esto se traduce en que las aplicaciones no deberán estar al tanto del proceso que se lleva a cabo en las capas inferiores del modelo, sino que serán las capas inferiores las que comuniquen a las superiores de los cambios que se produzcan en el contexto del usuario. Mediante el uso de eventos, una determinada aplicación puede suscribirse a una notificación producida por un servicio de nivel inferior, de manera que cuando se produzca el evento al cual se ha suscrito, el servicio enviará una notificación. Al recibir esta notificación, la aplicación será responsable de actuar en consecuencia. Por otro lado, el servicio de *Blackboard* conoce en todo momento los parámetros del contexto, así que será el responsable de enviar las notificaciones cuando se produzcan cambios en ellos que afecten a las aplicaciones suscritas.

Por último analizaremos otra clasificación para sistemas contextuales propuesta por Hong y Landay (An infrastructure approach to context-aware computing, 2001). En este caso se identifican cuatro categorías principales en las que se pueden englobar las aplicaciones basadas en contexto:

- *Librerías.* Una librería consiste en un conjunto de funcionalidades y algoritmos que permiten trabajar con una serie de datos. Las librerías son empleadas en la mayoría de los casos para realizar un reaprovechamiento del código. La funcionalidad ofrecida por las librerías se realiza con independencia del software en el que se emplea, por lo que también favorece un

desarrollo modular de los sistemas. En la actualidad, los sistemas operativos ofrecen una gran cantidad de librerías para permitir al desarrollador implementar soluciones a partir de una funcionalidad básica. En cambio, la mayoría de las veces estas librerías no son suficientes y es necesario recurrir a otras para ampliar el conjunto de algoritmos susceptibles de ser empleados.

- *Framework.* En general, un framework es un conjunto estandarizado de conceptos, prácticas, diseños y criterios que permite abordar un problema general, el cual servirá como punto de partida para desarrollar soluciones a problemas de índole similar. En el desarrollo de software concretamente, un framework consiste en un conjunto de sistemas, librerías, funcionalidades y a veces un lenguaje específico, que define una metodología de trabajo con el fin de desarrollar soluciones válidas y eficaces a un problema computacional concreto. Los frameworks facilitan el trabajo del desarrollador puesto que abstraen el nivel del problema, por tanto es posible centrarse en el diseño y el análisis del sistema en lugar de en los detalles técnicos de bajo nivel. En el ámbito concreto de los sistemas contextuales, existen varios frameworks que permiten el desarrollo de aplicaciones de este tipo. Entre todos ellos destacamos dos, los más utilizados a día de hoy, que son JCAF (The Java Context Awareness Framework (JCAF) - a service infrastructure and programming framework for context-aware applications, 2005) y J2ME. El objetivo de JCAF es proporcionar al desarrollador una arquitectura que permita manejar la información contextual proporcionada por sistemas de bajo nivel que son implementados como proveedores de contenidos. El principal inconveniente de este framework es que está desarrollado específicamente para J2ME, un sistema en desuso y con unas posibilidades muy limitadas. Por último, el inconveniente general de todos los frameworks es la dependencia de la aplicación, es decir, la aplicación debe realizarse específicamente para un framework y compilarse y ejecutarse sobre éste.
- *Toolkits.* Los toolkits son extensiones para un determinado framework que permiten añadir funcionalidad de manera que ésta sea accesible para todas las aplicaciones desarrolladas para dicho framework. Por tanto, un toolkit podría considerarse una aplicación diseñada para un framework que provee de información al resto de aplicaciones. Este diseño favorece claramente la reusabilidad de la funcionalidad, por lo que las posibilidades de error se reducen. Un ejemplo de este tipo de sistemas es el desarrollado por Anind K. Dey (The Context Toolkit: Aiding the Development of Context-Aware Applications, 2000).

Estas clasificaciones no forman en absoluto grupos cerrados, sino que la mayoría de los sistemas contextuales forman parte de más de un grupo, permitiéndose el solapamiento entre ellos. Aunque la determinación de la pertenencia de un sistema a cada uno de estos grupos nos permitirá conocer a grandes rasgos su funcionamiento, defectos y virtudes.

Una vez definidas las principales clasificaciones establecidas en la bibliografía sobre los distintos sistemas de información contextual, se presenta una estructura tradicional de este tipo de sistemas. A partir de esta estructura sentaremos las bases funcionales del middleware desarrollado y servirá como punto de partida para identificar las necesidades concretas de los sistemas basados en contexto.

ESTRUCTURA GENÉRICA DE LOS SISTEMAS CONTEXTUALES

Los middlewares para la ejecución de sistemas contextuales suelen estar diseñados mediante una estructura en capas, la cual permite alcanzar el objetivo de abstraer la información desde el más bajo nivel, generalmente asociada a los sensores, al más alto nivel, en el que se encuentran las

aplicaciones. En la Figura 12 se puede observar esta estructura general, en la que se identifican 5 capas distintas.

Figura 12: Arquitectura genérica de un middleware

En la capa inferior aparecen los sensores, elementos a través de los cuales el sistema es capaz de obtener información del exterior. Tradicionalmente, estos sensores se clasifican en tres grupos en función del origen de los datos (Location Management in Pervasive Systems, 2003):

- *Sensores físicos.* Son los más empleados ya que proporcionan información del entorno a partir de un hardware específico. Su precisión es conocida a priori, lo cual es una ventaja a la hora de conocer la fiabilidad de los datos. En la actualidad existen multitud de sensores que son capaces de capturar la mayoría de los datos físicos del entorno que rodea a los usuarios, como por ejemplo la temperatura, humedad, posición, acústica, luminosidad e incluso parámetros biológicos del propio usuario.
- *Sensores virtuales.* Este tipo de sensores obtienen la información a partir de otra aplicación accesible para ellos. Generalmente los sensores virtuales pueden obtener información acerca del propio usuario como por ejemplo su actividad diaria, relaciones con otras personas, trabajos de interés o viajes recientes. Esto se hace mediante el acceso a agendas, cuentas de correo electrónico, agencias de viaje, citas o redes sociales. Aunque es extremadamente difícil obtener la precisión de la información, el valor de los datos procedentes de sensores virtuales son cada vez más valiosos, ya que en la era digital la mayoría de información personal de los usuarios está almacenada en medios digitales como internet o agendas electrónicas.
- *Sensores lógicos.* Es una combinación de los dos anteriores, en los cuales las fuentes de información son los propios sensores físicos instalados en el dispositivo o en el entorno y un conjunto de bases de datos e información digital almacenada en el mismo dispositivo o en un servidor de datos. Los sensores lógicos son capaces de resolver tareas más complejas gracias a la doble naturaleza de los datos que manejan.

En un segundo nivel de la arquitectura anterior se encuentra el *Proveedor de datos RAW*, encargado de obtener la información que brindan los sensores y procesarla para que sea comprensible por el middleware. En la mayoría de los casos esta adaptación pasa por el uso de una ontología o un esquema de datos que permite unificar el formato de la información manejada por el middleware. Esta capa permite además independizar el procesamiento de la información de los sensores concretos, ya que la labor de adaptación de la información traducirá los datos RAW de los sensores a un esquema común. Generalmente este proceso da una sensación de transparencia al usuario frente a las capas inferiores de la arquitectura.

El siguiente elemento de la arquitectura clásica de los middlewares es la capa de *Pre-procesamiento* de la información. En esta capa se realizan todas las labores de filtrado y obtención de información a partir de los datos adaptados procedentes de los sensores. Por otra parte esta capa proporciona un nivel de abstracción aún más alto, de manera que la información generada puede ser manejada por cualquier aplicación que necesite hacer uso de ella. Esto es posible gracias al razonamiento e interpretación de la información básica procedente del nivel inferior.

Hasta el momento la información contextual obtenida está dispersa y no existe ninguna relación entre los distintos parámetros obtenidos del contexto del usuario. La tercera capa de la arquitectura se encarga además de relacionar la información contextual con el fin de obtener nueva información más útil. En la mayoría de los casos esta nueva información se consigue mediante la agregación o la composición de información más básica. Para ilustrar la funcionalidad de esta capa, imagínese un sistema que sea capaz de medir la temperatura ambiente y otro que permita determinar la actividad física que está llevando a cabo el usuario. Ambas informaciones por separado podrían llegar a ser interesantes dependiendo del caso, pero si fuera posible relacionarlas, podríamos generar aplicaciones que fueran capaces de alertar al usuario en caso de que sea detectada una temperatura elevada y el usuario lleve realizando la actividad "correr" demasiado tiempo.

La capa de *Gestión de la información* es la encargada de brindar un punto de acceso al middleware para que las aplicaciones hagan uso de la información contextual generada. En este punto, cuando la aplicación final decide acceder a una determinada información, puede realizarlo siguiendo dos alternativas:

- *Acceso síncrono.* Este tipo de acceso consiste en que la aplicación deberá solicitar la información al middleware, el cual acudirá al módulo de almacenamiento para obtenerla y así devolverla a la aplicación. Este proceso se realiza siguiendo un proceso de bloqueo que mantiene a la aplicación solicitante en espera hasta que recibe el dato del framework. Si la recepción es inmediata no hay ningún problema, sin embargo si es preciso generar la información o esperar a que esta llegue de algún sensor, se produce un bloque de la aplicación que podría afectar al rendimiento y a la experiencia del usuario final.
- *Acceso asíncrono.* Trabaja siguiendo un sistema de suscripciones, de manera que cuando una aplicación necesita un determinado dato, se *inscribe* en la lista de interesados. De esta forma, cuando la información es procesada por el middleware, se mandará una notificación (*callback*) a cada una de las aplicaciones ingresadas en la lista anterior, pudiendo así recoger la información solicitada. Durante el tiempo que transcurre entre la suscripción de la aplicación al evento de difusión y el lanzamiento de éste evento por parte del middleware, la aplicación continua con su funcionamiento habitual, sin que exista ningún tipo de bloqueos.

Por último, en la arquitectura descrita aparece la capa de aplicación, en la cual se encuentra todo el conjunto de aplicaciones que harán uso de la información contextual generada a lo largo del proceso. Desde un punto de vista general, las aplicaciones tan sólo deberán asignar funcionalidad a los eventos generados por el middleware sobre el contexto del usuario. Sin embargo, en la realidad este proceso es más complejo, dado que las aplicaciones pueden introducir una lógica más elaborada que la ofrecida directamente por el framework.

Desde el nacimiento de la computación ubicua y el interés tecnológico de dotar a los sistemas de información relacionada con el entorno del usuario, se han dedicado muchos esfuerzos a intentar representar este entorno de una manera eficaz. El principal problema que poseen las aplicaciones contextuales a la hora de representar el conocimiento, es que la información puede ser compartida y utilizada por un elevado número de sistemas, los cuales no tienen por qué haber generado dicha información. Esto se traduce en que los datos deben ser representados de una manera común que sea descifrable por los diferentes módulos que componen el sistema global. A diferencia de los sistemas ubicuos, en las aplicaciones tradicionales no aparecen estos inconvenientes, ya que la información es obtenida, procesada, almacenada y utilizada por el mismo sistema.

El nacimiento de la web semántica en 2004 marca un antes y un después en la manera de representar el conocimiento. La filosofía de la independencia entre las aplicaciones abre paso a un nuevo concepto de información, que permite compartir el conocimiento entre diferentes sistemas sin necesidad de que estos hayan sido desarrollados específicamente para la manera de representar la información. El objetivo de estos sistemas es generar una información lo suficientemente completa y estructurada para que sea comprensible por todas aplicaciones.

Volviendo a los sistemas contextuales, los primeros intentos de generar información común tuvieron lugar con datos relacionados con la localización y el tiempo. Como se comentó anteriormente, sobre estos dos tipos de contexto se basaron las primeras aplicaciones ubicuas, con lo cual es comprensible que fueran los primeros en experimentar una *estandarización* en su manera de ser compartidos.

Sin embargo, la información contextual es mucho más amplia y no sólo incluye datos relacionados con la localización o el tiempo. Por tanto, la definición de un sistema común para almacenar y compartir los datos contextuales no es una tarea sencilla, ya que debe ser lo suficientemente general para poder representar cualquier tipo de información, pero a la vez lo bastante concreto para no perder conocimiento en la representación. A continuación se exponen los distintos modelos que se han empleado a lo largo de los últimos años para la representación del contexto, clasificándolos según la estructura de datos empleada para compartir la información contextual entre los distintos sistemas que la necesitan.

- *Pares clave-valor*. Es el método de representación más simple para el modelado de la información contextual. Tal es así que se lleva empleado desde hace casi dos décadas (Schilit, 1995) en sistemas contextuales con información de localización y variables ambientales. La principal ventaja de esta estructura es la facilidad de manejo y la rapidez con que puede ser procesada en cualquier dispositivo. Además, debido a su simplicidad las operaciones de búsqueda e inserción de nuevos contextos son computacionalmente sencillas. Sin embargo, presenta un grave problema a la hora de generar estructuras más complejas basadas en la relación entre atributos o la creación de dependencias.
- *Esquemas de marcado*. Esta estructura consiste en la creación de un metamodelo o perfil común basado en una jerarquía de etiquetas que se agrupan en base a la relaciones entre las mismas. La posibilidad de serializar la información permite que pueda ser transmitida mediante un canal de comunicación digital y por tanto, brinda la posibilidad de compartir la información entre diversos sistemas remotos. El lenguaje de etiquetas más común es el *eXtensbile Markup Language (XML)*.

- *Modelo gráfico*. Existen varias herramientas para representar gráficamente información contextual en base a una jerarquía determinada. Todas ellas se basan en el modelado de la información mediante una sintaxis concreta que permite que sea reconocida por todas las aplicaciones que deseen hacer uso de ella. El modelo por excelencia es el *Unified Modeling Language (UML)*, el cual puede ser representado gráficamente mediante multitud de herramientas. El método de representación basado en UML posee un mapeo casi inmediato a los tradicionales modelos de *entidad-relación (ER)*, lo cual permite almacenar de manera rápida y sencilla la información contextual en un repositorio de datos como por ejemplo, una base de datos tradicional (Modeling context information in pervasive computing systems, 2002).
- *Modelo orientado a objetos*. Mediante este modelado, un contexto es una estructura con una serie de propiedades que lo definen y unos métodos o funcionalidades que son capaces de ofrecer. Este sistema dota de un elevado grado de abstracción a la información, de manera que puede ser encapsulada, procesada y empleada mediante la definición de interfaces que la representan. Por tanto, mediante este proceso de abstracción el dato procedente del sensor es convertido en un modelo específico del contexto que representa, de forma que dicho modelo del concepto se mantendrá inalterado frente a cambios en los sensores que proporcionan los datos (How to build smart appliances?, 2001).
- *Modelado basado en reglas*. En ese tipo de modelado, los contextos se representan mediante hechos, expresiones y reglas. Una expresión consiste en la evaluación de un conjunto de hechos a partir de unos valores mediante el uso de reglas o expresiones que los rigen. De esta forma las reglas pueden ser interpretadas como un conjunto de condiciones sobre los datos. Los modelos basados en reglas necesitan de una gran formalidad en la definición de los datos, ya que las reglas tan sólo se pueden aplicar cuando el conjunto de datos está bien definidos. Un ejemplo de sistema basado en este tipo de modelado es GAIA (Gaia: A Middleware Infrastructure to Enable Active Spaces, 2002). Otras soluciones amplían la funcionalidad básica que puede ser llevada a cabo mediante la lógica de primer orden, y usan sistemas difusos (*fuzzy)* para la evaluación de las expresiones o incluso sistemas de aprendizaje más complejos para la toma de decisiones (OPF : A Distributed Context-Sensing Framework for Ubiquitous Computing Environments, 2006).
- *Modelos ontológicos*. Según la definición, una ontología es la *formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados, con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades*[10]. Desde el punto de vista semántico, una ontología permite describir un contexto dentro de un dominio específico de conocimiento. Los modelos ontológicos han experimentado un notable avance en los últimos años en el campo de los sistemas ubicuos, debido en gran medida a su generalización sin pérdida de expresividad, lo que permite construir entornos dinámicos en el que las entidades no están definidas de manera rígida al comienzo. Entre los sistemas contextuales que representan la información mediante ontologías cabe destacar CONON (Ontology based context modeling and reasoning using owl, 2004) y SOUPA (Standard ontology for ubiquitous and pervasive applications, 2004).

MÓDULOS FUNCIONALES

Una vez introducido el trabajo que vamos a llevar a cabo, en esta sección procederemos a dar las ideas fundamentales sobre las que se basa el sistema que vamos a desarrollar. En dicha sección

[10] Definición de Wikipedia

haremos una clara diferencia entre las distintas tecnologías que se emplearán para dar soporte a la obtención del contexto en el que se encuentra el usuario en cada momento. Dichas tecnologías estarán sujetas a la disponibilidad de las mismas en el dispositivo sobre el que se va a ejecutar la aplicación y de la presencia de un módulo de provisión de servicio adecuado, de manera que no todos los dispositivos podrán soportar todas las funcionalidades que serán descritas a continuación. Sin embargo, las tecnologías que se describen a lo largo de esta sección están incluidas en la mayoría de dispositivos de última generación que actualmente se comercializan. Además, tras realizar un estudio sobre cientos de aplicaciones que emplean tecnologías basadas en contexto para la obtención de información, se ha detectado que todos los elementos empleados en estas aplicaciones pueden englobarse en uno de los grupos que serán expuestos en esta sección.

Todo sistema que se base en el contexto para obtener información necesaria en el proceso de ejecución, necesita conocer un conjunto de información que modele el medio que rodea al usuario. Este modelado permite por un lado tomar decisiones de ejecución y por otro lado aprender las decisiones tomadas por el usuario en un contexto determinado como base para posibles decisiones automáticas en el futuro. Por tradición, como se ha presentado anteriormente, es necesario conocer quién realiza la acción, dónde se lleva a cabo, cuándo se ha producido y qué acción se ha realizado. Las siguientes tecnologías nos permitirán obtener respuestas a todas estas preguntas y añadir nueva información que en sistemas tradicionales no podrían ser tenidas en cuenta debido a sus limitaciones.

Además de explicar en qué consiste cada uno de los módulos funcionales identificados en el contexto de aplicaciones ubicuas, definiremos una clasificación que nos permitirá representar la información obtenida por dichos módulos de una forma unificada para todas las aplicaciones que empleen la arquitectura. De esta forma conseguimos una mayor compatibilidad entre aplicaciones lo que al final, llevará a conseguir el objetivo de permitir el intercambio de información de manera automática entre aplicaciones a través del middleware descrito posteriormente.

Posicionamiento en exteriores

En la actualidad existen sistemas muy precisos de localización en exteriores. El más extendido de todos ellos es el sistema de localización GPS. El sistema GPS se trata de un sistema de posicionamiento global que permite determinar en cualquier lugar la posición de un objeto que integre o porte un receptor de señal GPS. El sistema se basa en una red de 27 satélites que se encuentran en órbita alrededor de la tierra a una distancia de 20.200 kilómetros de la superficie.

Para la obtención de la posición de un objeto, en primer lugar, el receptor utilizado para este fin localiza de manera automática al menos tres satélites de la red. Dichos satélites envían una señal al receptor en la que se indica la posición y la hora interna del satélite. Una vez que la señal llega al receptor, éste calcula el retraso de las señales y, mediante un proceso de triangulación, calcula la posición en la que se encuentra.

Como queda patente debido al funcionamiento del sistema, el posicionamiento basado en GPS necesita una visión directa de los satélites para su correcto funcionamiento, por lo que en interiores el sistema no es válido. A pesar de este inconveniente, la cobertura global y su precisión hacen de este sistema el más utilizado a nivel mundial. Sin embargo, el principal inconveniente de este sistema es el elevado gasto energético del receptor GPS, además de la limitación inherente de ser inválido en interiores. El tema energético cobra especial importancia en sistemas embebidos en dispositivos móviles.

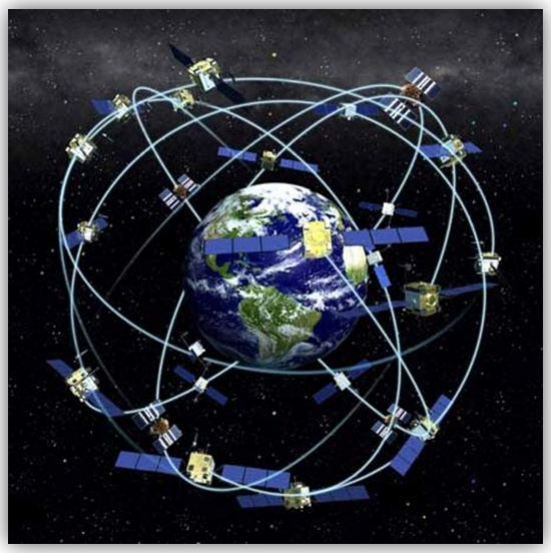


Figura 13: Ilustración de la red de satélites GPS

En la actualidad, existen sistemas de detección de entradas y salidas a exteriores (Soria-Morillo, y otros, 2009). Esto hace posible desconectar el dispositivo GPS cuando el usuario se encuentra en el interior de un lugar techado y volver a conectarlo cuando sale de él. Esto hace que las baterías del dispositivo móvil aumenten su tiempo de uso en un 230% entre carga y carga. El aumento masivo del tiempo de uso se debe a que según un estudio realizado en esta tesis, el tiempo medio de permanencia en interiores de una persona es del 93.4% del día (dejamos fuera de este supuesto a trabajadores relacionados con el transporte y otros colectivos cuyo trabajo habitual se realice en exteriores).

Este sistema de detección de salidas es especialmente útil en sistemas de seguimiento y posicionamiento constantes, ya que en ellos se intenta obtener la posición del usuario en todo momento. Además de lo anterior, el posicionamiento en sistemas basados en contexto posee una gran importancia tal y como comentábamos en la sección anterior, por lo que un ahorro energético en dicho sistema tendrá un impacto considerable en el ahorro energético global del sistema.

Sin embargo, el sistema GPS no es el único método de posicionamiento que existe en la actualidad. En 1999 aparece la idea de localización basada en identificación de celdas (GSM standards activity on location, 1999). Esta técnica emplea la señal emitida por las torres de difusión de señal telefónica para triangular la posición del usuario. Las apariciones de teléfonos móviles inteligentes en la última década, ha dado a esta técnica de posicionamiento una gran importancia.

Sin embargo, existen varios problemas que hacen de este método un sistema poco eficiente. En primer lugar se encuentra la precisión, ya que en determinados lugares, la distancia entre torres de telecomunicaciones es de más de 35 km, por lo que podríamos realizar una localización a nivel de áreas, pero en ningún momento a un nivel de detalles de calles. Otro problema existente es la necesidad de modificar el sistema de control de las propias torres de telecomunicación, ya que para conseguir un nivel de precisión adecuado se debe conocer el retraso de la señal producido entre la emisión por parte del dispositivo móvil y la recepción por parte de la torreta de telecomunicación.

A continuación presentaremos seis métodos distintos de posicionamiento basado en identificación de celdas (Laitinen, 2001) (Database correlation method for GSM location, 2001):

- Cell-ID: se trata del método clásico de posicionamiento GSM. Consiste en la detección del identificador de la celda GSM correspondiente al lugar donde se encuentra el dispositivo móvil. Una vez ha sido obtenido el identificador, se consulta una base de datos[11] en la cual están recogidos la mayoría de torretas GSM junto a su localización medida en coordenadas de latitud y longitud, así como los identificadores de las celdas generadas por dicha torreta. De esta forma el sistema es capaz de conocer la posición del dispositivo con un margen de error dependiente de la distancia entre torretas, aunque suele ser de unos 900 metros en ciudades y 30 kilómetros en zonas periféricas. Las ventajas de este método son el bajo coste necesario tanto computacional como temporal, es fácilmente implementable en cualquier dispositivo y el nivel de cobertura es relativamente elevado. Sin embargo, la principal desventaja es la precisión que anteriormente citábamos.

- Nivel de señal: este método extiende al anterior e introduce una nueva variable en el sistema que es la potencia de la señal. Empleando diferentes modelos de propagación, es posible determinar en base a la potencia inicial de la señal y la final, la distancia a la que se encuentra el receptor (en este caso el dispositivo móvil) del emisor (la torreta de comunicaciones). De esta forma, es posible proporcionar una distancia estimada al lugar donde se encuentra la torreta de comunicaciones, dando así al sistema una precisión que depende en gran medida del modelo de propagación empleado en función de las características del entorno. Las ventajas siguen siendo las mismas que para el método anterior, y el inconveniente principal sigue siendo la precisión, ya que siguiendo este método tampoco se consigue un aumente demasiado cuantitativo de la exactitud del posicionamiento.

- Diferencia temporal de llegada (TOA): este método consiste en la medición del tiempo empleado por la señal en viajar desde la torre de comunicaciones hasta el dispositivo móvil. Esto se realiza gracias a la transmisión de la hora de la torre en la propia trama GSM, lo cual, mediante triangulación hace posible un posicionamiento suficientemente preciso. Sin embargo, esta técnica se encuentra con un gran problema que es la sincronización entre relojes. Las señales del reloj de las distintas torres de comunicación no tienen por qué estar sincronizados, de modo que el proceso de triangulación y cálculo de la posición podrían sufrir graves aberraciones relacionadas con esta característica. Por tanto, la principal desventaja de este método sería el coste requerido para la sincronización de los relojes en las torres de telecomunicación, el cual debería ser sufragado por la propia operadora telefónica.

- Diferencia temporal observada en el enlace (E-OTD): en este caso se unen las características del método TOA y el de Cell-ID, dando lugar un sistema híbrido capaz de posicionar al dispositivo sin necesidad de abordar ningún cambio en el lado de las torres de telecomunicación. Este sistema dota a la red de comunicaciones de un medio de pseudo-sincronización basado en el envío de una trama de sincronización a una torre de comunicación intermedia. A dicha trama se le conoce como LMU, la cual permite al dispositivo conocer la diferencia temporal entre los relojes de las torres de comunicación reales. Se debe tener en cuenta que la torre intermedia de telecomunicación es un elemento adicional de la red, por lo que el coste de la localización seguiría siendo elevado. Sin embargo, existen algunos trabajos en los que se describe un sistema alternativo basado en E-OTD sin necesidad de tramas LMU

[11] http://www.opencellid.org

(LMUs, Mobile location without network-based synchronization or how to do E-OTD without, 2003), por lo que el coste del sistema de posicionamiento sería mucho menor.

- Técnicas híbridas: las técnicas descritas anteriormente pueden ser empleadas al mismo tiempo con el fin de obtener un sistema de posicionamiento más preciso y con el menor coste posible. En la actualidad, la técnica de posicionamiento híbrida más empleada es TDOA + AOA, que consiste en la fusión entre el método de diferencia temporal de llegada y el de ángulo de llegada. Este segundo método no es más que la determinación del ángulo de incidencia de la señal en el dispositivo, por lo que es posible conocer la distancia a la torre de la cual procede dicha señal mediante técnicas geométricas.

Por último existe otro método de localización surgido en los últimos años basado en la tecnología inalámbrica Wifi. El proceso empleado para la localización es semejante al detallado anteriormente para las redes de datos GSM, sólo que en este caso se usarán puntos de acceso Wifi como emisores.

La tecnología Wifi, en la que sus balizas son los puntos de acceso a internet inalámbricos, posee una precisión muy elevada en grandes ciudades y en general, en cascos urbanos muy poblados. Además, la cobertura de este sistema se amplía día a día gracias a la creación de nuevos puntos de accesos Wifi por parte de los usuarios y al geoposicionamiento de los mismos. Según un informe realizado en 2009 por el Grupo Gowex de servicios de telecomunicación, el número de redes Wifi instaladas en España a finales de 2008 era de 4891 y en Estados Unidos este número ascendía a los 69180 puntos de acceso.

Aunque el posicionamiento basado en redes Wifi es una buena alternativa, no es viable en lugares con baja densidad de redes por habitante. Además, incluso con una densidad media, el posicionamiento no es lo suficientemente exacto como para registrar un diario de las rutas realizadas por el usuario.

El proceso consiste en primer lugar en obtener una base de datos de puntos de acceso geoposicionados, es decir, asociar el identificador de un punto de acceso inalámbrico a unas coordenadas espaciales concretas. Una vez que se ha creado dicha base de información, el dispositivo móvil obtendrá las redes que le son “visibles” en un momento dado. A partir de ellas, mediante un proceso de triangulación basado en el nivel de señal, es posible determinar con una precisión variable la posición del usuario. Dicha precisión se basará, como hemos dicho anteriormente, en la densidad de redes Wifi que existan en el lugar donde se encuentre el dispositivo.

Etiquetado semántico de lugares

Debido principalmente a la escasa seguridad de los métodos tradicionales de almacenamiento de la información obtenida acerca de la posición del usuario, es necesario desarrollar un nuevo modelo. Este modelo se basa en el etiquetado semántico de lugares, a diferencia de los métodos tradicionales de almacenamiento que guardan la latitud y la longitud del lugar donde se encuentra el usuario. El etiquetado semántico de lugares introduce una fuerte componente de subjetividad a la información, por lo que tan sólo el usuario y los individuos que formen parte de su entorno podrán entender dicha información.

El ejemplo más simple de etiquetado de lugares es el almacenamiento de la posición donde se encuentra el hogar del usuario. Mediante los métodos tradicionales se almacenaría la latitud y la longitud, dando una información objetiva y totalmente precisa del lugar donde habita el usuario. Sin

embargo, mediante el etiquetado semántico, la información que se almacenaría sería simplemente "casa", por lo que la información no tendría validez para cualquier otro individuo que no conozca la ubicación de la casa del usuario.

Sin embargo, aunque conceptualmente la idea parezca sencilla, el verdadero problema se encuentra a la hora de etiquetar los lugares. Algo tan sencillo como saber que el lugar donde está el usuario es su casa, conlleva un complejo estudio de los hábitos del usuario y sus tendencias. Precisamente esto es de lo que se encarga el paquete de trabajo de inducción semántica del contexto.

A grandes rasgos, el etiquetado de lugares se basará en la obtención de lugares de interés cercanos y en función de la hora, la frecuencia de visitas, la compañía y otros parámetros, podremos generar un perfil del lugar. A partir de este perfil y gracias a diversos métodos de comparación de características, podrá ser determinado el tipo de lugar en el que se encuentra el usuario, es decir, si el individuo está en el trabajo, en casa, de compras en un centro comercial o cualquier otra posición. Todo esto dará lugar a un subsistema capaz de posicionar al usuario en un lugar entendible por otras personas con un cierto grado de relación con el usuario a partir de información semántica y de contexto, sin necesidad de trabajar con coordenadas objetivas que pondría en entredicho la seguridad del sistema si se produce una filtración de los datos del usuario.

Posicionamiento en interiores

Las técnicas actuales de posicionamiento en interiores resultan de utilidad en espacios sensorizados, ya que la mayoría de ellas se basan en etiquetas RFID o sensores de proximidad para determinar la posición del usuario en un lugar techado. Sin embargo, este tipo de técnicas de posicionamiento resultan caras y complejas dado que se precisa de una infraestructura para su funcionamiento. Existen otros trabajos que basan su posicionamiento en la triangulación de redes Wifi. El principal problema de este tipo de soluciones es que los puntos de acceso Wifi deben ser de una calidad elevada con el fin de eliminar fluctuaciones de intensidad de señal.

Por otro lado, en los últimos años la tecnología inalámbrica RFID ha sido objeto de investigación para su uso como sistema de posicionamiento. La tecnología RFID es un sistema que permite almacenar y recuperar información de manera remota empleando un conjunto de dispositivos electrónicos llamados etiquetas. Tradicionalmente, este tipo de elementos son empleados para reconocer objetos y recuperar información sobre ellos, ya que surgió como un sustituto a los antiguos códigos de barras. Sin embargo, este uso ha dado paso a un verdadero sistema de posicionamiento alternativo, el cual permite identificar la posición del usuario en base a los objetos etiquetados mediante etiquetas RFID que se encuentran a su alrededor. En este campo existen numerosos trabajos (The study for the RFID with bluetooth positioning system, 2010) (RFID indoor positioning using RBFNN with L-GEM, 2010) (PBIL PDR for scalable Bluetooth Indoor Localization, 2009) y varias empresas que han basado su actividad tecnológica en este ámbito[12].

Por último, existe otro método bastante extendido de posicionamiento en interiores denominado *Dead Reckoning* o navegación por estima (Positioning using Acceleration and Moving Direction, 2008) (Bayesian indoor positioning systems, 2005). En este caso, las técnicas para la detección de la posición del usuario se basan en el uso del acelerómetro para la detección del paso y en algunos casos, la longitud del mismo, y por otro lado se emplea el giroscopio o la brújula, los cuales ofrecen información sobre el rumbo que se ha seguido. Aunque se trata de un método poco preciso, ofrece

[12] http://www.smartoxide.com/

buenos resultados en lugares donde otras alternativas como el posicionamiento Wifi o basado en RFID no es posible.

Predicción de destinos

La predicción de destinos es un tema de gran actualidad que consiste en la determinación del destino más frecuente en base a un trayecto recorrido previamente por el usuario (Trip Destination Prediction Based on Past GPS Log Using a Hidden Markov Model, 2010). En general este tipo de sistemas se basan en el análisis de recorridos anteriores y la extracción de puntos característicos del recorrido que puedan estar asociados a posibles destinos o paradas intermedias en el trayecto del usuario. Una vez realizado este proceso, se clasifica la información y se efectúa un estudio.

Dado que el comportamiento de la mayoría de las personas en sus desplazamientos es repetitivo y regular, es decir, nos dirigimos normalmente a los mismos destinos utilizando las mismas rutas, las necesidades para realizar predicciones en cuanto a mapas y cartografía no son las mismas que en un sistema de navegación y guiado de vehículos. Por tanto, en este tipo de sistemas es habitual propuestas sobre modelos predictivos utilizando técnicas de aprendizaje supervisado basadas en mapas personales generados por los propios usuarios. Estos mapas personales permiten una drástica reducción en el coste de almacenamiento con respecto a los Sistemas de Información Geográfica (GIS) y, lo más importante, su especialización (Trip Destination Prediction Based on Past GPS Log Using a Hidden Markov Model, 2010).

Además en este aspecto, determinando por anticipado la ruta que el usuario va a seguir, es posible realizar múltiples aplicaciones (A Markov model for driver turn prediction, 2008), (Where will they turn: predicting turn proportions at intersections, 2009). En esta sección se recogen temas como el almacenamiento de rutas finalizadas, la extracción de destinos frecuentes a partir de las rutas almacenadas, emisión de destinos probables (en base a criterios de similitud entre rutas) y segmentación de rutas (para la obtención de puntos singulares y destinos intermedios).

Luminosidad

Los sensores de luminosidad embebidos en los dispositivos móviles de última generación permiten obtener la intensidad lumínica del lugar donde se encuentra el usuario. De hecho, este tipo de sistemas están ampliamente extendidos con el fin de reducir al máximo el gasto energético mediante el control automático del entorno en sistema domóticos (A Study on Saving Energy in Artificial Lighting by Making Smart Use of Wireless Sensor Networks and Actuators, 2009). Por otro lado, gracias a este valor de intensidad es posible averiguar si el usuario se encuentra en un lugar expuesto al sol, bajo la luz de un fluorescente, en la calle frente a una farola o dando un paseo nocturno. Esta información recopilada por el middleware permitirá dar a conocer a las aplicaciones basadas en contexto las características de iluminación del lugar donde el usuario se encuentra.

En cierta medida puede que la luminosidad no aporte excesiva información de contexto a las aplicaciones, pero unida a otras características de contexto es posible obtener información muy útil con el objetivo de ejecutar sistemas disparados por las condiciones del lugar que rodea al usuario.

Además, del propio nivel de luminosidad, este módulo funcional podrá aportar información adicional como el parpadeo, la tendencia de luminosidad u otros aspectos relacionados con los fotosensores instalados en los dispositivos móviles.

Detección de personas

La detección de personas cercanas al usuario abre un abanico de posibilidades al sistema ubicuo y por tanto, las funcionalidades del middleware que se pretende desarrollar se disparan. Un sistema de detección de personas hace que la aplicación sepa en todo momento los individuos conocidos que rodean al usuario, incluso es capaz de almacenar información sobre nuevas personas que no habían interactuado anteriormente con el usuario del sistema.

Para realizar una detección de personas cercanas al usuario se emplearán dos tecnologías distintas aunque complementarias:

- Detección basada en sonido. Este tipo de detección se basa en la segmentación y el análisis de las conversaciones del usuario. Con este análisis se obtienen características sonoras de las voces de las personas que interaccionan con el usuario mediante la obtención del sonido a través del micrófono del dispositivo. Gracias al uso de GMMs (Gaussian Mixture Models) es posible segmentar y clasificar una determinada conversación (Unsupervised speaker segmentation and tracking in real-time audio content analysis, 2005), (Universal Background Models for Real-Time Speaker Change Detection, 2003) en función de los usuarios participantes en ella. Gracias al almacenamiento de las características, es posible reconocer a la persona la próxima vez que el usuario interaccione con ella. Este método elimina totalmente la interacción del usuario con el sistema y es éste último el encargado de determinar quién realiza la acción que se está llevando a cabo y qué usuarios han participado en ella. Esto se traduce en que poseemos una nueva información sobre el contexto que rodea el dispositivo, con el fin de ofrecer esta nueva información para que las aplicaciones que trabajen sobre el middleware que se presenta puedan hacer uso de ella.

- Detección basada en señales Bluetooth. Gracias a la búsqueda de dispositivos Bluetooth, es posible detectar usuarios cercanos sin necesidad de interacción directa con ellos. Esto hace posible que el sistema alerte de la presencia de una persona previamente conocida cerca del usuario del sistema.

Temperatura

La determinación de la temperatura exterior también es de interés para conocer el contexto de ejecución de una determinada aplicación. Esta característica del contexto cobra especial interés en aplicaciones cuyo objetivo es aumentar la eficiencia energética en entornos domóticos. Conociendo la temperatura exterior que rodea al usuario es posible adaptarla con el fin de aumentar el bienestar del usuario y a la vez disminuir el consumo energético gracias a la determinación de características óptimas del ambiente.

Además de aplicaciones centradas en entornos inteligentes, es posible disparar aplicaciones en función de la temperatura que rodea al dispositivo, e incluso mostrar información al usuario sobre actividades a realizar en función de dicha temperatura.

Existen varios trabajos relacionados con el campo de e-Health que permiten conocer el estado de forma de un usuario en función de la actividad llevada a cabo por éste. Sin embargo, añadiendo el parámetro de temperatura no sólo es posible conocer el estado de forma, sino basarse en la temperatura y la actividad que está realizando el usuario para mostrar consejos relacionados con la salud como, por ejemplo, no realizar demasiada actividad física en momentos en los que la temperatura sea elevada.

Humedad

La humedad es otro dato que puede resultar de interés para determinados sistemas ubicuos. Por este motivo, se ha decidido identificar la información proporcionada por este sistema para que pueda ser compartida entre distintas aplicaciones de una manera sencilla.

Sin embargo, hoy en día son pocos los dispositivos que incorporan sensores de humedad y aquellos que lo hacen, suelen tener fines muy específicos y en ningún caso son dispositivos móviles de uso general. Sin embargo son muchos los sensores externos[13] que permiten ser conectados al dispositivo mediante tecnología inalámbrica, como por ejemplo Bluetooth, y ofrecer información sobre la humedad presente en el entorno del dispositivo.

Campo magnético

En los dispositivos móviles de última generación es habitual la presencia de un magnetómetro digital, el cual permite medir los cambios en el campo magnético producido en los tres ejes del espacio. Esta medición permite además obtener información sobre la orientación del dispositivo, por lo que habitualmente se emplea de manera conjunta con el acelerómetro. Esta orientación es el proceso a través del cual es posible obtener un sistema de brújula digital en estos dispositivos.

Analizando las capacidades de estos sensores, los desarrolladores han comenzado a aprovechar los datos provenientes de estos de forma masiva para la creación de aplicaciones para los diferentes sistemas móviles. Tal es el caso de, Layar Reality Browser[14], desarrollada para equipos con sistema operativo Android. Dicha aplicación permite superponer información extraída de diferentes enciclopedias digitales al apuntar un dispositivo móvil a un lugar como un museo, una plaza, un restaurante o cualquier locación que pudiera tener documentación en Internet.

Audio

En algunos casos es interesante analizar la señal de audio proporcionada por el micrófono ambiental del dispositivo o bien por el micrófono vocal. Esto permite funciones relacionadas con el usuario como por ejemplo el reconocimiento de voz, identificación de usuarios por voz o manejo automático mediante lenguaje natural.

Por otro lado, el análisis del audio ambiental es un campo actual de investigación en el área de aplicaciones contextuales. Dichas investigaciones se basan en el análisis del sonido para la determinación de ciertas características del entorno del usuario, como por ejemplo el nivel de ruido, características concretas de algunos escenarios o identificación de lugares por huella sonora.

Reconocimiento de actividades

La actividad que está llevando a cabo el usuario también posee un gran valor para determinar el contexto de ejecución actual. Cuando hablamos de actividad nos referimos a la acción que está llevando a cabo el usuario a partir del análisis de los valores obtenidos a partir de sensores de acelerometría.

Mediante el aprendizaje y la clasificación de una serie de parámetros obtenidos desde el acelerómetro del dispositivo, el sistema es capaz de clasificar la actividad que el usuario está

[13] http://www.kestrelmeters.com/Kestrel-4500-Weather-Meter.pro
[14] http://www.layar.com/

realizando. Esto lo llevamos a cabo empleando un sistema de clasificación basado en tablas de probabilidad y árboles de decisión y será explicado en destalle en el capítulo 4.

En este aspecto se ha desarrollado un método de detección de actividades que permite clasificar con un porcentaje de acierto de un 95% la actividad que se lleva a cabo. Además, mediante este sistema es posible añadir nuevos perfiles de actividad al sistema sin que esto implique una reprogramación de la plataforma. De esta forma es posible detectar nuevas actividades que en principio no estaban establecidas con tan solo añadir una nueva tabla de probabilidades al sistema o mediante un reentrenamiento de la nueva actividad, en función de la técnica de reconocimiento elegida.

Además de este sistema de reconocimiento de actividades, se ha desarrollado toda una plataforma para la publicación de la información de las actividades reconocidas. Gracias a esta plataforma es posible realizar un seguimiento de la actividad llevada a cabo por un determinado usuario mediante una interfaz web.

ARQUITECTURA OSGI

OSGi (acrónimo de Open Services Gateway Iniciative) es un estandar de sistemas de módulos dinámicos para Java nacido en marzo de 1999 con el objetivo de definir especificaciones abiertas de software de cara al diseño de plataformas compatibles para proporcionar servicios. Más concretamente, OSGI define una plataforma de intercomunicación entre módulos (llamados bundles), dejando la comunicación en manos de la plataforma.

La tecnología OSGi proporciona a los desarrolladores un entorno orientado a servicios y basado en componentes y ofrece métodos estandarizados para la gestión del ciclo de vida del software. Estas capacidades aumentan en gran medida el valor de una amplia gama de equipos y dispositivos que utilizan la plataforma Java.

Principalmente, fue pensado para la aplicación en redes domésticas, pero a lo largo de su historia ha visto como se iba ampliando su campo de acción, gracias a la gran compatibilidad de los servicios ofrecidos. De hecho, pese a que OSGi define su propia arquitectura está diseñada para ser compatible con Jini o UpnP.

La gestión relacionada con OSGi se realiza a través de OSGi Alliance, un gran número de grandes empresas las cuales se encargan de:

- El mantenimiento y la difusión de la especificación OSGi.
- Certificación de implementaciones.
- Organización de eventos

Generalmente, se entiende por OSGi al conjunto del sistema de módulos y la plataforma de servicios que implementa el modelo de componentes completo y dinámico. Algo que, a partir de 2011, no existe de manera nativa en Java y entornos de VM independientes. Sin embargo, el sistema define la plataforma de intercomunicación y, anexo a estas, se generan los módulos que definirán la aplicación propiamente dicha. Otra de las ventajas de OSGi es que varios módulos distintos pueden interaccionar entre ellos sin que exista la necesidad de definir un canal específico de comunicación para cada una de estas interacciones, lo que simplifica el desarrollo de aplicaciones y el espacio ocupado por las mismas. Por otro lado, la plataforma se ejecuta en background observando los bundles

que se conectan, de manera que cuando un bundle se instala en el dispositivo, éste lo reconoce y avisa a todos los bundles implicados. De esta manera, las aplicaciones o componentes de las mismas se pueden instalar de manera dinámica, iniciar, detener, actualizar y desinstalar sin necesidad de reiniciar el mismo. La gestión del ciclo de vida de las aplicaciones (inicio, parada, instalación, etc) se realiza a través de APIs que permiten la descarga remota de políticas de gestión. El registro permite a los paquetes de servicio detectar la adición de nuevos servicios, o la supresión de los mismos, y adaptarse en consecuencia.

Debido a su larga historia, las especificaciones OSGi se han movido más allá del enfoque original de gateways de servicios, y ahora se utilizan en aplicaciones que van desde teléfonos móviles hasta el de código abierto. Otras áreas de aplicación incluyen automóviles, automatización industrial, automatización de edificios, PDA's, computación grid, gestión de flotas y servidores de aplicaciones.

Por lo tanto, el framework OSGi ofrece la interfaz de comunicación perfecta para el proyecto, gracias a su dinamismo, que permitirá añadir módulos en función de las necesidades del cliente sin reiniciar el servicio, a que está diseñado para Java, plataforma muy similar a la del Sistema Operativo, y simplificará la comunicación necesaria entre módulos. Por otro lado, dado su largo recorrido, se trata de una plataforma fiable y reconocida en la que ya confían miles de desarrolladores y de la que existe multitud de información relacionada.

ARQUITECTURA

OSGi define una arquitectura por capas en su estándar, de manera que cada una de estas capas gestionará una serie de servicios proporcionados por la capa inferior, los cuales se emplearán para ofrecer funcionalidades a la capa inmediatamente superior. En el caso de la última capa, esta funcionalidad será ofrecida a la aplicación cliente, es decir, a aquellas aplicaciones que hacen uso del framework proporcionado por *OSGi* para gestionar los diferentes módulos del sistema desarrollado. Esta arquitectura se puede observar en la Figura 14.

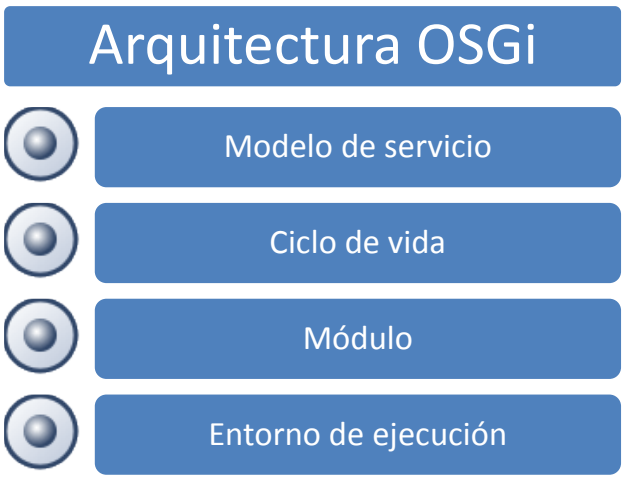


Figura 14: Diagrama de la arquitectura de la plataforma OSGi para la intercomunicación de aplicaciones

- *Modelo de servicio*. En esta capa se lleva a cabo toda la intercomunicación entre módulos. Cuando un módulo se instala en el dispositivo, este se puede definir como proveedor de servicio (*provider*) o solicitante (*requester*). En el primer caso, el bundle informará al

sistema que se trata de un *service provider* a través de la publicación de su interfaz (*Publish service*) en el *service registry*. Cuando se trate del segundo caso, es decir, el bundle pretende obtener información de otros bundles del sistema, el servicio realizará una búsqueda en el *service registry* a partir de la interfaz solicitada (*Service find*).

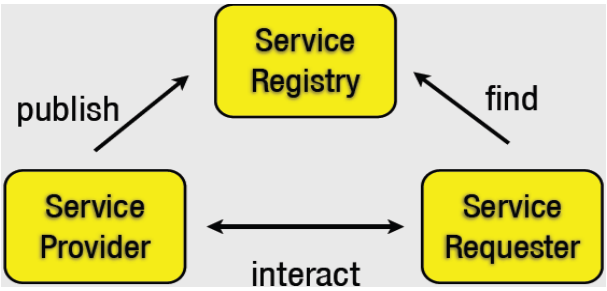


Figura 15: Arquitectura genérica de la capa de modelo de servicio del framework de OSGi

- *Ciclo de vida*. En esta capa se gestiona el ciclo de vida de los diferentes bundles que conforman el sistema. Dicha capa es realmente importante para realizar la autogestión de los módulos, ya que en ella se llevan a cabo las funciones de instalación, desinstalación, lanzamiento y parada de los bundles del sistema. Resulta igualmente importante para el proyecto, ya que esta capa permite la instalación dinámica de bundles y dará al mismo una característica tan importante como es la instalación "en caliente" y la ampliación dinámica del software. Además de todo lo anterior, a través del ciclo de vida es posible informar al resto de módulos del sistema acerca de las operaciones que se realizan sobre los bundles del sistema. En la Figura 16 se representa un diagrama de flujo que contiene los principales estados del ciclo de vida de los bundles *OSGi*.

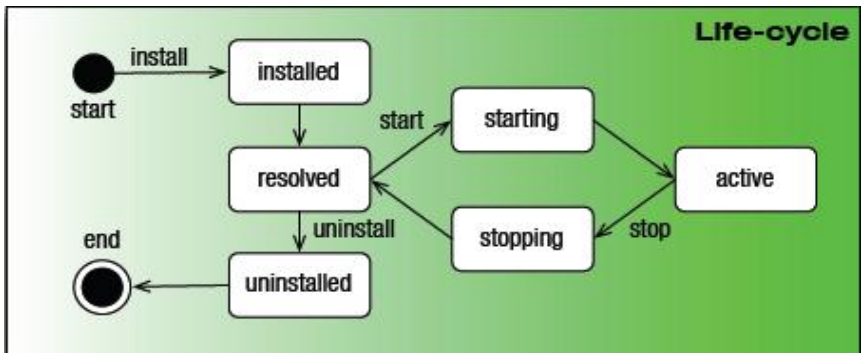


Figura 16: Arquitectura genérica de la capa de ciclo de vida del framework de OSGi

- *Módulo*. En esta capa se definen los módulos o bundles que podrán ser empleados a lo largo de la ejecución de las aplicaciones. A este nivel, cada módulo o bundle será independiente de los demás, dejando la comunicación entre ellos para las capas superiores de la arquitectura. Gracias a esta capa es posible realizar una separación entre los distintos bundles, por lo que la implementación de los mismos puede llevarse a cabo de manera independiente. Es precisamente esta funcionalidad la que hace del framework *OSGi* un método ampliamente empleado para el desarrollo de sistemas modulares. En la Figura 17 se puede observar una representación gráfica del sistema de módulos empleados por *OSGi*.

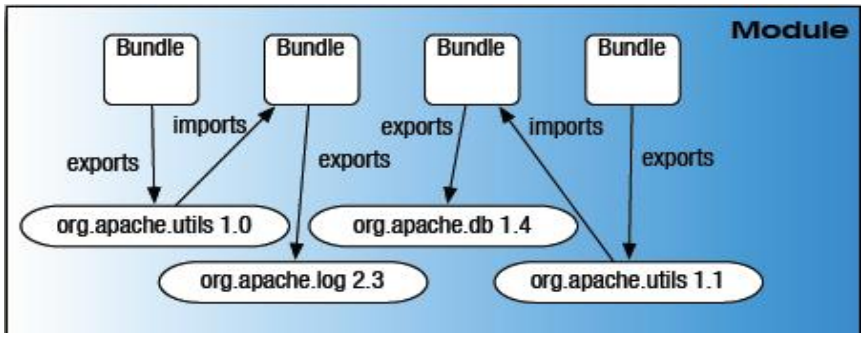


Figura 17: Arquitectura genérica de la capa de módulos del framework de OSGi

- *Entorno de ejecución*. Un bundle puede crear un objeto y registrarlo en una o más interfaces dentro del *Service Registry*. Del mismo modo, otro bundle puede acceder al *Service Registry* y buscar los que estén registrados sobre una determinada interfaz o clase. Un bundle puede incluso esperar un determinado servicio (Observación o listen) y actuar en consecuencia cuando el framework de *OSGi* lo obtenga. Una vez se obtenga el bundle al que se desea acceder para obtener información, se establece el 'bind' entre el *service provider* y el *service requester*. Llegados a este punto existen varias formas de comunicación posibles:

    - Mediante funciones estáticas: es la manera más simple. Únicamente es necesario que un bundle exporte el paquete contenedor y otro lo importe.

    - Mediante Servicios: un servicio es un objeto registrado en el framework por un bundle, aunque puede ser implementado por varios. Es recomendable definirlos usando interfaces y pueden ser encontrados dinámicamente. También es necesario que un bundle exporte el paquete contenedor y que el otro lo importe, que se registre y se busque.

    - Eventos: los eventos únicamente pueden ser generados por el framework. Estos pueden ser de tres tipos: Bundle, framework y servicio. Para escucharlos, se debe implementar una interfaz y suscribir al bundle observador en una lista de listeners.

    - Wires: Es la manera más común de implementar la comunicación entre bundles. Un wire (cable) se asocia entre dos servicios, un proveedor y un consumidor. Un consumidor puede hacer polling a un wire, lo que permite que este acceda al objeto asociado. Por contrario, un productor puede ejecutar un update y actualizar el objeto asociado.

IMPLEMENTACIONES

Existen varias opciones de implementaciones de OSGi. Algunas de las empresas que ofrecen frameworks para esta plataforma son: Prosyst (con su framework mBS), wakewave (con su plataforma knopflerish pro) o Siemens (con varias plataformas como, por ejemplo, siemens HiPath 8000 Assistant V2.0), todas ellas plataformas propietarias. Por otro lado, se ofrecen igualmente alternativas de software libre, entre las que se destaca Apache Felix.

Según su propia descripción, *Apache Felix es un esfuerzo comunitario para implementar la Plataforma de Servicios OSGi R4 y otras tecnologías relacionadas con OSGi bajo la licencia Apache*. Esta herramienta es de código abierto y de gran uso por parte de los desarrolladores, lo que ofrece a la plataforma una estabilidad adicional. Además, Apache Felix está desarrollado en Java, por lo que se puede implementar igualmente para Android o cualquier sistema operativo que soporte el uso de aplicaciones Java. Basado en el elevado número de dispositivos móviles que emplean o soportan el sistema Java, será el framework empleado para llevar a cabo la implementación del middleware para el desarrollo de aplicaciones contextuales desarrollado en esta tesis.

Dentro de los sistemas *OSGi* para móviles, existen varios frameworks adaptados que permiten la ejecución de la plataforma sobre el dispositivo móvil, pero la mayoría de ellos consumen demasiados recursos y su funcionalidad está demasiado alejada de los objetivos de esta tesis. Ejemplos de estos middlewares son MUSIC, mBS Mobile o EZ droid.

Otro inconveniente de estos sistemas es que necesitan la instalación y configuración de un software adicional, de manera que sirva de plataforma para las aplicaciones que se instalen sobre *OSGi*. Esto crea una serie de necesidades que deben ser realizadas por el usuario previamente al uso del sistema de implementación del estándar de *OSGi*:

- Instalación de una aplicación adicional
- Configuración manual de la misma
- Dependencia del middleware sobre el que se ejecutan las aplicaciones
- Consumo excesivo de recursos

Por todo esto, se ha decidido realizar un servicio que permita ejecutar la máquina de *OSGi* sobre el dispositivo. Este será el encargado de lanzar el framework de OSGi en la plataforma y detenerlo cuando sea necesario. En este servicio se hará uso de la implementación de las especificaciones de OSGi provistas por Apache Felix.

IDENTIFICACIÓN DE NECESIDADES

La arquitectura propuesta en esta tesis pretende dar solución a los problemas más frecuentes a nivel de intercambio y gestión de contexto entre aplicaciones. Dichos problemas se suelen dividir en 6 grandes grupos en función del área específica del tratamiento de información contextual a la que hacen referencia, las cuales se presentan en la Figura 18. Mediante el análisis de las 6 capas identificadas, es posible determinar las necesidades de cada una de ellas, pudiendo de esta forma elaborar un middleware que contenga todas las características necesarias. De esta forma, basándose en la implementación de un middleware que soporta todas las funcionalidades de los estratos identificados, una determinada aplicación será capaz de realizar todas las tareas necesarias identificadas en las aplicaciones basadas en contexto. A su vez, cada una de las capas recopiladas para las aplicaciones contextuales se compone de uno o más módulos, los cuales contendrán cierta funcionalidad específica que por razones de aislamiento se ha considerado que deben formar parte de grupos independientes.

No debemos perder de vista en ningún momento que el motivo de esta sección es la identificación, de manera jerárquica, de las diferentes necesidades que todas las aplicaciones contextuales existentes en la actualidad emplean durante su ejecución. Como es obvio, posiblemente algunas aplicaciones no harán uso de ciertas funcionalidades, aunque la Figura 18 podría ser entendida como una agrupación de funcionalidades genéricas presentes en las aplicaciones contextuales.

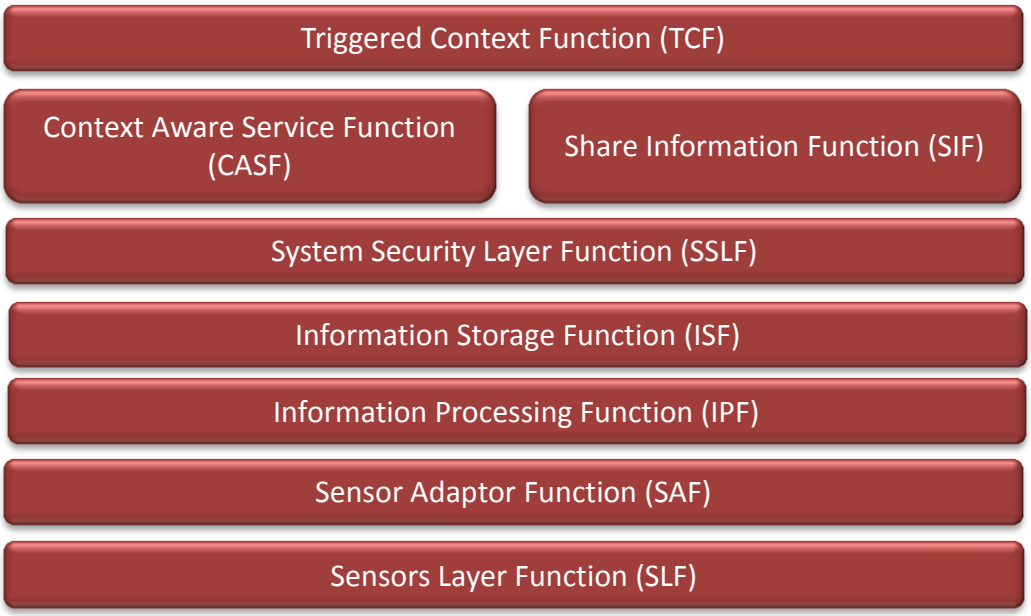


Figura 18: Diagrama de la funcionalidades identificadas en las aplicaciones contextuales

A continuación se realizará una breve descripción sobre cada una de las capas de necesidades identificadas, con el fin de ofrecer una idea inicial de los objetivos que se perseguirán a lo largo de la realización del middleware que pretender cubrir dichas funcionalidades. Debemos tener en cuenta que la arquitectura debe poseer dos requisitos indispensable para que sea viable a la hora de desarrollar aplicaciones reales. El primero de ellos es que debe poder ofrecer unas capacidades que cumplan con las necesidades de cualquier aplicación contextual, lo cual requiere que las capas sean suficientemente genéricas y ampliables. Sin embargo, debemos tener en cuenta que el middleware debe ser un medio de interconexión entre aplicaciones y contextos, por lo que la funcionalidad del mismo debe estar totalmente definida.

Como veremos cuando analicemos en profundidad cada una de las capas, las principales ventajas que aporta esta arquitectura es por un lado, la independencia de las especificaciones concretas de los sensores empleados y por otro lado, la posibilidad de compartir la misma información contextual entre varias aplicaciones, con el consecuente ahorro temporal y computacional derivado de la reutilización de las funcionalidades.

A continuación, se procede a describir cada una de las capas de necesidad identificadas como resultado de realizar un estudio sobre multitud de aplicaciones contextuales, tanto a nivel conceptual y prototipo como a nivel comercial, muchas de las cuales son empleadas por una gran mayoría de usuarios de dispositivos móviles de última generación.

FUNCIONES DE SENSORIZACIÓN (SLF)

La información contextual puede provenir de sensores virtuales o sensores físicos. Los primeros no son más que funcionalidad implementada sobre un dato primitivo, la cual genera un nuevo dato derivado del primero. De esta forma, no es necesario el uso de un sensor físico para la generación de la información contextual, sino que se basa en determinados datos procedentes de un origen lógico. Sin embargo, en el caso de los sensores físicos, es inevitable el acceso a los sensores instalados en el propio dispositivo para la obtención de la información.

El proceso de obtención de la información desde determinados sensores suele ser una tarea que necesita de un cierto coste computacional. Un ejemplo de ellos es la lectura constante cada cierto tiempo de los datos procedentes del sensor de acelerometría del dispositivo. Si el desarrollador tan sólo realiza una aplicación que haga uso de esta funcionalidad, tendría sentido incluirla dentro de la aplicación pero, sin embargo, si es una función usada con relativa frecuencia entre las diferentes aplicaciones, ¿por qué no abstraerla y hacerla común para todas ellas?

El principio de diseño del middleware debe basarse en la reutilización de recursos físicos y lógicos, así como en la facilidad de uso tanto para el usuario desarrollador de aplicaciones como para aquel encargado de obtener información desde los sensores. Por este motivo, los sensores es una parte especialmente importante en el desarrollo de aplicaciones contextuales y ubicuas en general.

FUNCIONES DE ADAPTACIÓN DE DATOS SENSORIALES (SAF)

Generalmente los datos deben ser procesados previamente para que la aplicación pueda proceder con el análisis, la clasificación y el uso de dichos datos. Sin embargo, aún antes del proceso de generación de información *útil* a partir de la información RAW procedente de los sensores, debemos comprender el *lenguaje* que habla el sensor, con el fin de adaptar el procesamiento de la información a los datos procedentes del mismo.

En este sentido es importante determinar los valores que pueden tomar los diferentes datos pueden ser obtenidos a partir de los sensores con el fin de que todas las aplicaciones puedan compartir dichos datos independientemente del proveedor que suministre el contexto. De esta forma es posible aislar el proceso de obtención de datos del proceso de tratamiento y generación de información a partir del mismo.

Aislando el proceso de adaptación se aumenta las posibilidades de reutilización entre aplicaciones. Además, se aísla de una manera más eficaz los errores producidos por un incorrecto funcionamiento de los sensores o de la codificación de la información.

FUNCIONES DE PROCESAMIENTO DE LA INFORMACIÓN (IPF)

La información procedente de los sensores u otras aplicaciones del sistema no es suficiente generalmente para dar funcionalidad a la aplicación a desarrollar, sino que dichos datos deben ser procesados con el fin de generar nueva información. Es precisamente en este punto en el que las diferentes aplicaciones centran sus esfuerzos y costes del sistema.

Puesto que cada aplicación debe generar la funcionalidad completa por sí misma, el resultado de la funcionalidad de procesamiento tan sólo va a ser empleado por esta aplicación, sin que el resto de aplicaciones del sistema sea consciente del recurso obtenido. Pudiendo darse el caso, lo cual es muy usual en un sistema de ejecución real, que varias aplicaciones se encuentren generando el mismo recurso de manera paralela. Esto hace que todos los costes derivados de la obtención de la información se dupliquen.

De esta forma, las aplicaciones a este nivel necesitan generar información, a partir de la cual la aplicación actuará de una u otra manera, dependiendo de los valores concretos de los datos obtenidos.

FUNCIONES DE ALMACENAMIENTO DE LA INFORMACIÓN (ISF)

En general, las aplicaciones necesitan de un sistema de almacenamiento permanente o temporal de la información generada a través de capas inferiores de la aplicación. Es decir, necesitan almacenar la información procesada con el fin de poder ser accesibles más tarde. Dicha funcionalidad suele brindar métodos de acceso, modificación y eliminación de la información almacenada.

Por otro lado, las aplicaciones deben definir una política de seguridad adecuada para que otras aplicaciones no deseadas hagan uso de la información almacenada. Esto tiene especial interés en las aplicaciones contextuales, ya que generalmente la información almacenada en este tipo de aplicaciones posee un elevado grado de privacidad. Por este motivo es imprescindible evitar que otras aplicaciones o sistemas accedan de manera incontrolada a dicha información.

FUNCIONES DE SEGURIDAD DEL SISTEMA (SSLF)

Las distintas funcionalidades de seguridad presentes en las aplicaciones contextuales brindan a la propia aplicación un sistema eficaz para evitar el acceso a la información por parte de aplicaciones sin permisos para ello. Hasta el momento, el almacenamiento de la información es responsabilidad de la aplicación, así que para el usuario es totalmente indiferente el método de seguridad o las características de confidencialidad de los datos almacenados.

En este sentido, sería conveniente centralizar todas las funcionalidades de seguridad, de manera que una sola aplicación sea la encargada de velar por la correcta distribución de los datos entre las restantes aplicaciones. Lejos de esto, las aplicaciones presentes actualmente en el mercado de dispositivos móviles son totalmente independientes, lo que hace que la seguridad no pueda ser controlada de manera general por el responsable del dispositivo.

Gracias a la centralización de la seguridad es posible aislar e independizar a la aplicación de los datos almacenados, pudiendo acceder a ellos o almacenarlos mediante las funcionalidades que una tercera aplicación ofrece.

FUNCIONES DE DISTRIBUCIÓN DE LA INFORMACIÓN (SIF)

Como se ha comentado anteriormente, el acceso a la información almacenada en la base de datos de manera persistente o temporal debe estar limitado a aplicaciones con suficientes privilegios para ello. Además, la información almacenada debe estar marcada como accesible para unos determinados perfiles de aplicación para dotar de una mayor seguridad al sistema. Sin embargo, esto generalmente no se realiza en las aplicaciones reales debido al coste del desarrollo derivado de la implementación de esta funcionalidad, por lo que se suelen producir determinados leaks de seguridad que podrían comprometer la estabilidad y sobre todo, la confidencialidad de la información manejada por el sistema.

En este sentido, sería realmente interesante delegar todo el sistema de políticas de distribución en una tercera aplicación, la cual se encargaría de determinar si es posible o no la distribución de cierta información. Además, la distribución de información entre aplicaciones debe ser controlada, de forma que se debe permitir o rechazar la distribución en función de los perfiles determinados de cada aplicación.

En este nivel, las aplicaciones realizan una cierta funcionalidad en base a la información adquirida a través del procesamiento de los datos obtenidos desde los sensores. Generalmente es complejo reaprovechar la funcionalidad realizada por una aplicación, por lo que podríamos decir que llegados a este punto, las aplicaciones son realmente distintas.

De esta forma, en el middleware desarrollado se debe tener en cuenta que toda aplicación debe realizar una determinada funcionalidad a partir de los datos, los cuales serán obtenidos aprovechando al máximo los recursos del dispositivo desde el cual se obtienen. Sin embargo, todas las aplicaciones tendrán su propio entorno de ejecución totalmente independizado del resto de las aplicaciones.

MÓDULOS DEL MIDDLEWARE

A continuación serán descritas las características generales del middleware para el desarrollo de aplicaciones móviles basadas en información contextual. La arquitectura constará de tres grandes capas que permitirán a las aplicaciones acceder a la información contextual generada por un conjunto de sistemas a los que se les denomina *Context Providers*. Los *Context Providers* serán los encargados de obtener los datos generados por los sensores y procesarlos con el objetivo de conseguir información útil para las aplicaciones, por lo que se podría decir que es un sistema que obtiene información a partir de datos RAW. De esta forma, un *Context Provider* no realizará ninguna acción de cara al usuario, sino simplemente se limitará a proveer de información basada en orígenes sensoriales al resto del sistema que necesita hacer uso de esta información.

Los *Context Providers* deben comunicarse con un macro-módulo superior, al que se denomina *Core*. Es la estructura más importante del middleware en cuanto a funcionalidad, dado que este elemento será el encargado de obtener la información generada en la capa inferior, almacenarla y distribuirla entre el resto de aplicaciones que la necesiten, ya sea de manera síncrona o asíncrona. Será precisamente el *Core* de la arquitectura el que se describirá en profundidad a lo largo de este capítulo, ya que su funcionalidad es esencial para el desarrollo adecuado de aplicaciones basadas en contexto. El *Core* deberá conocer las aplicaciones que necesitan una cierta información, qué información concreta desean recibir y a qué *Context Provider* debe recurrir el *Core* para obtener la información solicitada por la aplicación.

Por último, en la capa superior de la arquitectura se encuentran las *Applications*. Una A*pplication* hará uso de la información proporcionada por el *Core*, de manera que será posible abstraer totalmente la aplicación del *Context Provider* concreto que proporciona la información. La gran ventaja de este método de comunicación es que se libera a las aplicaciones de las labores de inspección de la información, es decir, no es necesario que una aplicación controle directamente los valores del contexto, sino que pueden delegar esta funcionalidad en el *Core* gracias a la utilidad comentada anteriormente. Para realizar una acción disparada por contexto, la aplicación tan sólo tendrá que informar al *Core* de las condiciones de disparo, de manera que éste último será el encargado de comprobar dichas condiciones y gestionar las llamadas a los puntos de entradas correspondientes de cada aplicación.

En la Figura 19 se puede observar un diagrama que representa la arquitectura general propuesta para el desarrollo de aplicaciones basadas en contexto.

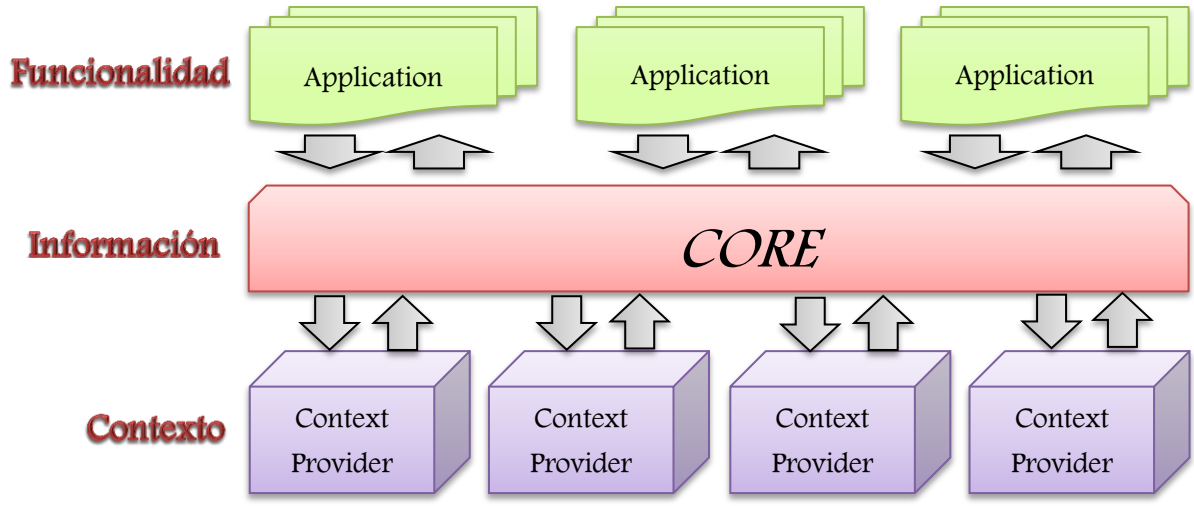


Figura 19: Arquitectura general para el desarrollo de aplicaciones

Como se ha comentado anteriormente, nuestro middleware será implementado sobre la plataforma OSGi. De esta forma, cada uno de los módulos citados se corresponderá con un *Bundle* de OSGi, de forma que la comunicación que se llevará a cabo entre ellos será a nivel de interfaz. Por este motivo, debemos establecer de manera concreta las diferentes interfaces que poseerán los módulos empleados en el middleware, es decir, el *Core*, los *Context Providers* y las *Applications*. Dichas interfaces deben dar soporte a todas las funcionalidades que tanto el núcleo del middleware como los contextos y aplicaciones puedan ofrecer. De esta forma, la única manera de comunicar los distintos módulos del sistema es a través de llamadas a puntos de entradas definidos en sus interfaces.

Mediante el uso de la plataforma OSGi, aseguramos que se está tomando una solución eficaz y robusta para el intercambio de información y la interconexión entre los distintos módulos de la aplicación. De esta forma, gracias al *Service Provider*, un módulo del middleware podrá ofrecer funcionalidad al resto, mientras que por medio del *Service Requester* se permitirá a los diferentes módulos que forman el middleware desarrollado, acceder al *Core* con el fin de generar o consultar datos del contexto del usuario.

El comportamiento de las tres capas así como los distintos módulos que permitirán emplear el middleware debe estar establecido. Por ello, una vez detalladas cada una de las capas se procederá a describir la interconexión entre ellas para permitir la consulta y la actualización de información contextual, así como establecer las políticas de seguridad necesarias para la subscripción tanto de contextos como de aplicaciones.

PROVEEDOR DE CONTEXTO

Se define un proveedor de contexto como todo aquel agente capaz de aportar información relevante y completa sobre el usuario objeto de observación. Para ello, el proveedor de contexto puede utilizar sensores físicos, lógicos o mixtos indiferentemente, combinándolos para obtener un nuevo valor empleando como única referencia la información obtenida desde los sensores o desde otros elementos externos al middleware sobre el que se ejecuta.

Un ejemplo de proveedor de contexto puede ser la velocidad de desplazamiento, la posición o la altitud a la que se encuentra un usuario, los cuales pueden ser obtenidos a través del sensor GPS. Además de contextos simples, como el del ejemplo anterior, pueden generarse información contextual

más compleja, como por ejemplo la actividad física realizada por el usuario, en la cual intervienen sensores de aceleración y un procesamiento de aprendizaje y reconocimiento.

Todos los proveedores, a partir de los cuales se obtiene información del entorno del usuario, se encuentran englobados en la primera capa de la arquitectura. Dicha capa será la encargada de obtener toda la información que, posteriormente, las aplicaciones emplearán para llevar a cabo la funcionalidad para la que estén diseñadas.

Con el fin de establecer un diseño común para todas las estructuras del middleware, todos y cada uno de los *Context Providers* instalados en el entorno de ejecución deben estar caracterizados mediante una interfaz a la que denominaremos *IContextProvider*. De esta manera se asegura que la aplicación instalada se compone de toda la funcionalidad necesaria para formar parte del conjunto de proveedores de contexto de la aplicación. Esto se ha tenido en cuenta a la hora de definir la propia interfaz, de manera que un proveedor de contexto implementará toda la funcionalidad necesaria para generar valores a partir de los sensores y entregar dichos valores a la capa superior del middleware con el fin de que sean procesados y distribuidos entre las aplicaciones que lo necesiten.

Gracias a la implementación del módulo de *Context Provider* mediante el uso de interfaces, es posible que el *Core* busque entre todos los bundles instalados en el entorno de ejecución del middleware, aquellos que estén caracterizados mediante la interfaz *IContextProvider*, incluyéndolos así en la lista de contextos manejados por el sistema. Sin embargo, del concepto de inscripción de contexto nos encargaremos más adelante.

Una vez el *Core* ha identificado los proveedores de contexto instalados, se establecerá una conexión entre éste y el bundle correspondiente, de forma que sea posible acceder a los datos generados y de esta forma, ser redistribuidos entre las diferentes aplicaciones. En el ejemplo anterior, cuando los *Context Providers* de velocidad, posición y actividad física sean instalados en el entorno de ejecución, el *Core* detectará la presencia de éstos y establecerá sendas conexiones con el fin de obtener la información necesaria. Veremos que cuando un *Context Provider* genera una nueva información, inmediatamente éste lo comunicará al *Core*, de manera que se pueda realizar las acciones pertinentes del dato por el núcleo del middleware.

A modo de resumen podríamos decir que un *Context Provider* se limitará a inscribirse en el *Core* del middleware e informar a este cuando tenga un nuevo dato disponibles. Para estas dos acciones hará uso del *Core*, concretamente de los módulos *Context Subscription* y *Context Updater*. El primero de ellos será el encargado de suscribir un determinado contexto en el middleware, de forma que a partir de ese instante las aplicaciones que lo soliciten podrán hacer uso de la información generada por éste. En cambio, el *Context Updater* será el módulo responsable de actualizar los datos provenientes del *Context Provider* cada vez que este último le comunique que existe un nuevo dato disponible.

Diseño

Desde el punto de vista del diseño, un *Context Subscription* no maneja información directamente, sino que lo hace a través de las denominadas *Functions*. Una *Function* no es más que el valor de un dato concreto que forma parte de un contexto más genérico. De esta forma, mediante las *Functions* es posible establecer ciertas relaciones grupales entre los distintos valores contextuales, concretamente las relaciones se basarán en el contexto sobre el cual aporten información las diferentes funciones.

El concepto de función surge del hecho de que muchos datos contextuales están íntimamente relacionados, hasta el punto que en algunas ocasiones un valor se obtiene mediante la aplicación de alguna expresión sobre otro valor o, por ejemplo que para el cálculo de dos elementos se precisa del mismo sensor. De este modo, con el fin de reutilizar al máximo los recursos del dispositivo, se establece una cierta jerarquía en cuanto al desarrollo de contextos, de manera que un determinado *Context Provider* deberá tener al menos un *Function* asociado. Por tanto, está claro que el elemento realmente encargado de entregar el valor asociado a una variable contextual será la *Function*, de manera que el *Contexto Provider* deberá tener una referencia de todas y cada una de las funciones que lo componen.

En el caso del ejemplo, tanto la velocidad como la posición son valores que podrían corresponder al mismo ámbito contextual de *Localización*. Es más, ambos valores están estrechamente relacionados dado que son obtenidos a partir del mismo sensor. Todo esto hace que el coste computacional de obtener los dos datos anteriores en *Context Providers* independientes sea el doble que hacerlo en uno único. De esta forma, el *Context Provider* de localización tendrá varias funciones entre las que se encontrarán la encargadas de obtener la posición y la velocidad del usuario.

Paralelamente a la definición del tipo *Function* empleado para representar los diferentes valores relacionados con un contexto determinado a través del *Context Provider*, se establece el concepto de *Sensor*. Los diferentes *Context Provider* deben generar la información necesaria a partir de un conjunto de sensores (físicos y/o virtuales) que le proporcionan los datos básicos para generar el nuevo contexto. Es interesante determinar en todo momento los diferentes sensores de los que hace uso un determinado *Context Provider*, ya que como veremos más adelante podría ser necesario en caso de fallo de un determinado sensor, obtener todos los proveedores de contexto que se encuentren usándolo con el fin de enviar una señal de interrupción o de mal funcionamiento. Los *Sensors* empleados por el middleware deben seleccionarse de entre todos los sensores definidos en la clasificación *SensorTypes* que será definida más adelante.

El uso de clasificaciones en el middleware desarrollado es obligado, ya que estamos generando una solución de integración que permitirá comunicar una serie de aplicaciones y proveedores de contextos entre sí. Dicha comunicación deberá llevarse a cabo a través de una serie de reglas bien definidas y comprensibles por todas las entidades de la plataforma, por lo que se hace necesaria la tipificación de los diferentes valores tratados en el middleware. En el caso concreto de la clasificación *SensorTypes* se pretende definir un conjunto de sensores completo, agrupado en varias subcategorías, que integre todos los tipos de sensores empleados a día de hoy para el desarrollo de aplicaciones ubicuas.

NÚCLEO - CORE

El *Core* es la capa central del middleware, alrededor de la cual se distribuyen tanto los clientes (*Applications*) como los servidores de contexto (*Context Provider*), además de ser la única común a todos los dispositivos que empleen el middleware. En ella se engloba toda la lógica que permite comunicar a las aplicaciones con los proveedores de contexto con el fin de dotar de la información necesaria a ambas entidades.

De esta forma, las diferentes aplicaciones que usen el middleware reducirán su complejidad, pasando de detectar y procesar la información contextual a simplemente, reaccionar en consecuencia en base a la información provista por los diferentes *ContextProviders*. Dichos *ContextProviders* han

debido ser instalados en el sistema previamente. Por lo tanto, el *Core* deberá tener en cuenta las siguientes premisas:

- En la capa inferior existe un conjunto de proveedores de contexto que facilitan información continuamente o a petición de la capa superior, por lo que el *Core* deberá mantener el control sobre dichas entidades.
- En la capa superior existen múltiples aplicaciones suscritas a determinados contextos con el fin de obtener información procedente de ellos. Por lo tanto, el *Core* debe distinguir entre las diferentes aplicaciones inscritas, puesto que al producirse la actualización de un determinado valor de contexto, deberá alertarse tan sólo a aquellas aplicaciones que estén empleando dicho valor, aislando al resto de aplicaciones de las actualizaciones para las que no estén inscritas.
- Una determinada aplicación podrá combinar varios contextos para definir uno más complejo. Gracias a esto, se reutiliza el código de los detectores de contexto, se simplifica el funcionamiento y se minimiza el uso de memoria del dispositivo, lo cual es de gran importancia en el caso de los dispositivos móviles. Gracias al módulo de agregación de contextos del *Core*, será posible establecer unas determinadas condiciones para el lanzamiento de eventos, de manera que la aplicación se aislará totalmente de la evaluación de tal condición, siendo esta una tarea del *Core*.
- El núcleo debe guardar un historial de los valores generados por los diferentes proveedores de contextos a través de sus funciones. Esto hará posible que las aplicaciones puedan consultar ciertos valores a través de una serie de restricciones, tales como la precisión, la marca temporal o el propio valor del contexto.
- En el núcleo se deberán controlar en todo momento los accesos indebidos o restringidos a la información contextual almacenada, por lo que será necesario un proceso de suscripción al servicio tanto por parte de las aplicaciones como por los proveedores de contexto.

En base a las características descritas, el *Core* poseerá dos puntos de acceso, uno para la capa inferior y otro para la superior que serán respectivamente el módulo de provisión de contextos y el de aplicaciones. A través del primero, los proveedores de contexto pueden acceder al *Core* para publicar las actualizaciones producidas sobre los datos contextuales, mientras que a través del segundo, se procederá a la entrega de la información solicitada por parte de las aplicaciones. En ambos casos, el registro de componentes se hará empleando el framework de gestión de módulos provisto por *OSGi*, el cual se encargará de gestionar en todo momento de forma dinámica los *ContextProviders* y las *Applications* presentes en la arquitectura.

La sección de *Applications* empleará el *Core* para obtener y manejar toda la información necesaria para realizar una determinada tarea de alto nivel o que, en general, sea necesaria la interacción del usuario para poderse llevar a cabo. Esto podría resumirse en los siguientes criterios de usabilidad del *Core*:

- *Consultar un contexto actual*. Una *Application* podrá acceder en todo momento al último dato obtenido a partir de un *ContextProvider* siempre y cuando dicha aplicación se encuentre suscrita al middleware
- *Consultar un valor contextual a través del histórico*. El *Core* mantendrá en todo momento un histórico de los últimos valores obtenidos para cada uno de los contextos.

- *Suscribir una aplicación a un contexto*. En lugar de realizar consultas sobre el repositorio de datos gestionado por el *Core*, es posible la suscripción a un determinado contexto, de manera que la aplicación recibirá el dato obtenido siempre que se genere uno nuevo. Esto abstrae a la aplicación de las consultas, gracias a la comunicación mediante eventos que el *Core* iniciará a la llevada de nueva información conceptual.
- *Modificar los eventos de activación del contexto*. El módulo de *Application* no sólo podrá inscribirse a los diferentes eventos ofrecidos, sino también auto-eliminarse de la red de difusión del contexto o de modificar las condiciones de alerta.
- *Crear contextos agregados*. En la mayoría de las ocasiones, las aplicaciones no sólo necesitan información relacionada con un contexto, sino que sus funcionalidades se llevan a cabo cuando se produce una determinada condición que afecta a más de una información contextual. Esto se podrá realizar a través de la *agregación de contexto*, lo cual también se denomina como *contexto derivado*.
- *Consultar los proveedores de contexto subscritos a la plataforma*. Previo al intercambio de información contextual entre el *Core* y las *Applications* es necesario ofrecer la funcionalidad de búsqueda de los diferentes contextos y funciones para que la aplicación puede suscribirse al aquel que mejor se adapte a las necesidades.

A grandes rasgos, el proceso necesario para dotar a las aplicaciones de información contextual se describe a continuación. En primer lugar, todos los *ContextProvider* deberá subscribirse a la plataforma. Este proceso se realiza de una manera extremadamente sencilla gracias al framework de *OSGi* empleado para la definición del middleware, ya que se puede realizar en cualquier momento del ciclo de vida de la aplicación, e incluso aunque el middleware no se encuentre en funcionamiento. Para ello bastará con copiar el módulo del *ContextProvider* en el directorio empleado para este fin, siendo automático el reconocimiento por parte del middleware.

Posteriormente, el *Core* iniciará un proceso de comunicación con cada uno de los *ContextProviders* instalados, con el fin de comprobar los permisos de los mismos y completar la inscripción en la plataforma. Tras ello, la aplicación será registrada en el *Core*, pudiendo de este modo almacenar la información relativa a la propia aplicación. Por otro lado, la aplicación se suscribirá a uno o varios eventos, proceso mediante el cual el *Core* podrá alertar a la aplicación cuando se genere nueva información o cuando se satisfaga alguna de las restricciones introducidas en el módulo de *Context Aggregator*.

A partir de este momento, los proveedores de contexto comenzarán a enviar la información al *Core*, de manera que este la almacenará y comprobará si se cumple alguna de las condiciones de disparo de evento configurado por las aplicaciones anteriormente. En caso afirmativo, el *Core* comprueba las diferentes aplicaciones relacionadas con la activación del evento o condición satisfecha y se alerta mediante la ejecución de un evento.

A continuación, se presenta un ejemplo de funcionamiento basado en una aplicación cuya funcionalidad consiste en la realización de una llamada de emergencia a un número previamente establecido cuando el usuario sufra una caída. Este supuesto está basado en una aplicación real y extremadamente útil en el campo del seguimiento de personas ancianas.

En primer lugar, el usuario deberá tener instalado en el dispositivo el *ContextProvider* correspondiente el cual, en caso de no estar instalado, podría descargarse desde la plataforma Web que actualmente se está desarrollando para proporcionar a los usuarios de los diferentes *ContextProviders* realizados por la comunidad de desarrollo. Posteriormente, el framework de *OSGi* reconocería el

proveedor de contexto informando al *Core* de dicha instalación, por lo que este último inscribirá, si es posible, el proveedor de contexto en el *middleware* desarrollado. Después, la aplicación se inscribirá en la función encargada de reconocer la actividad llevada a cabo por el usuario dentro del contexto registrado. Concretamente, se generará un agregado que se evaluará positivamente cuando el valor de la función anterior se corresponda con el valor asociado a la caída. A partir de ese momento, el proveedor de contexto comenzará a enviar la información recogida en cada momento (Andando, parado, etc.) y registrará cada nuevo dato en la base de datos.

En el momento en que el proveedor de contexto registre en la aplicación una información asociada a caída, el *Core*, tras registrar el evento en la base de datos, lanzará un evento hacia la capa superior, es decir a la capa *Application*, concretamente a la aplicación suscrita al evento que se ha producido. En el preciso momento que la aplicación recibe este evento, realizará la funcionalidad asociada, en este caso la llamada de emergencia.

Debido a la complejidad de este módulo, el *Core* se dividirá en cuatro secciones menores, las cuales realizarán una funcionalidad concreta dentro de la arquitectura. Dicha funcionalidad será descrita a continuación de una forma general, siendo más adelante cuando se detallará los diferentes módulos de los que se compone cada (ver Figura 20). Las secciones son las siguientes:

- *Context Section.* En esta sección se engloba toda la funcionalidad encargada de inscribir las aplicaciones y los proveedores de contexto al *Core*. Además, también proveerá a los *ContextProvider* de un punto de acceso a la capa centra para poder actualizar la información generada.
- *Data Section.* En ella se almacena toda la información contextual procedente de los *ContextProviders*. Parte de esta información quedará almacenada con el fin de generar un historial sobre cada uno de los valores contextuales, gracias a lo cual las aplicaciones podrán consultar no sólo el último valor de contexto, sino también los anteriores. En el *Data Section* las aplicaciones también dispondrán de un punto de acceso al *Core*, el cual les permitirá realizar consultas sobre la base de datos de información contextual.
- *Event Section*. Es la encargada de registrar *Applications* cuando éstas desean suscribirse a un determinado evento. Además de almacenar la inscripción de la aplicación al evento concreto, el *Event Section* será encargado de alertar a la aplicación mediante un evento que se ha satisfecho la condición que la aplicación registró anteriormente.
- *Aggregation Section*. Debido a la complejidad de los métodos de agregación de información contextual, se ha decidido aislar toda la funcionalidad relativa a esta sección. En ella se combinarán una serie de eventos simples, valores asociados a dichos eventos así como funciones lógicas que los combinan. De este modo, una aplicación puede establecer una combinación de eventos simples a partir de la cual se generará un evento cuando se satisfaga.

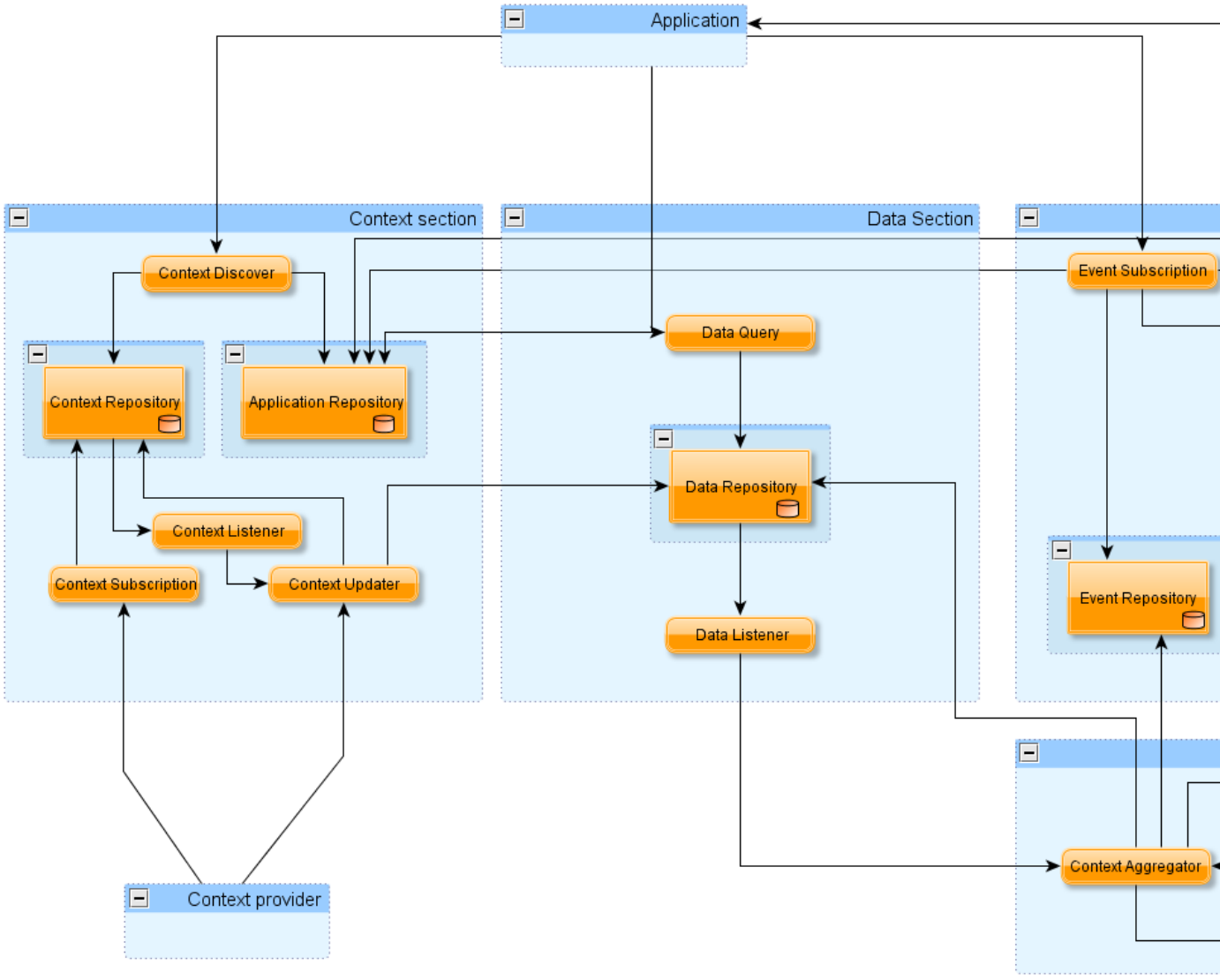


Figura 20: Estructura general del Core para el middleware propuesto

Dividiremos el diseño de la arquitectura en las diferentes secciones que se propusieron anteriormente. Por tanto, comenzaremos estudiando las interfaces y las funcionalidades de los módulos de *Context Section*.

Context Section

Este módulo puede ser dividido en dos grandes subgrupos en base a la funcionalidad que poseen y a los módulos con los que interaccionan. En primer lugar encontraríamos el subgrupo encargado de establecer la comunicación con los *ContextProviders*, el cual se compone por los módulos *Context Subscription, Context Updater, Context Listener* y *Context Repository*. Por otro lado se observan los módulos encargados de establecer la comunicación y gestionar las diferentes *Applications* residentes en el sistema, los cuales son *Context Discover* y *Application Repository*. Los distintos módulos de esta sección así como su interconexión con otros módulos pueden verse en la Figura 21.

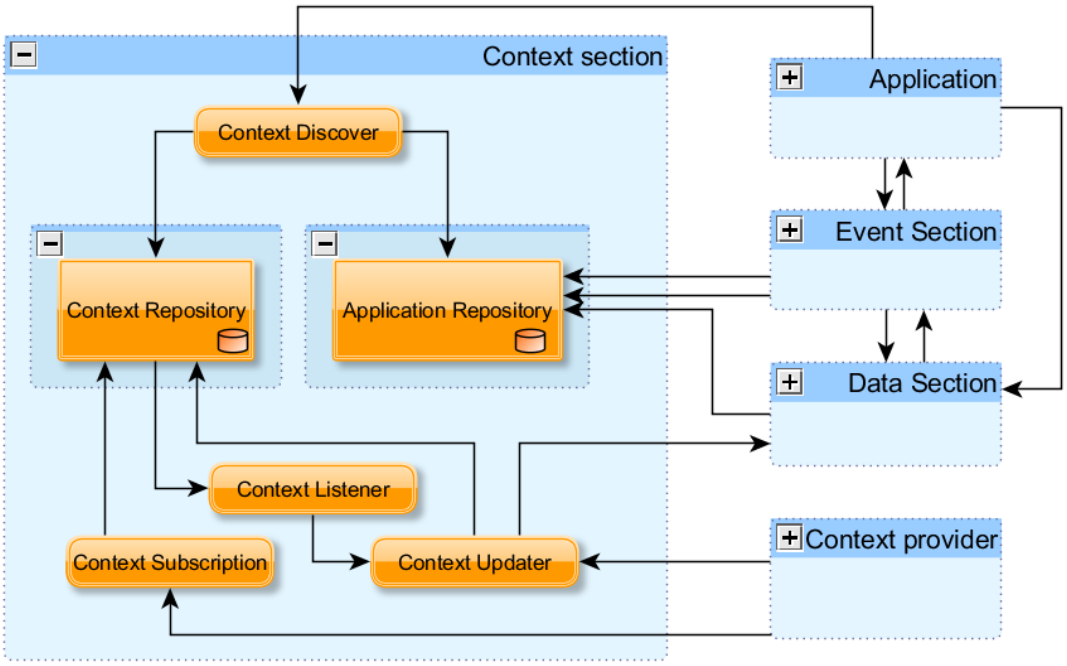


Figura 21: Representación de los módulos de Context Section y sus conexiones con el resto de secciones de la arquitectura

Comenzaremos estudiando el *Context Subscription*. La principal función de este módulo es brindar a los *ContextProviders* un punto de entrada al *Core* para que puedan registrarse de manera dinámica en la arquitectura. Además, este módulo será el encargado de iniciar la comunicación con un determinado *Bundle* con el fin de identificar si se trata de un *ContextProvider* y, si es así, solicitar la petición de inscripción en el *Core*. Dicha inscripción se formalizará cuando el *Context Subscription* inscribe al *ContextProvider* en la base de datos de repositorios, denominada *Context Repository*.

Por otro lado, el *Context Subscription* debe identificar aquellos *ContextProviders* que se encuentran activos, lo cual se hará mediante el envío de señales de petición de estado, a lo que el proveedor de contexto deberá responder con el estado en el que se encuentra. Si tras un número determinado de intentos de comunicación con el proveedor de contexto el *Context Subscription* no recibe respuesta, dicho *ContextProvider* será eliminado de la plataforma, por lo que no se permitirá el acceso a la información contextual que debería proporcionar.

Relacionado con el *Context Subscription* se encuentra el módulo de almacenamiento denominado *Context Repository*, encargado de registrar todos los *ContextProviders* inscritos en la arquitectura, así como llevar a cabo distintos tipos de búsqueda sobre los contextos almacenados. Gracias a este módulo es posible gestionar las diferentes políticas de suscripción de aplicaciones a diferentes fuentes de información procedentes de los proveedores de contexto.

Debido a que el *middleware* debe ser integrado en un dispositivo móvil, la memoria es un bien que debe ser administrado de manera eficiente. Por este motivo, los datos se almacenarán en formato *XML*, el cual no sólo permite llevar a cabo de manera eficiente el almacenamiento de información, sino que además ofrece la posibilidad de realizar consultas tipificadas sobre los datos almacenados. Esto último es de especial interés a la hora de realizar búsquedas por parte de las aplicaciones sobre funciones o información concreta provista por los *ContextProviders* instalados.

Para facilitar las labores de búsqueda y manejo de resultados de las mismas a las entidades de la capa de *Applications*, el módulo *Context Repository* también posee la capacidad de devolver las coincidencias de las consultas de búsquedas de *ContextProviders* como objetos, los cuales vienen caracterizados por la interfaz *IContextProviderInformation*.

Con el fin registrar los cambios producidos en el sistema de almacenamiento de información integrado en el módulo *Context Repository*, aparece el módulo *Context Listener*. Este módulo está orientado en gran medida a futuras ampliaciones que se produzcan en el diseño del middleware, ya que gracias a él los diferentes módulos podrán recibir información sobre los cambios que se produzcan en el conjunto de proveedores de contextos. Concretamente, la funcionalidad en la arquitectura actual del middleware es la de permitir al C*ontext Updater* registrar los cambios producidos en el conjunto de *ContextProviders* suscritos al sistema. El hecho de que el módulo *Context Updater* deba recibir esta información se debe a que éste módulo, almacenará internamente un registro de los proveedores de contextos activos en el sistema, con el fin de minimizar el tiempo de consulta de los *ContextProviders* disponibles.

Por otro lado, los *ContextProviders* podrán actualizar la información asociada a los contextos con los cuales trabajan. Esta información será actualizable a través del *Context Updater*, módulo que brinda los puntos de acceso necesarios para poder registrar las funciones concretas de cada uno de los contextos sobre las que se han producido los cambios en la información. El *Context Updater* no sólo deberá registrar los cambios de información de los valores contextuales, sino comprobar previamente que el *ContextProvider* que ejecuta la actualización tiene un origen seguro. Esto se hará comprobando que el proveedor ha sido inscrito anteriormente en la lista de proveedores de contextos de la plataforma. Para este propósito, como se comentó anteriormente, se mantiene una copia interna de los *ContextProviders* inscritos, gracias a la cual no es necesario recurrir al módulo *Context Repository* evitando así tiempo de procesamiento en las actualizaciones.

Desde el punto de vista de las *Applications*, el módulo principal de *Context Section* es el *Context Discover*. Este módulo brinda la posibilidad de realizar consultas acerca de la diferente información ofrecida por los *ContextProviders*. Esta funcionalidad es decisiva a la hora de hacer un uso eficiente del middleware, ya que el hecho de realizar una búsqueda adecuada de la información necesaria por las aplicaciones, determinará la efectividad de la arquitectura desde el punto de vista de la eficiencia. Una búsqueda excesivamente complicada daría lugar a que los desarrolladores realizaran aplicaciones dependientes totalmente de los *ContextProviders* realizados por los propios desarrolladores. Sin embargo, gracias a la definición de las interfaces necesarias así como de las clasificaciones

contextuales que serán recogidas más adelante, se estandariza la búsqueda de proveedores de contextos en base a la información necesaria por las aplicaciones.

Previo a la consulta de proveedores, las *Applications* deben registrarse en la plataforma para poder beneficiarse de las ventajas de la información manejada por ésta. El registro de *Applications* también será responsabilidad del *Context Discover*. En este caso, el *Context Discover* deberá monitorizar los cambios producidos en la inscripción de elementos del framework de *OSGi*, con el fin de detectar aquellos que representen a elementos *Applications* del middleware desarrollado.

Las *Applications* inscritas en el middleware serán almacenadas en el módulo *Applications Repository*, el cual posee una estructura semejante a la expuesta anteriormente en el *Context Repository*, es decir, un sistema de almacenamiento basado en *XML* y otro de consultas basado en interfaces. En general, el módulo de *Application Repository* será empleado para comprobar el origen adecuado de las aplicaciones así como para comprobar los identificadores de una *Application* dentro del middleware.

Data Section

La sección de datos del middleware posee tres módulos, los cuales serán los encargados de almacenar toda la información relacionada con orígenes contextuales proporcionada por los *ContextProviders* a través del módulo *Context Updater*. Dichos módulos pueden verse representados en la Figura 22. El módulo principal de esta sección, en torno al cual se distribuye las consultas de información es el *Data Repository*. El propósito de este repositorio es almacenar toda la información de contexto, teniendo la capacidad de recopilar un histórico de datos contextuales con el fin de posibilitar la realización de determinadas consultas.

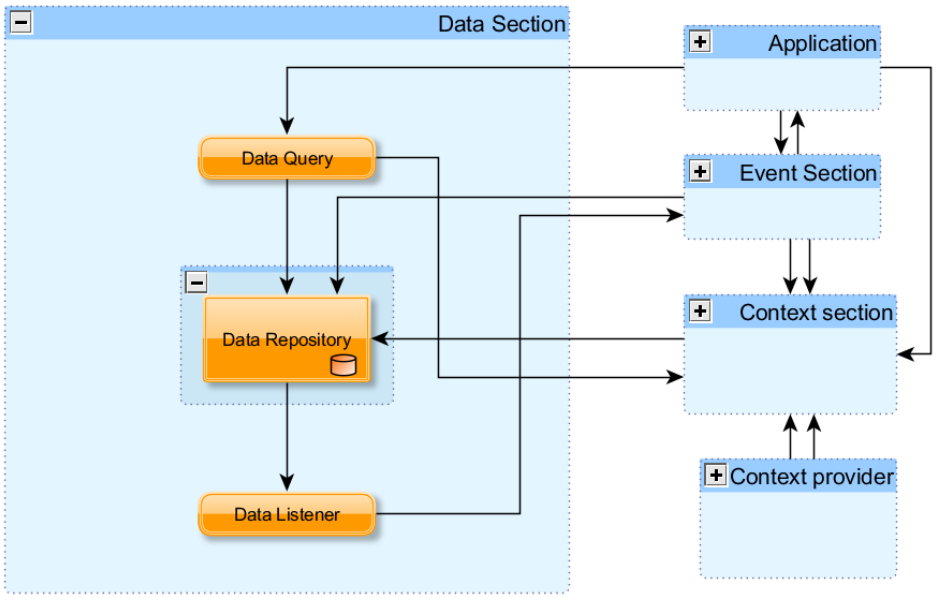


Figura 22: Representación de los módulos de Data Section y sus conexiones con el resto de secciones de la arquitectura

Una posible alternativa al almacenamiento de información contextual sería una gestión distribuida, es decir, cada aplicación se encargaría de almacenar su propia información relacionada con el contexto. El primer problema de esta solución sería que la gestión de datos (modificación, borrado o adición) sería más compleja desde el punto de vista computacional, ya que sería necesario actualizar

estos valores en los repositorios de todas las aplicaciones en los cuales se usa la información concreta. En segundo lugar, se produce un desperdicio considerable del espacio en memoria, ya cada aplicación deberá mantener una copia de la información manejada. Este problema se encuentra con mucha frecuencia en contextos como el posicionamiento, ya que muchas aplicaciones hacen uso de este tipo de datos de manera simultánea.

En contraste con la alternativa anterior, se ha optado por un almacenamiento y gestión de la información contextual centralizado, de manera que todas las aplicaciones accederán al mismo módulo con el fin de obtener dicha información. Esto facilita las labores de mantenimiento, gestión y consulta de los datos proporcionados por los proveedores de contexto. Otro beneficio de esta opción es la capacidad de permitir un almacenamiento de históricos de información, pudiendo ser éstos consultados por las diferentes *Applications* inscritos en el sistema. Toda la funcionalidad anterior será gestionada a través del módulo *Data Repository*, el cual mantendrá también una referencia a los últimos valores de cada función de contexto en una estructura biyectiva que relacionará el identificador de la función y del contexto con el último dato recibido del proveedor de contexto. Esta forma de almacenar la información permitirá al sistema de evaluación de agregados acceder a la información de una manera eficiente y menos compleja computacionalmente debido a la reducción y la indexación de los datos.

Paralelamente a este módulo, el *Data Listener* proveerá al resto de módulos del middleware un método a través del cual recibir información acerca de los cambios producidos sobre el repositorio, aislando a este del resto de módulos del sistema. Esto será indispensable a la hora de evaluar las condiciones del lanzamiento de eventos, las cuales serán expuestas más adelante. Mediante este método de actualización, el módulo que desee obtener información sobre las actualizaciones de los datos contextuales, tan sólo deberá registrarse a la lista de difusión de eventos provista por el *Data Listener*, y será este último el encargado de enviar una notificación con el nuevo dato provisto así como con toda la información relativa al *ContextProvider* que brinda la información contextual.

De igual forma que ocurría con el *Context Listener*, el módulo *Data Listener* podía haber sido incluido en el propio *Data Repository*. Sin embargo, por motivos de cohesión y reducción del acoplamiento, así como por la previsión de futuras actualizaciones de la arquitectura, se ha determinado que es una decisión de diseño justificada el hecho de independizarlo del módulo de almacenamiento de datos.

Por último, el módulo *Data Query* del *Data Section* es el encargado de actuar de punto de acceso frente a las consultas síncronas llevadas a cabo por parte de las *Applications* sobre la información contextual almacenada en el *Core*. Ya se comentó anteriormente que las *Applications* pueden obtener la información contextual mediante dos métodos, en primer lugar mediante la suscripción a un evento, el cual será estudiado posteriormente y en segundo lugar, mediante un acceso síncrono a la información mediante el envío de una consulta que se ejecutará contra el *Data Repository*. En este caso estudiaremos el segundo método propuesto.

Para asegurar que las consultas se realizan de manera controlada, el módulo *Data Query* deberá comprobar el origen de la aplicación. Esta comprobación se hará a través del *Application Repository*, el cual comprobará que la aplicación que desea hacer la consulta se suscribió anteriormente a la plataforma, cumpliendo así los criterios de seguridad establecidos. Si por el contrario, la *Application* no se suscribió a la plataforma, el módulo *Data Query* no efectuará la búsqueda de la información requerida, evitando de este modo la consulta descontrolada de datos por parte de aplicaciones no seguras o, en general, de cualquier aplicación que no haya realizado la inscripción en la plataforma.

El proceso de consulta por parte del *Data Query* se realizará aplicando una búsqueda sobre los datos *XML* almacenados en el *Data Repository*. Las condiciones concretas de las búsquedas vendrán determinadas por la propia aplicación que las realiza, siendo estas condiciones las descritas por la interfaz *IDataQuery*, la cual se expondrá más adelante. El resultado de estas consultas será en general, un conjunto de elementos de tipo *IDataContext*, cada uno de los cuales contendrá información relativa a cada uno de los valores contextuales filtrados.

Event Section

En esta sección se llevará a cabo toda la lógica necesaria para realizar la suscripción a los eventos del sistema por parte de las *Applications*, así como la comunicación de eventos por parte del *Core* cuando se cumplan las condiciones establecidas para sus lanzamientos.

A diferencia del I*DataQuery* a lo largo del *Event Section* se trabajará tan sólo con eventos asíncronos, es decir, las aplicaciones deberán suscribirse a un determinado evento que puede ser configurado según sus necesidades concretas, y posteriormente será el *Core* el encargado de comunicar cuando se evalúan satisfactoriamente las condiciones del evento.

Comenzaremos la descripción de esta sección por el módulo *Event Subscription*, cuyos objetivos serán ofrecer a las aplicaciones un punto de entrada al *Core* que permita registrar eventos en el sistema, de manera que cualquier aplicación podrá seleccionar las condiciones concretas, con el fin de que el *Core* lanzará el evento dirigido a dicha aplicación cuando se evalúen a cierto las condiciones seleccionadas. Por otro lado, el *Event Subscription* deberá aplicar las políticas de seguridad necesarias a la suscripción de eventos, del mismo modo que se aplicó anteriormente para el caso del módulo *Data Query*. Esta acción se llevará a cabo mediante la realización de una consulta sobre el módulo *Application Repository* en la que será solicitada toda la información referente al conjunto de *ContextProviders* a los que está suscrita la aplicación que realiza la suscripción. De este modo, tan sólo se permitirá el acceso al módulo de registro de eventos a aquellas *Applications* que pretendan acceder a eventos provistos por *ContextProviders* en los que se haya inscrito previamente.

Por último, el *Event Subscription* también se encargará de registrar eventos agregados al sistema, es decir, eventos complejos que se forman a partir de un conjunto de operaciones lógicas aplicadas sobre cada par de eventos de la aplicación. De esta forma, se podría interpretar como un evento en el que la evaluación dependerá de varias informaciones procedentes de uno o más *ContextProviders*.

Con el fin de identificar a la aplicación que suscribe el evento, es necesario almacenar el identificador de la aplicación así como una referencia a la misma. Esto será necesario cuando, posteriormente, sea necesario alertar a la *Application* que se ha evaluado satisfactoriamente el evento establecido, produciéndose así la llamada a un método concreto de la aplicación a modo de evento generado por el *Core*.

Siguiendo con la dinámica de módulos de almacenamientos del *Core*, es necesario definir un módulo encargado de almacenar la información relativa a los eventos a los que se suscriben las diferentes aplicaciones del sistema. En este caso, el nombre de este módulo es *Event Repository*, el cual almacenará objetos de tipo *IEvent*. El hecho de poseer un almacenamiento centralizado de los eventos del sistema, permite la reutilización de los mismos, de formas que si dos *Applications* desean suscribirse al mismo evento de información, tan sólo será necesario almacenar una vez el evento y establecer sendas relaciones para cada aplicación. De esta forma, cada objeto *IEvent* almacenado en el *Event Repository* llevará asociado un identificador que lo representa de manera unívoca en el sistema,

por lo que si ya existía el evento que se desea almacenar, el identificador con el que se trabajará será el del evento previamente almacenado. Este identificador le será devuelto al módulo *Event Subscription* cuando añada el evento al conjunto de eventos almacenados, con el fin de que sea posible manejar los eventos en el nivel superior.

El módulo *Event Subscription* es accedido también desde el módulo *Context Aggregator* que, como se comentará posteriormente, deberá evaluar un determinado evento cuando se produzca una actualización de la información contextual de los diferentes *ContextProviders* que componen el sistema.

Continuaremos la descripción del *Event Section* con el módulo *Event Provider*, cuyo objetivo es brindar a las aplicaciones un sistema de avisos que se activará cuando se evalúe positivamente los diferentes eventos a los cuales se han suscrito a través del *Context Subscription*. Este módulo deberá filtrar el sistema de eventos tan sólo a aquellas aplicaciones que se hayan suscrito a la alerta adecuada, es decir, si existen un conjunto de 10 eventos inscritos en el sistema para 8 aplicaciones distintas, la evaluación positiva de uno de los eventos tan sólo le será comunicado al conjunto de aplicaciones que se suscribieron previamente a dicho evento. Esto es indispensable para convertir la capa de *Core* de la arquitectura en un elemento lo más transparente posible para las aplicaciones, las cuales tan sólo se verán beneficiadas por la funcionalidad que este otorga.

El sistema de comunicaciones a las aplicaciones se llevará a cabo mediante el almacenamiento de una referencia a la aplicación, como se comentó anteriormente, permitiendo así la posibilidad de llamar a la función *onNewEvent* presente en la interfaz *IApplication*, la cual deben implementar todas las aplicaciones que deseen hacer uso del middleware.

Por último, concluiremos la descripción del *Event Section* con el módulo de almacenamiento *AppX*. Este módulo surge de la necesidad de almacenar una referencia de las diferentes aplicaciones del sistema para que puedan ser accesibles por parte de los módulos necesarios, entre los que se encuentra el módulo *Event Provider*. La manera de almacenar el conjunto de aplicaciones será mediante un conjunto de pares clave-valor, donde la clave será un identificador unívoco de la aplicación, el cual se calculará de manera automática en el propio *Core*, mientras que los valores serán objetos que deberán implementar la interfaz *IApplication*. Este módulo será accesible desde el *Event Subscription* y desde el *Event Provider*, ya que serán los únicos módulos que necesitarán acceso a los diferentes métodos de las *Applications*.

Posteriormente se comentará en profundidad el modo en el que se relacionan los distintos módulos de esta sección a la hora de realizar la suscripción de un evento, aunque estas relaciones se podrían resumir en los siguientes puntos:

- En primer lugar, para llevar a cabo una suscripción, una determinada aplicación deberá almacenar en el *Core* un registro que determine los diferentes *Context Provider* sobre los que desea obtener información. Para ello, será necesario realizar una solicitud al módulo *Event Subscription*.
- Posteriormente a la suscripción de la aplicación en el *Event Subscription*, el módulo *Event Repository* realizará una solicitud al módulo *Application Repository* con el fin de comprobar que la aplicación está suscrita a los *ContextProviders* asociados al evento creado. En dicha solicitud se incluye el identificador de la aplicación y el identificador de los diferentes *ContextProviders* involucrados. En caso de que la aplicación se haya suscrito previamente a todos los *ContextProviders*, se comenzará con el almacenamiento de la solicitud de

suscripción al evento. En general, una denegación de la solicitud de inscripción a un evento suele provenir por dos motivos; por un lado que no sea correcto el evento generado (por ejemplo un evento que incluya un contexto no existente) o, por otro lado, que éste no haya pasado las políticas de seguridad establecidas. De esta manera se protege al usuario final de aplicaciones que puedan resultar potencialmente peligrosas o que puedan acceder a información sensible sin la aceptación del usuario.

- Posteriormente, el módulo *Event Subscription* se encargará de insertar los registros necesarios en el sistema, es decir, inscribir a los eventos en el conjunto de eventos del módulo *Event Repository*.
- Por otro lado se deberá almacenar el agregado correspondiente (generado dentro del módulo a partir de los eventos de entrada) dentro del módulo Aggregation Repository. En este caso, el agregado será un evento sencillo, es decir, un agregado compuesto por tan sólo un evento
- En último lugar, también será necesario almacenar la relación entre la aplicación solicitante y el agregado generado, de tal manera que cuando se cumplan las restricciones del agregado sea posible llamar al evento adecuado de la *Application* que solicitó la inscripción del evento. Esto se hará a través del módulo *AppX*.

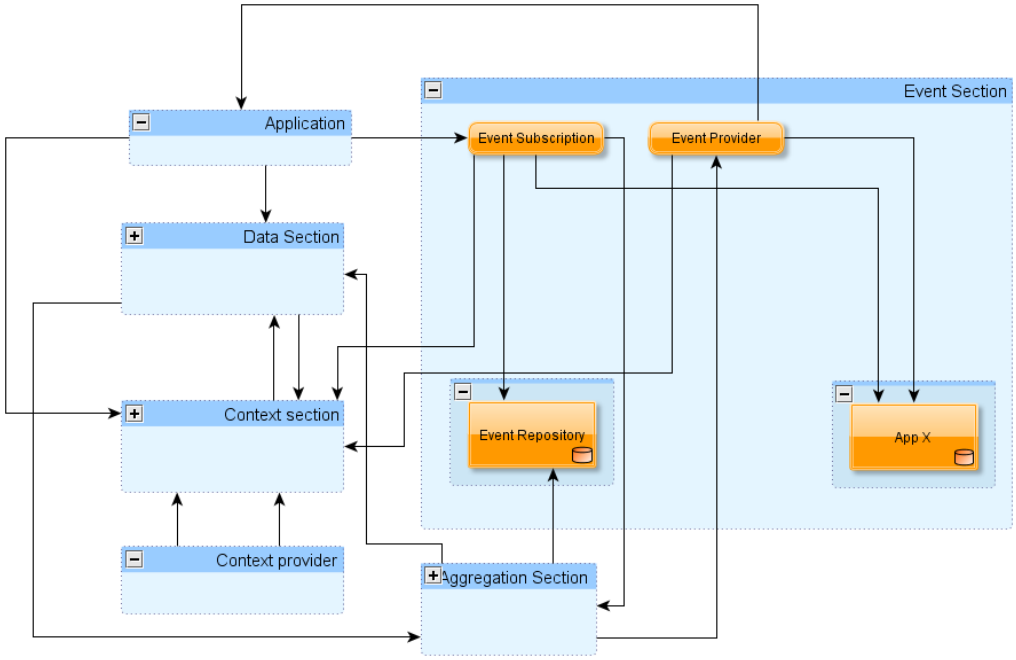


Figura 23: Representación de los módulos de Event Section y sus conexiones con el resto de secciones de la arquitectura

Aggregation Section

Debido a la complejidad de la agregación de eventos, se ha decidido desde el punto de vista del diseño, independizarla del reto de módulos del *Event Section*. Concretamente, los módulos de la *Aggregation Section* se encargarán de almacenar los diferentes agregados de eventos así como evaluarlos cuando se produzca un cambio en la información contextual que los componen. En la Figura 24 se muestran los diferentes módulos de los que consta el *Aggregation Section*.

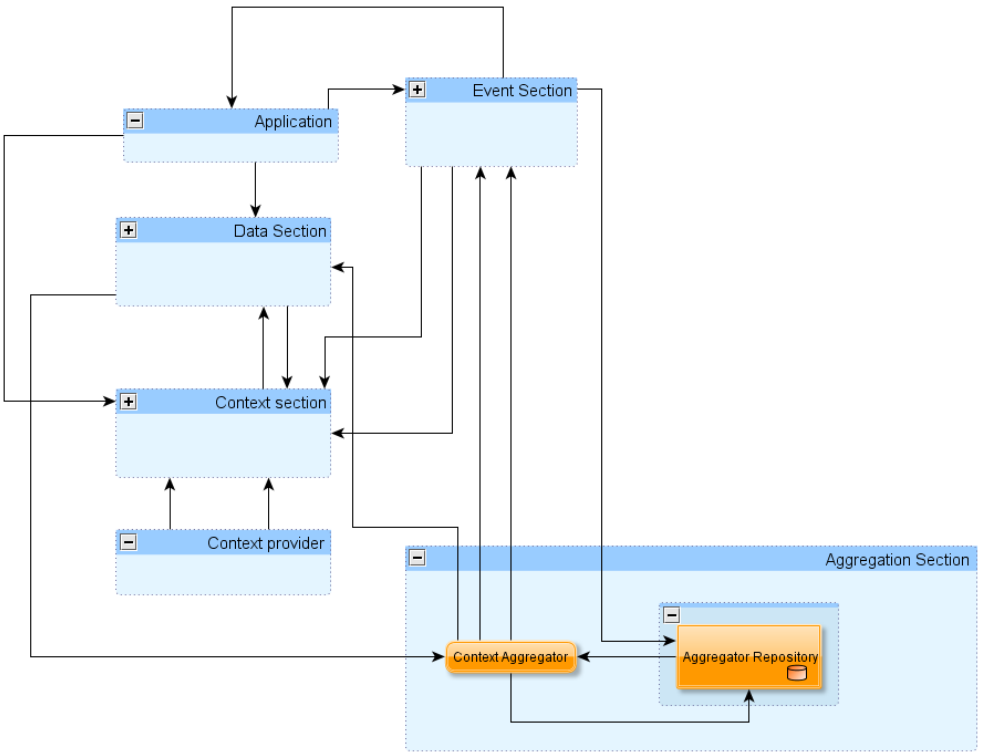


Figura 24: Representación de los módulos de Aggregation Section y sus conexiones con el resto de secciones de la arquitectura

En la sección anterior, no se hizo referencia en ningún momento a la evaluación de las condiciones de los eventos, ya que estos se realizarán en el módulo *Context Aggregator*, que será el primer de los dos módulos que compongan la sección actual. De este modo, sea cual sea la complejidad del evento o de la asociación de eventos a los que se suscriba una *Application*, éstos serán convertidos a agregado de eventos, que serán evaluados continuamente según las restricciones que se enuncian a continuación.

Con el fin de aumentar la eficiencia del sistema, un agregado tan sólo se evaluará cuando se produzca algún cambio en cualquier información que lo componga. En tal caso se dice que el agregado se evalúa. Una evaluación positiva, es decir, todas las restricciones impuestas en los eventos que componen el agregado se satisfacen, conllevará un mensaje en forma de evento a la *Application* que inscribió el evento, lo cual se describió en el apartado anterior.

Según la lógica expuesta, el módulo *Context Aggregator* deberá acceder por un lado al *Aggregator Repository*, el cual almacenará la información relativa a los diferentes eventos individuales que componen el agregado. Dado que el *Aggregator Repository* tan sólo almacena los identificadores de los eventos así como las operaciones lógicas que los relacionan, será necesario acceder al *Event Repository* para obtener la definición de cada uno de dichos eventos. Una vez obtenido los eventos completos de los agregados, se deberán evaluar para determinar si se satisfacen las condiciones impuestas por el usuario. Para ello el *Context Aggregator* debe recurrir al *Data Repository*, desde el que obtendrá los valores para cada uno de los valores contextuales que componen los eventos. Tras evaluar todos los eventos de un agregado, si la evaluación es satisfactoria, el *Context Aggregator* deberá informar al *Event Provider*, el cual será el encargado de enviar el evento a las aplicaciones inscritas al agregado concreto.

El segundo módulo del *Aggregation Section* encargado de almacenar los diferentes agregados presentes en el sistema se denomina *Aggregator Repository*. En él se almacenará, asociado a cada identificador de agregado, el conjunto de identificadores de eventos que lo componen. Además, se almacenará la operación lógica que relaciona los diferentes agregados, pudiendo de esta forma crear agregados del tipo *Evento1 OR Evento2 AND Evento3 OR NOT Evento4*, ofreciendo así sin duda un gran potencial lógico a las diferentes aplicaciones que usen el sistema de agregado de eventos.

APLICACIÓN

En esta sección se describe la capa superior del sistema desarrollado. En esta capa se sitúan las aplicaciones que reaccionan a los eventos generados por el núcleo del middleware. Gracias a los puntos de acceso al *Core*, las aplicaciones podrán suscribir un determinado agregado, de manera que se ejecute un determinado método de la aplicación cuando dicho agregado se evalúe satisfactoriamente. Para permitir al núcleo que acceda al método adecuado cuando se dispare un evento, es necesario que las *Applications* implementen la interfaz adecuada. Esto permitirá al *Core* enviar un mensaje que indique el agregado concreto que se ha disparado. Por otro lado, una aplicación podrá acceder al *Core* para consultar un dato particular (actual o histórico) o acceder directamente a un proveedor de contexto, sin necesidad de realizar una suscripción a ningún evento.

Diseño

Las aplicaciones tendrán dos métodos de acceso a la información, bien mediante eventos o a través de consultas directas al middleware. Diremos por tanto que los métodos de acceso a la información serán síncronos o asíncronos en función de si la llamada se hace mediante el acceso a la información almacenada o a eventos respectivamente.

Tanto en un caso como en el otro, la aplicación deberá registrarse previamente en el contexto del que desea obtener información, en caso contrario no se permitirá el acceso a la misma. Para ello deberá realizar una llamada al método correspondiente del *Context Discover*. Una vez realizada esta acción, la aplicación tendrá total libertad para acceder a la información proporcionada por el proveedor de contexto al que se ha inscrito.

En caso de que se desee realizar un acceso síncrono a la información, la aplicación tan sólo deberá acceder al punto de entrada del *Core* proporcionado por la interfaz *IDataQuery*. Dicha interfaz permite a la aplicación realizar consultas de varios tipos sobre el repositorio de almacenamiento de información contextual. El resultado de esta llamada será un *IDataContext* con la información solicitada o bien un conjunto de ellos en caso de que el resultado de la consulta sea más de una información contextual. Generalmente, para llevar a cabo una consulta de este estilo, la aplicación deberá conocer el identificador del *Context Provider* y de la *Function* que proporciona la información.

Por otro lado, la información se puede obtener de manera asíncrona, es decir, mediante eventos que se activen cuando se cumplan una serie de restricciones sobre el contexto. Para ellos, la aplicación deberá de hacer uso de los puntos de entrada proporcionados por el *Event Section*. En primer paso que deberá llevar a cabo la aplicación será la suscripción a un agregado. Sin embargo, previamente la aplicación debe haber configurado el agregado con el conjunto de eventos y las funciones relacionales que deben cumplirse para el disparo del evento. Una vez hecho esto por parte de la aplicación, podrá inscribir el objeto *IAggregate* compuesto en el *Core* del sistema a través del módulo *Event Subscription*. A partir de este momento, el *Core* se encargará de determinar el momento en el que se activa o desactiva el agregado inscrito. En el preciso momento en el que el *Core* deba comunicar un

evento a la aplicación, se realizará una llamada al método adecuado de la interfaz *IApplication* por parte del módulo *Event Provider*, enviando la información necesaria para identificar el agregado activo.

Mediante el flujo descrito anteriormente, se produce una independencia entre las aplicaciones y los proveedores de contexto gracias al uso del *Core*, el cual actúa de intermediario y gestor de la información, controlando en todo momento el flujo de información del sistema.

INTERCAMBIO DE INFORMACIÓN CONTEXTUAL

La mayoría de aplicaciones cuyo funcionamiento implica el uso de información contextual, necesitan conocer determinados aspectos del entorno que rodea al usuario en instantes puntuales o cada vez que éste cambia. Para ello es necesario que las aplicaciones que deseen consultar dicha información se suscriban al proveedor de contexto que la proporciona.

Un ejemplo claro lo tenemos cuando una aplicación necesita realizar una determinada acción, por ejemplo cuando cambia la posición del usuario que porta el dispositivo. Para ello la aplicación deberá inscribirse al evento de cambio de posición que un determinado C*ontext Provider* deberá ofrecer. Cuando el proveedor de contexto encargado de detectar cambios en la posición determine que el usuario ha variado su posición, informará al middleware de dicho evento y éste, a su vez, a todas las aplicaciones que se han suscrito al mismo de manera individual o mediante un agregado de más eventos, entre las que se encontrará la aplicación de nuestro ejemplo.

Otra alternativa que podría ser viable en otros casos es la consulta síncrona de una determinada información generada por un proveedor de contexto. De esta forma, el proveedor de contexto generará continuamente información que será almacenada por el *Core*. A su vez, otra aplicación leerá continuamente la información generada por la primera mediante la petición al núcleo del middleware de la información específica.

Por esto motivo, en el middleware se realizarán dos modelos distintos de notificaciones, como se comentó anteriormente, uno basado en comunicación asíncrona y otro basado en comunicación síncrona. A continuación se presenta un proceso BPMN que describe las etapas por las que deben transitar las aplicaciones para la obtención de información del middleware generado tanto para el modelo síncrono como para el asíncrono. En él se detallan todos los pasos que el middleware seguirá a la hora de inscribir una determinada aplicación al sistema de notificaciones de la plataforma. En el caso de la notificación síncrona, la plataforma inscribirá a la aplicación cliente en la lista de notificaciones del agregado almacenado por *Core*.

En las siguientes figuras se muestran los modelos de proceso que deberán seguirse a la hora de añadir y eliminar un proveedor de contexto al sistema, actualizar información contextual del sistema, añadir una aplicación al sistema y acceder de manera síncrona y asíncrona a la información. Todos estos diagramas ayudarán a determinar la información que deben intercambiar los distintos módulos del sistema.

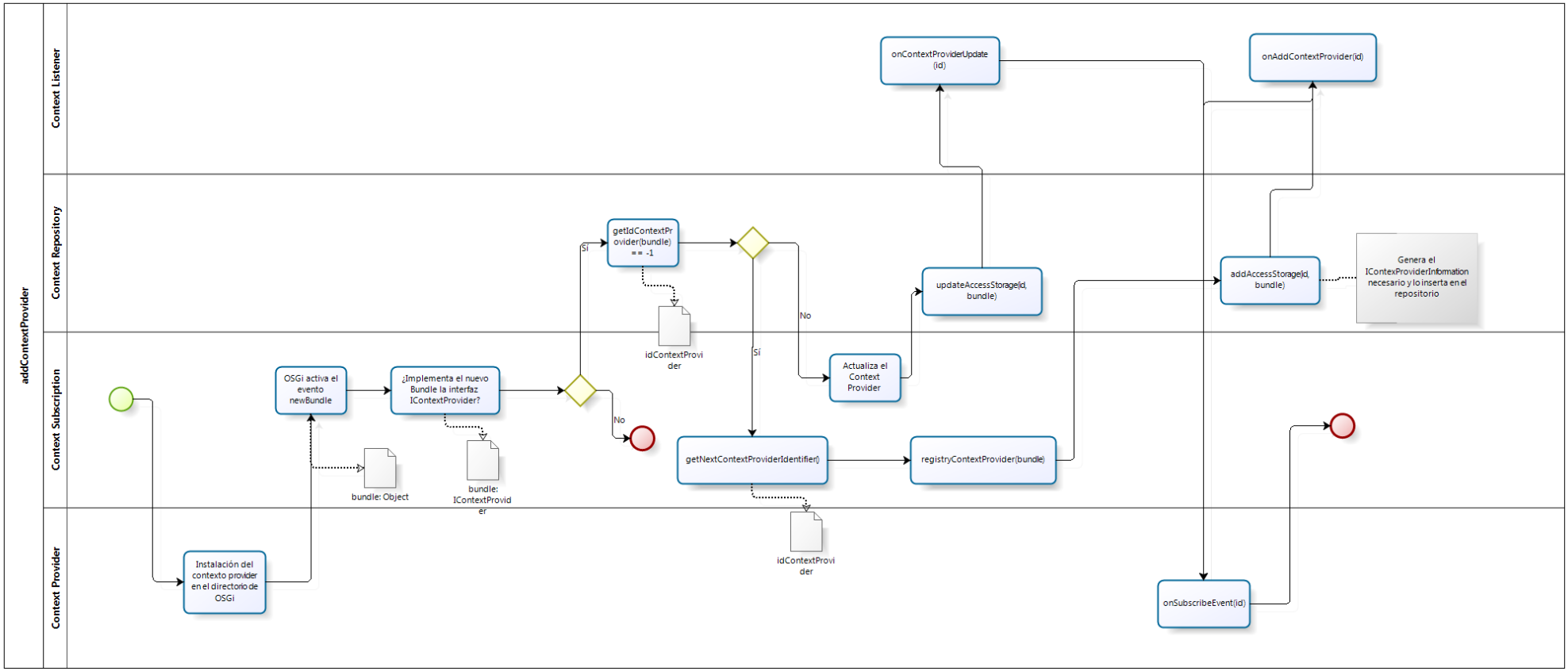


Figura 25: Modelo del proceso de adición de un proveedor de contexto al sistema

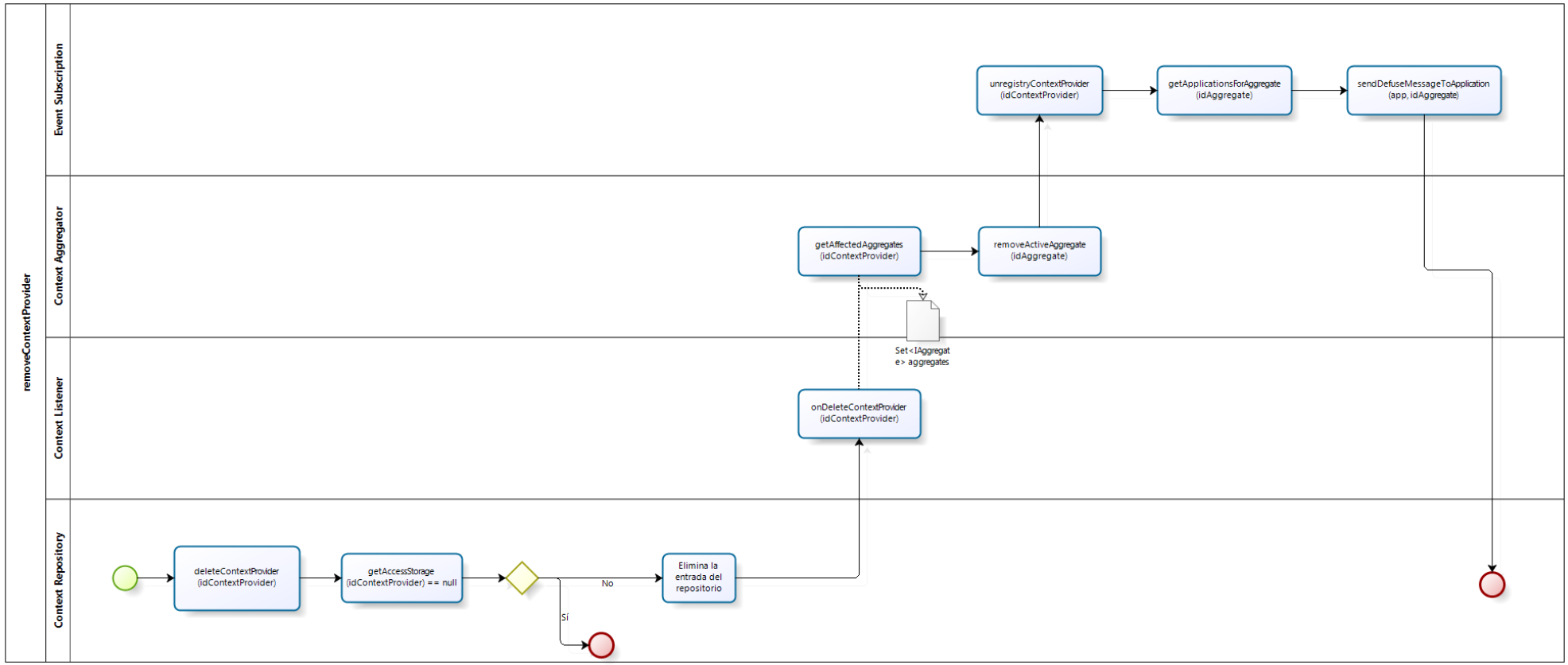


Figura 26: Modelo del proceso de eliminación de un proveedor de contexto del sistema

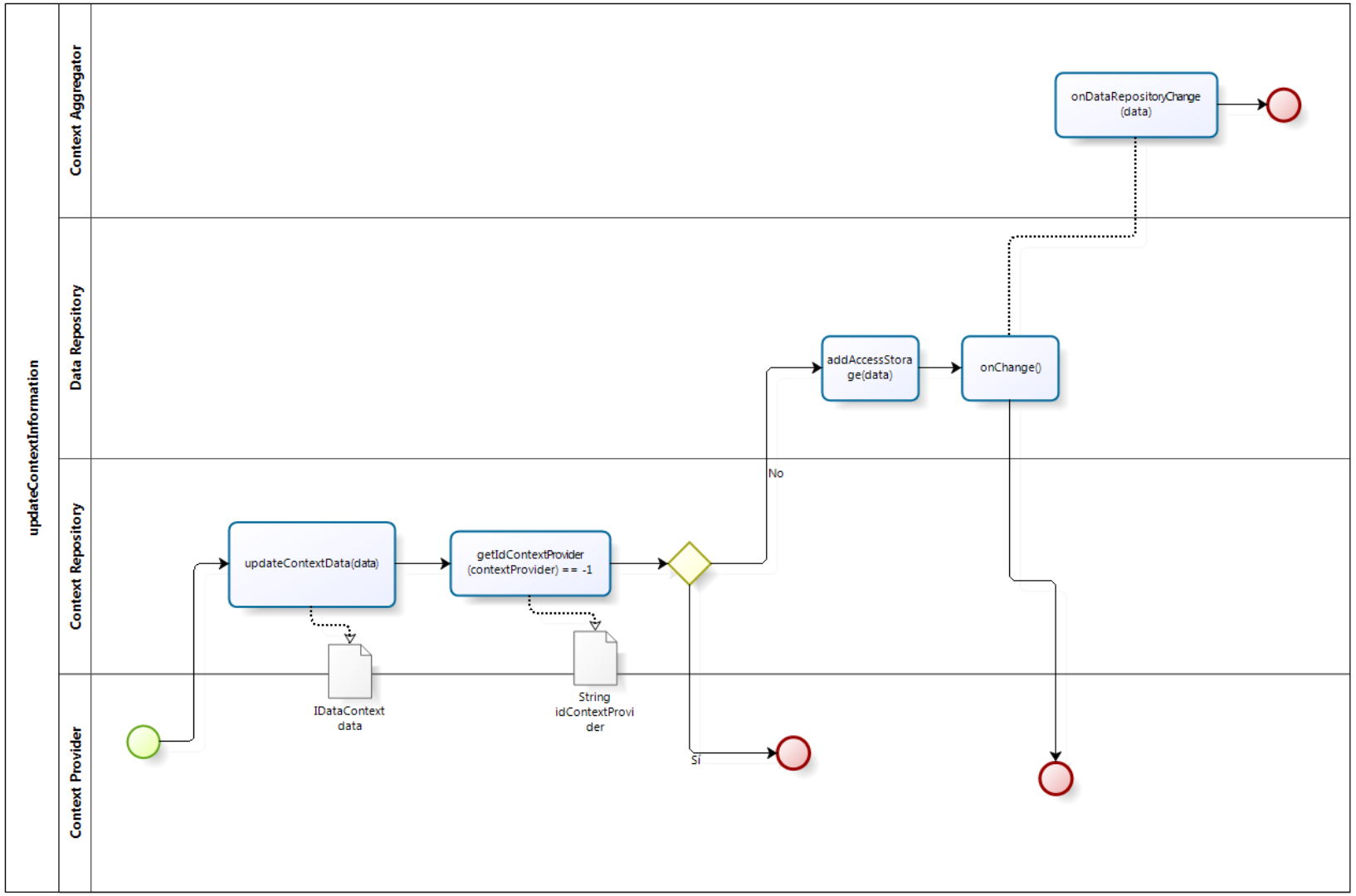


Figura 27: Modelo del proceso de actualización de información contextual del sistema

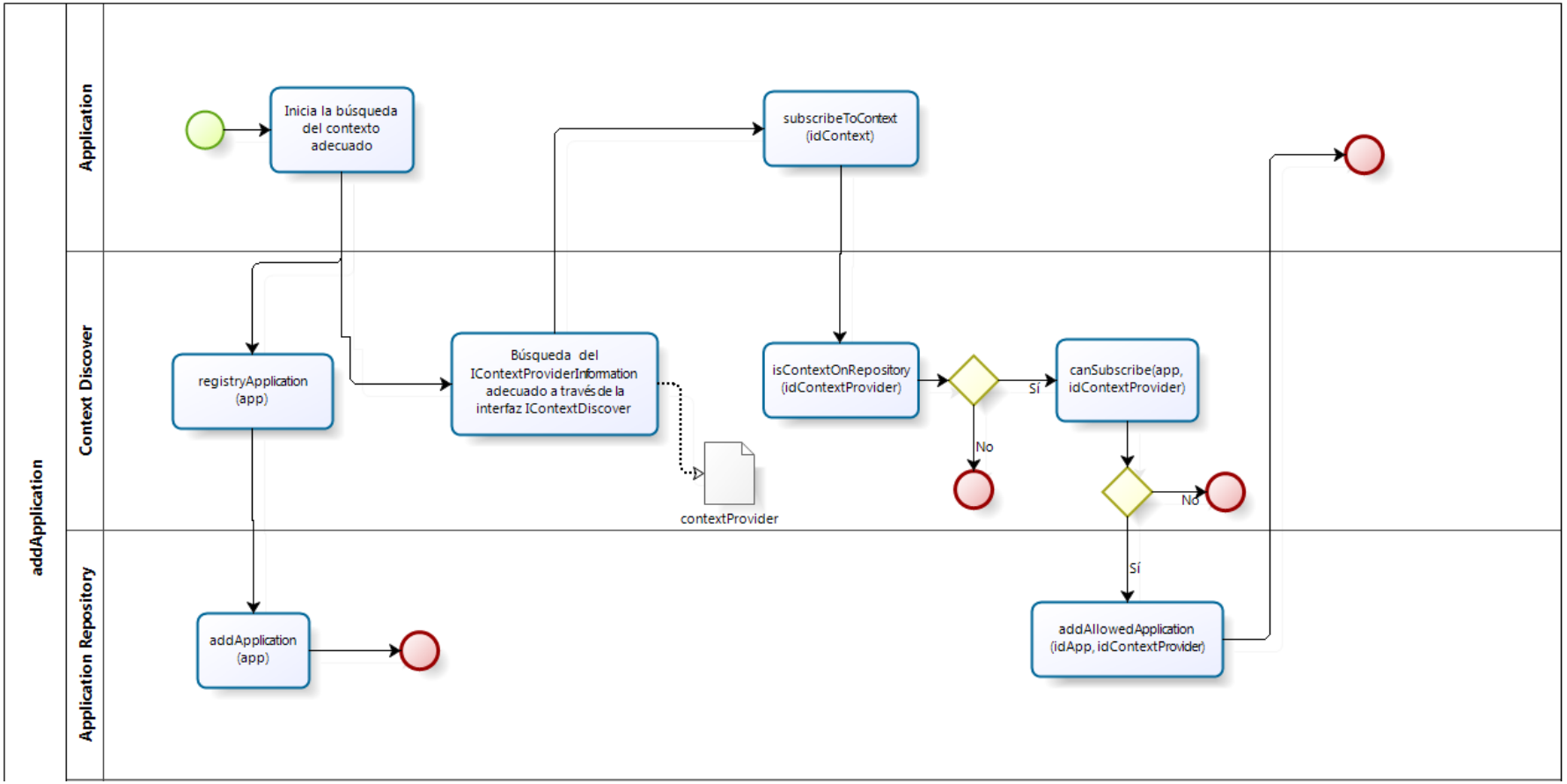


Figura 28: Modelo del proceso de inscripción de una aplicación al sistema

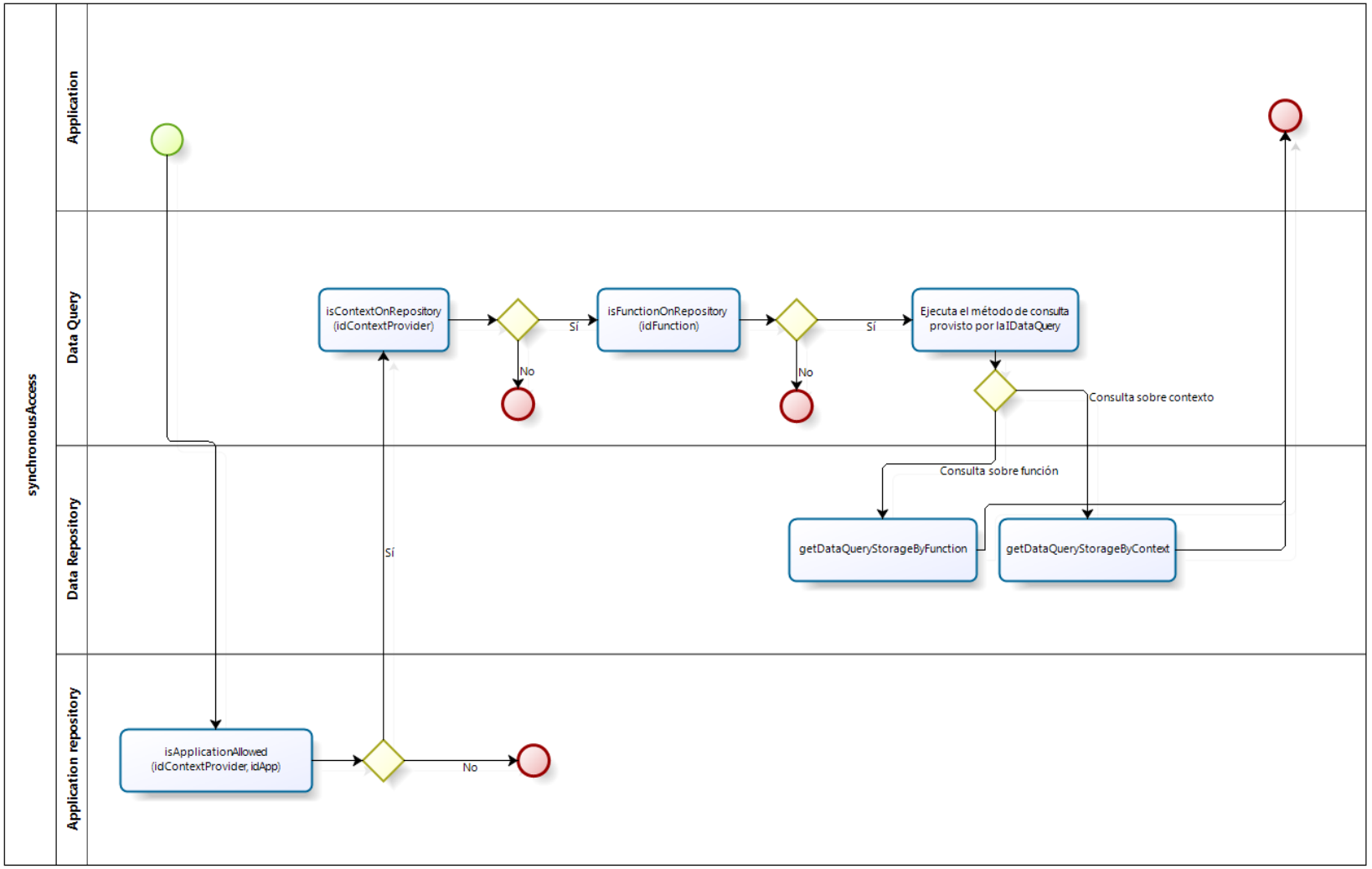


Figura 29: Modelo del proceso de acceso síncrono a la información

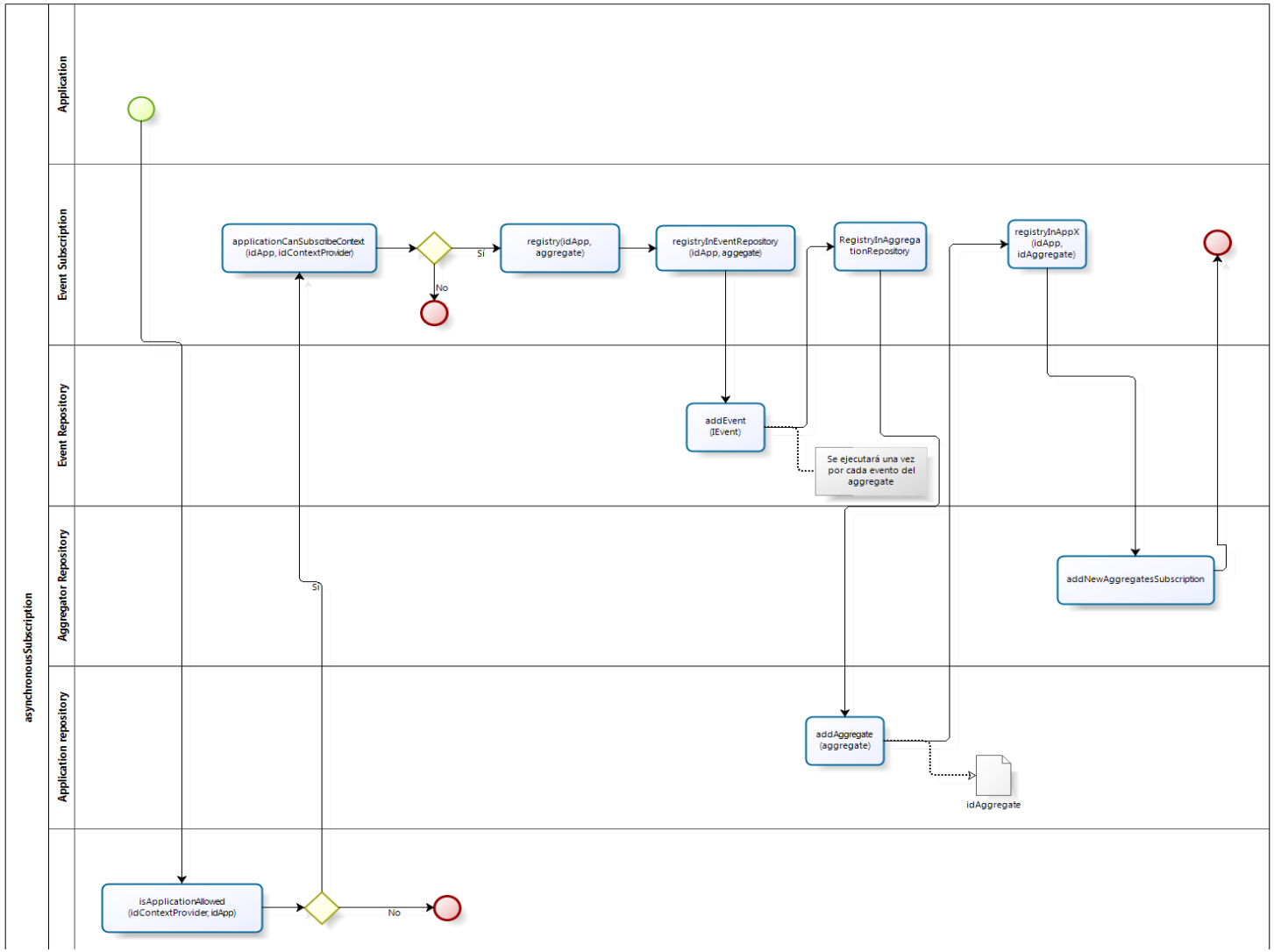


Figura 30: Modelo del proceso de acceso asíncrono a la información

A lo largo de las últimas secciones, se ha expuesto el funcionamiento y la estructura del middleware que permitirá el intercambio de información entre las aplicaciones y los proveedores contextuales. Este hecho será explotado posteriormente para la realización de aplicaciones que hagan uso de información contextual, concretamente para el reconocimiento de actividades de los usuarios y la detección de salidas de lugares cubiertos.

Gracias al middleware propuesto, las aplicaciones podrán reutilizar al máximo la información provista por los *Context Providers* lo que hace que el rendimiento del sistema se incremente debido a la reducción del tiempo de computación. Por otro lado, lo proveedores de contextos se independizan totalmente de las aplicaciones que harán uso de ellos, delegando el acceso a dicha información así como el almacenamiento y la gestión de la misma en el *Core* de la arquitectura.

RESUMEN

En este capítulo se ha llevado a cabo un middleware para dispositivos móviles basado en la plataforma *OSGi* que permite el intercambio de información contextual entre proveedores de contexto y aplicaciones. Este middleware permitirá independizar de manera completa la obtención de la información contextual a partir de los sensores físicos o lógicos del dispositivo, la cual se llevará a cabo en los proveedores de contexto, y el uso de dicha información por parte de las aplicaciones. Para ello se han definido una serie de módulos y de puntos de acceso tanto para las aplicaciones como para los proveedores contextuales. El uso del middleware permitirá a las aplicaciones no sólo consultar la información concreta, sino suscribirse a eventos complejos que alertarán a las aplicaciones cuando la información posea un valor determinado. En general, el middleware propuesto no sólo facilitará el desarrollo de aplicaciones contextuales, sino incrementará el rendimiento del dispositivo, dado que todo el proceso de recopilación de la información se hará de manera centralizada en los proveedores de contexto.

CAPÍTULO 4: RECONOCIMIENTO DE ACTIVIDADES

Es imposible amar algo ni odiar algo sin empezar a conocerlo.

- Leonardo Da Vinci -

En este capítulo pretendemos desarrollar un sistema dinámico, eficiente y fiable para el reconocimiento de las actividades que llevan a cabo los usuarios sometidos a un seguimiento de actividad. Además, el seguimiento propuesto será lo menos intrusivo posible, ya que los usuarios en general, son reacios a ser controlados mediante sistemas de tele-vigilancia tradicionales como por ejemplo, cámaras de vídeo o sensores repartidos por toda la vivienda. Además, un problema de estos sistemas es el ámbito reducido de operación.

Los métodos que se describen en este capítulo determinarán la actividad de la persona allá donde vaya. Esto se puede realizar gracias a que los dispositivos móviles actuales incorporan sensores de acelerometría, lo que significará que el usuario podrá llevar el dispositivo en todo momento. Por ello el ámbito de nuestra aplicación no será la casa del individuo, su trabajo o su lugar de entrenamiento, sino que podrá ser controlado allá donde esté.

Nuestro objetivo es registrar todos los movimientos que realice el usuario y clasificarlos en distintas actividades como, por ejemplo, caminar, andar, correr, saltar, bajar escaleras, subir escaleras o caídas. Una vez hecho esto, el resultado de la clasificación será visible a través de un portal web a los familiares del usuario, a los médicos que llevan el historial del individuo y a todas las personas que el administrador del sistema estime oportuno.

Además, el seguimiento del usuario se debe llevar a cabo sin que sea una carga demasiado grande para el propio individuo. Como veremos en el apartado de estado del arte, existen algunos dispositivos que permiten hacer un seguimiento del usuario, pero el principal inconveniente es que sólo están disponibles mediante hardware propietario. Además de este problema, nos encontramos con que el usuario debe llevar encima un dispositivo adicional, con la incomodidad y el riesgo de olvido que esto provoca.

En contraposición a lo anterior, nuestro sistema, en el lado del usuario, pretende estar integrado en el dispositivo móvil del individuo sometido al seguimiento. El motivo de esta decisión es que el móvil forma parte (cada día más) de la forma de vida de los usuarios y de esta forma, el riesgo de olvido o pérdida es mucho menor que con un dispositivo adicional. Además de esto, la potencia y la versatilidad de estos dispositivos hace posible que puedan ser ampliadas las posibilidades del sistema.

ESTADO DEL ARTE

En los últimos años, gracias en gran medida al aumento del interés de monitorizar ciertos sectores de la población, como ancianos o usuarios con movilidad restringida, los sistemas de reconocimiento de actividades han experimentado un aumento tanto en número como en la calidad de los resultados. Sin embargo, como analizaremos en esta sección, la mayoría de ellos emplean complejos sistemas de

procesamiento, que hace que no sea posible llevarlos a cabo en un dispositivo móvil de propósito general.

El cálculo de la actividad física de un usuario en base a los datos obtenidos a partir de un acelerómetro es un tema de investigación actual. Existen varios sistemas capaces de realizar esta tarea y entre todos ellos destacaremos los más importantes y los analizaremos para observar similitudes y diferencias con respecto a nuestra propuesta.

La primera diferencia que podemos observar entre los distintos sistemas desarrollados hasta el momento es la colocación y el dispositivo empleado para tomar los datos. Existen sistemas que emplean hardware propietario (Activity Monitoring System using Dynamic Time Warping for the Elderly and Disabled people, 2009) (Activity Recognition from Accelerometer Data, 2005), mientras que otros emplean hardware de propósito general (Activity Recognition using Wearable Sensors for Elder Care, 2008), como es el caso del trabajo expuesto en esta tesis doctoral. Obviamente, el emplear hardware genérico supone un beneficio para el usuario, ya que el coste de dichos dispositivos y la versatilidad son bazas a su favor. Sin embargo, estos dispositivos tienen un inconveniente: las limitaciones en la calidad de los datos. Los acelerómetros integrados en dispositivos genéricos, como es el caso de los móviles de última generación, poseen, en general, una menor calidad en la obtención de datos frente a los integrados en dispositivos específicos. De esta forma, los estudios que emplean hardware general deben centrarse en solventar esta dificultad.

Otra diferencia que podemos encontrar entre las distintas propuestas es la finalidad de las mismas. En (Activity Recognition from Accelerometer Data on a Mobile Phone, 2009) podemos ver que el sensor de acelerometría se sitúa en un guante, el cual debe llevar puesto el usuario y es capaz de reconocer multitud de actividades en función del movimiento de la mano. Sin embargo, en otros estudios como (Activity Recognition using Wearable Sensors for Elder Care, 2008) el sensor se sitúa en el bolsillo del usuario. En este punto podemos preguntarnos qué método es más eficaz. En base a los resultados, (Activity Recognition from Accelerometer Data on a Mobile Phone, 2009) posee una mayor precisión y es capaz de reconocer un mayor número de actividades que el propuesto en (Activity Recognition using Wearable Sensors for Elder Care, 2008). Sin embargo, esta solución puede resultar incómoda para el usuario, por lo que el sistema elegido debería seleccionarse en base a las necesidades requeridas.

Además de las descritas anteriormente, existen otra serie de propuestas para la vigilancia de personas (Classification Technique of Human Motion Context based on Wireless Sensor Network, 2005) que no sólo se basan en acelerometría, sino que añaden otros elementos como, por ejemplo, cámaras de vídeo vigilancia.

Por otro lado, la precisión de otros métodos alcanza unos valores realmente interesantes, aunque se basan en el reconocimiento multipunto (Comparative study on classifying human activities with miniature inertial and magnetic sensors, 2010), es decir, el usuario necesita llevar diferentes sensores acelerométricos repartidos a lo largo del cuerpo. Esto hace que el sistema se convierta en un elemento más intrusivo y a veces, no es sencilla la colocación correcta de los sensores.

A pesar de que estos sistemas son empleados generalmente en adultos, su uso se ha extendido al ámbito de la salud de los más pequeños. En este aspecto, los últimos avances en este campo han hecho posible la monitorización de un bebé residente en una unidad de cuidados intensivos, gracias al uso de un conjunto de sensores acelerométricos colocados en las extremidades del paciente (ISWC 2010: The

Latest in Wearable Computing Research, 2011), obteniendo así todos los movimientos realizados y registrándolos en el sistema. Un ejemplo de este tipo de sistema se puede observar en la Figura 31.

Figura 31: Uso de acelerometría en pacientes neonatos

En general, una característica común de todos los métodos proporcionados por la bibliografía es que emplean métodos continuos para la obtención de las diferentes actividades. Este tipo de métodos, como se verá a lo largo de esta sección, poseen una serie de inconvenientes sobre la complejidad del reconocimiento, lo cual hace que su uso en dispositivos móviles sea complejo computacionalmente. En cambio, a lo largo de este capítulo se presentarán una serie de métodos alternativos a los revisados, los cuales poseen una serie de ventajas en cuanto a la complejidad, al tiempo y a la precisión del reconocimiento.

OBTENCIÓN DE LOS DATOS

En esta sección se explicará la manera en que son obtenidos los datos necesarios para la realización del proceso de detección de actividades. Además, se detallarán las variables del sistema en base a las cuales se realizará el proceso de aprendizaje y reconocimiento de actividades.

TRABAJO CON SENSORES EMBEBIDOS

El sistema desarrollado para la detección de actividades, se basará en los valores obtenidos mediante el acelerómetro del dispositivo para determinar la actividad que está llevando a cabo el individuo. Un acelerómetro es un elemento instalado en el dispositivo cuyo fin es medir las aceleraciones producidas en éste. En la actualidad estos sensores están integrados en una gran cantidad de dispositivos móviles. Además, los acelerómetros son cada vez más precisos y su frecuencia de lectura de datos es más elevada. Por lo tanto es posible obtener más datos y de manera más precisa, por lo que la capacidad de señalar la actividad que el individuo sometido a seguimiento está llevando a cabo será más fiable.

El principio que subyace detrás del funcionamiento de estos elementos es un resorte amortiguado con una masa en el extremo. De esta forma, la aceleración producida sobre el dispositivo es directamente proporcional a la distancia entre la masa y el punto de referencia. Generalmente el punto de referencia se corresponde con 0g, es decir con la aceleración nula, la cual se alcanza durante un proceso de caída libre del dispositivo. En los extremos se encuentra la aceleración mínima y la aceleración máxima, las cuales dependerán de las características concretas del sensor. En la Figura 32

se representa el funcionamiento básico de un acelerómetro mecánico de resorte sometido a una aceleración de -1g.

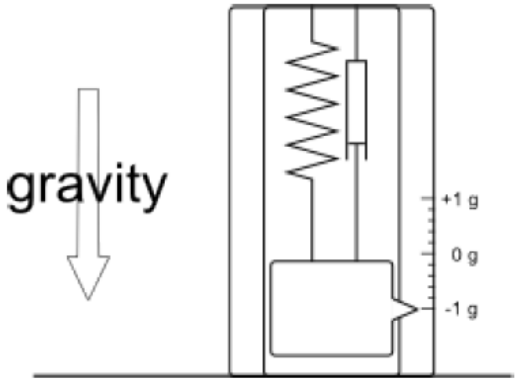


Figura 32: Funcionamiento de un acelerómetro mecánico

Evidentemente, el tamaño necesario para la creación de este tipo de acelerómetros es demasiado elevado como para ser instalados en dispositivos móviles. Para este fin se emplean los denominados acelerómetros piezoeléctricos, los cuales se basan en que al comprimir un retículo cristalino, se produce una carga eléctrica proporcional a la fuerza aplicada. Este elemento, al que se denomina masa gravitacional, es introducido en el interior de dos placas conductoras, por lo que al mover la masa se produce una diferencia de potencial entre ellas. Esta diferencia de potencial da lugar a la lectura de la aceleración a la que está sometido el dispositivo.

Existen diversos tipos de acelerómetros en función del número de ejes sobre los que sea posible medir las aceleraciones. Partiendo desde acelerómetros mono axiales donde la aceleración se mide tan sólo en un eje, hasta los triaxiales, en los cuales las aceleraciones son medidas en cada uno de las direcciones ortogonales del dispositivo.

El estado de reposo de un acelerómetro triaxial instalado en un dispositivo móvil suele corresponder a una aceleración de 1g en uno de los ejes y 0g en el resto. El valor de 1g generalmente en el eje vertical, se corresponde con la aceleración gravitacional. La Figura 33 muestra los valores obtenidos por un acelerómetro triaxial durante un periodo temporal, lo cual da como resultado una representación de la señal a lo largo del tiempo, a lo que llamaremos generalmente perfil de acelerometría. El estudio de dicho perfil mediante diferentes técnicas que se describirán a lo largo de esta tesis dará como resultado la determinación de la actividad física llevada a cabo por un determinado usuario.

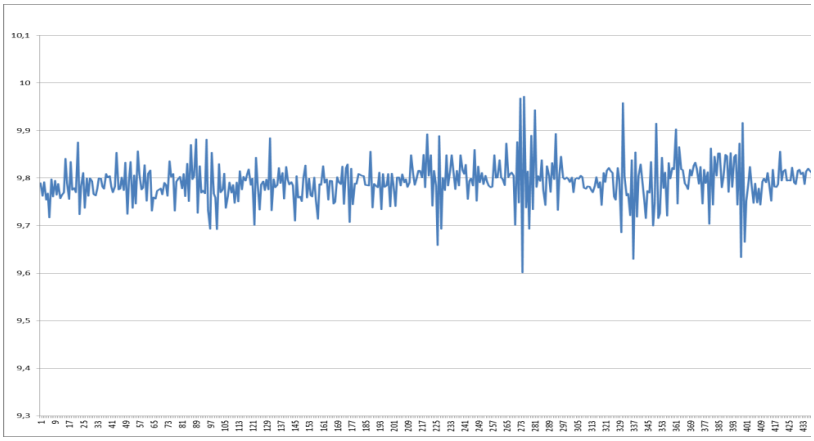


Figura 33: Señal de acelerometría

Otro beneficio de los sistemas de detección de actividad basados en acelerometría es el reducido coste energético que se produce en el dispositivo. Esto es un aspecto esencial cuando hablamos de desarrollar aplicaciones para dispositivos móviles, ya que el cuello de botella de éstos es la batería. Además el usuario no necesitará recargar el dispositivo constantemente, proporcionando así un uso más cómodo del sistema.

Aunque los datos obtenidos son completamente válidos, será necesario realizar un filtrado de los datos previo a la clasificación. El objetivo del filtrado es eliminar el ruido de la señal obtenida. Aunque, generalmente, el ruido del sensor de acelerometría es prácticamente despreciable, existe un tipo de ruido que puede afectar seriamente a la clasificación de la actividad, producido por las vibraciones del dispositivo al ser transportado por el usuario.

Un aspecto importante a tener en cuenta en nuestro sistema es que no impone el lugar donde el usuario debe situar el dispositivo, a diferencia de muchos trabajos relacionados, sino que el usuario puede portar el dispositivo donde desee. Esto aumenta la complejidad del desarrollo dado que la sujeción del dispositivo móvil podría no ser óptima y podrían producirse vibraciones indeseadas cuando se está desarrollando la actividad. Este es el motivo por el que es necesario realizar un filtrado para obtener valores válidos.

Los acelerómetros que actualmente se instalan en los dispositivos móviles, suelen medir las aceleraciones producidas en los tres ejes del dispositivo (acelerómetro triaxial). A lo largo de la sección titulada *Elección* del conjunto de datos y variables se justificará el uso del módulo de la señal de acelerometría por parte del sistema de reconocimiento de actividades y se estudiarán las diferencias entre esta técnica y la basada en el tratamiento de la señal triaxial. En la Figura 34 podemos ver una representación gráfica de los ejes de acelerometría en un dispositivo móvil genérico.

Definimos módulo del vector de acelerometría en el instante t y lo denotamos como $|a_t|$ al valor dado por la siguiente igualdad:

$$|a_t| = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$$

donde a_x, a_y y a_z son los valores de la aceleración en un instante determinado en los ejes x, y, z respectivamente.

Figura 34: Acelerometría triaxial

Para desarrollar el reconocimiento de actividad del usuario dividiremos las lecturas de acelerometría en ventanas temporales, de tal forma que el análisis de una ventana temporal dará como resultado una posible actividad que el usuario está llevando a cabo durante el periodo de tiempo que engloba dicha ventana. Sin embargo, previamente a todo este procesamiento será necesario aplicar un filtro a la señal procedente de los sensores

En esta sección serán introducidos los conceptos básicos con los que trabajaremos a lo largo de esta tesis, sentando así las bases para la aplicación de los métodos expuestos a lo largo de todo el proceso de reconocimiento.

VENTANAS TEMPORALES

Cuando trabajamos con datos que son obtenidos de manera continua, sin un tamaño determinado y por una técnica de *streaming*, es necesario segmentar dichos datos con el fin de estudiarlos de manera global en un determinado intervalo temporal. Dicha técnica hace que sea posible trabajar con los datos agrupados de una manera aislada, pudiendo hacer todos los cálculos necesarios y aplicarlos sobre el conjunto finito. Por otro lado, el uso de ventanas temporales reduce la complejidad del sistema, dado que los datos serán procesados en lotes sin que sea necesario actualizar de manera continua los resultados obtenidos al aplicar una determinada técnica al histórico de los datos.

De igual manera, para el desarrollo del sistema descrito anteriormente, es necesario tener en cuenta una serie de consideraciones, las cuales deben ser satisfechas para la realización de un sistema en tiempo real eficiente.

- No existen conocimientos sobre eventos futuros: todo evento actual debe ser reconocido en base a datos procedentes del pasado o del momento en el que se lleva a cabo la acción. Esto implica que este tipo de sistemas dinámicos no pueden basarse en eventos futuros para predecir eventos actuales.
- El tamaño de los datos almacenados a través del buffer de reconocimiento debe ser limitado. La consecuencia directa de esta restricción es que los datos deben ser acotados para su procesamiento y posteriormente eliminados de la memoria del sistema. Aunque la memoria de los dispositivos móviles actuales es elevada, sigue siendo insuficiente para el almacenamiento de gran cantidad de datos los cuales, deberán ser procesados posteriormente saturando a su vez la capacidad del procesador del dispositivo.
- El tiempo disponible para llevar a cabo el procesamiento de los datos es limitado. Esta limitación se establece debido a que los datos deben ser procesados en bloque, de manera que a la finalización del bloque actual, el procesamiento del bloque de datos anterior debe haber finalizado. En otro caso, cabría el riesgo de que fueran solapados los diferentes procesos de

reconocimiento, dando como resultado una saturación del sistema y un uso elevado de batería debido a la continua creación y eliminación de hilos de proceso.

Para satisfacer las anteriores restricciones detectadas en sistemas de procesamiento en tiempo real, se ha decidido hacer uso de ventanas temporales, las cuales permiten acotar el volumen de datos a procesar, controlar el tiempo de procesamiento y evitar el solape, así como almacenar las últimas evidencias sobre el origen de datos escogido. Esta técnica es especialmente útil cuando el número de datos es muy grande o cuando la frecuencia de actualización de los mismos es elevada. En el caso concreto del reconocimiento de actividades, como hemos descrito anteriormente, los datos proceden del sensor de acelerometría de una manera ininterrumpida. Además, la frecuencia de trabajo de estos sensores puede llegar a los 100Hz dependiendo del modelo concreto, por lo que no es viable reprocesar los datos cada vez que sea leído un nuevo valor.

Una ventana temporal no es más que la división del conjunto total de datos en conjuntos más pequeños cuyo factor común es que el primer dato y el último dato son obtenidos con una diferencia temporal determinada, a la que llamamos tamaño de la ventana. De esta forma, podemos almacenar los datos procedentes del sensor de acelerometría hasta que la ventana temporal alcance un tamaño determinado. En este punto, los datos almacenados podrán ser procesados mientras que la siguiente ventana temporal se sigue almacenando, sin que se produzca ninguna saturación del sistema de reconocimiento.

En la Figura 35 podemos ver la representación gráfica de un conjunto de datos dividido en dos ventanas temporales disjuntas. En el eje de coordenadas se representa el valor del módulo de acelerometría, mientras que en el eje de abscisas se muestra el número de datos obtenidos.

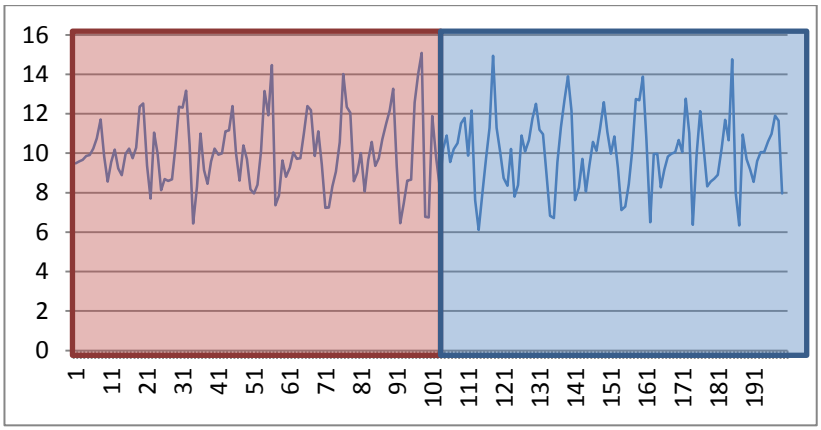


Figura 35: Ventanas temporales aplicadas a datos de acelerometría

La elección del tamaño de la ventana temporal no es en absoluto un proceso arbitrario, sino que debe cumplir una serie de requisitos. En el caso particular del reconocimiento de actividades, los requisitos vienen impuestos por el retraso provocado en el reconocimiento. Mientras se almacena la ventana temporal, obviamente el usuario está realizando una actividad, y es al finalizar el tamaño de la ventana temporal cuando se procesarán los datos para determinar dicha actividad física. De esta forma, se produce un retraso en el reconocimiento que sigue la siguiente relación:

$$delay = tam(v) + k * f(tam(v))$$

donde $tam(v)$ indica el tamaño de la ventana temporal en segundos, k representa una constante de procesamiento que dependerá de las características del dispositivo, y f representa el tiempo empleado en el proceso de reconocimiento en función del tamaño de la ventana temporal. De esta forma, si elegimos una ventana temporal demasiado grande, estaríamos comprometiendo la fluidez en el reconocimiento, puesto que el retraso producido entre la identificación de la actividad y la realización por parte del usuario sería demasiado elevado. Por tanto, siguiendo este criterio deberíamos escoger una ventana temporal lo más pequeña posible, que además nos aporte suficiente información para llevar a cabo nuestro objetivo, es decir, asegurar que la clasificación de la actividad se realiza de manera correcta, para lo cual necesitaremos un conjunto de datos lo suficientemente elevado. En consecuencia, cuantos más datos sean obtenidos capaces de caracterizar la actividad que el usuario está llevando a cabo, mayor precisión obtendremos en la clasificación. Como podemos observar, este criterio entra en conflicto con el expuesto anteriormente, por lo que deberemos llegar a un compromiso entre el retraso provocado por una ventana demasiado grande y la poca fiabilidad en el reconocimiento por la elección de una tamaño de ventana demasiado pequeño.

Para el proceso llevado a cabo durante la recopilación de datos procedentes del sensor de acelerometría, se ha considerado un tamaño de ventana temporal de 5 segundos. Esto hace que el retraso en el reconocimiento sea de 5 segundos más el tiempo de procesado, lo cual no es demasiado elevado. En consecuencia, la usabilidad del sistema así como la precisión en el reconocimiento será elevada, sin que sea apreciable un retraso excesivo entre la realización de la actividad por parte del usuario y el reconocimiento de la misma. Además, este tamaño hace que se alcance una precisión lo suficientemente elevada como para hacer del método de reconocimiento de actividades que será expuesto a lo largo de este capítulo, un sistema preciso y totalmente eficaz.

Existe otro método distinto al anterior para la aplicación de ventanas temporales sobre un conjunto de datos, el cual se conoce como *solapamiento de ventanas temporales*. En esta técnica se divide el conjunto de datos en ventanas temporales de longitud constante, al igual que en el caso anterior, con la salvedad de que dichas ventanas poseen un determinado solapamiento entre ellas. En la Figura 36 se puede observar gráficamente el método de aplicación de esta técnica.

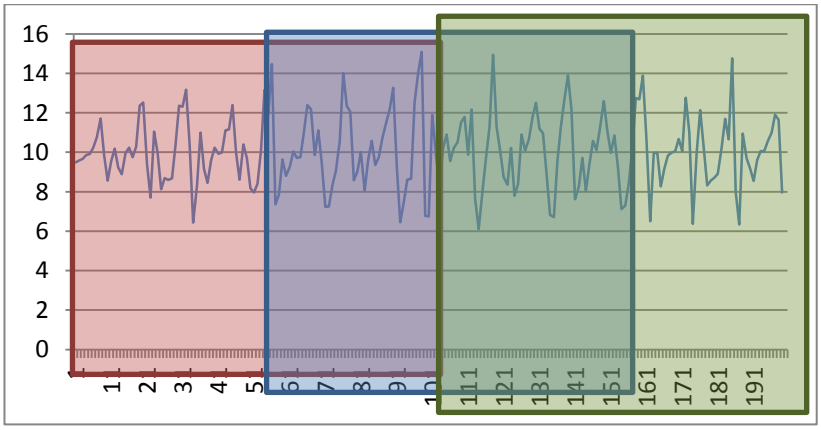


Figura 36: Solapamiento de ventanas temporales

Generalmente se suele aplicar un máximo de un solapamiento entre ventanas. El uso de este método permite eliminar el error producido en los saltos de ventanas en el caso de la técnica sin solapamiento. Este error se produce debido a que el proceso de reconocimiento se basa en el estudio del histórico de los datos, por lo que en las transiciones de ventanas, los primeros datos carecen de histórico, puesto que coinciden con el comienzo de las ventanas. De esta manera, el uso de ventanas

temporales con solapamiento permite tener en todo momento un conjunto de datos previos al comienzo natural de la ventana.

Sin embargo, existe un problema derivado del uso de esta última técnica, y es el aumento del número de datos a procesar debido a que algunos son procesados por duplicado. Esto podría minimizarse si realizamos un solapamiento parcial, es decir, si en lugar de procesar por duplicado todos los datos procedentes de acelerómetro, eliminamos parte de este procesamiento. Se ha determinado que el principal inconveniente del uso de ventanas disjuntas es la transición entre ventanas, por lo que podemos hacer que las ventanas se solapen en un tiempo menor, tan sólo en el límite entre ellas.

Si observamos la Figura 37 podemos apreciar que el solapamiento se produce tan sólo en el límite de las ventanas, así que los datos centrales sólo serán procesados una vez.

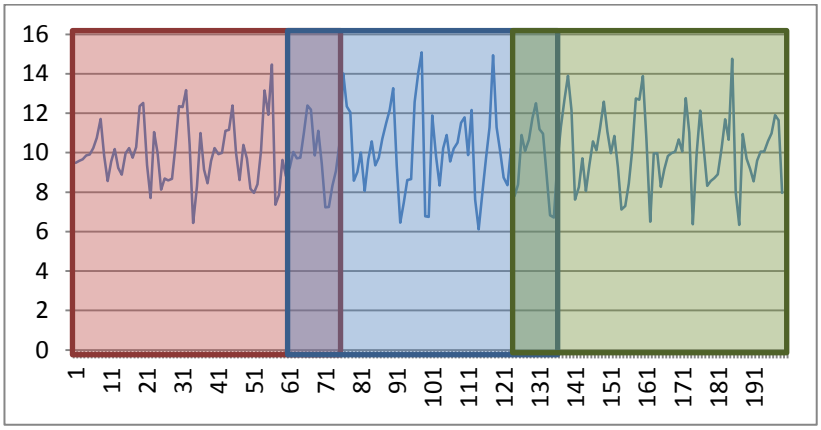


Figura 37: Solapamiento de ventanas temporales límites

En general, cuando se emplean ventanas temporales con solapamiento puede ser aumentado el tamaño de la ventana, puesto que desde el punto de vista del reconocimiento de actividades los resultados serán obtenidos al final de cada ventana, con independencia del comienzo de la misma. Este efecto se puede observar en la Figura 38, donde la tasa de datos es mayor en el caso del sistema de ventanas temporales con solapamiento a pesar de tener un mayor tamaño de ventana.

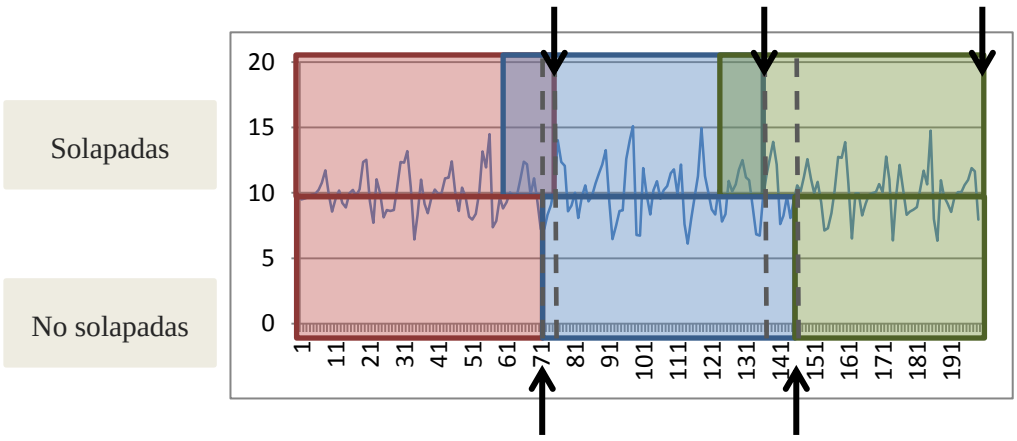


Figura 38: Tasa de rendimiento temporal del sistema de reconocimiento de actividades

Una de las principales tareas que se debe llevar a cabo a la hora de realizar un sistema de reconocimiento de actividad y de manera general, cualquier sistema de clasificación automática, es la elección del conjunto de datos que servirán para que el sistema sea capaz de reconocer y aprender las actividades que realizan los usuarios. Para ello, como hemos comentado anteriormente, trabajaremos con los datos proporcionados por los sensores de acelerometría del dispositivo móvil.

En la actualidad, los acelerómetros permiten registros continuos de varios días o una semana de las actividades del sujeto en su propio entorno, arrojando una información más ajustada a la realidad, que la obtenida únicamente mediante cuestionarios de actividad física (G, y otros) (Ness AR, 2007). Los acelerómetros miden movimientos corporales en términos de aceleración, información que puede ser usada para estimar la actividad física a lo largo del tiempo (Crouter SE, 2006), además de ser posible obtener información acerca de la frecuencia, intensidad y duración de la actividad física (KY, y otros, 2005).

En concreto, en el reconocedor de actividades que será expuesto a lo largo de este trabajo se emplearán las variables que se presentan a continuación. Sin embargo, como veremos más adelante, algunas de estas variables serán eliminadas debido al alto grado de correlación entre ellas con el fin de disminuir el tiempo de procesamiento.

De esta forma, las variables estadísticas que se describirán a continuación serán aplicadas a cada una de las ventanas temporales generadas a lo largo del proceso de reconocimiento de la actividad por parte del sistema.

- *Media aritmética*

  Dados *n* valores de acelerometría $\{a_1, a_2, \dots, a_n\}$, la media aritmética se define como:

$$\bar{a} = \frac{1}{N}\sum_{i=1}^{n} a_i$$

  Generaremos tres modificaciones sobre la media aritmética y en general, sobre todos los estadísticos descritos en esta sección. En primer lugar generaremos el valor del estadístico para cada uno de los ejes de acelerometría independientemente, por lo que tendríamos tres valores para el estadístico, uno para el eje X, otro para el eje Y, y el último para el eje Z. Además, generaremos otro valor más para el módulo de la acelerometría, es decir:

$$|a_t| = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$$

  donde $a_x, a_y \; y \; a_z$ son los valores medios para cada uno de los ejes.

- *Mínimo*

  Se refiere al valor mínimo de los datos dentro de la ventana temporal procesada. Se generarán cuatro valores, uno para cada eje y otro para el módulo:

$$a_v^{min} = min\{a_1, a_2, \dots, a_n\}$$

donde *v* es *X,Y,Z* o el módulo de la señal de acelerometría.

- *Máximo*

Se refiere al valor máximo de los datos dentro de la ventana temporal procesada. Se generarán de nuevo cuatro valores, uno para cada eje y otro para el módulo de acelerometría:

$$a_v^{max} = max\{a_1, a_2, \dots, a_n\}$$

donde *v* podrá ser *X,Y,Z o M*, correspondientes al máximo para cada eje de datos de acelerometría triaxial o para el módulo en caso de que se trabaje con acelerometría modular.

- *Mediana*

Valor del dato acelerométrico que deja el mismo número de por encima y por debajo de su valor una vez ordenados éstos. De acuerdo con esta definición el conjunto de datos menores o iguales que la mediana representarán el 50% de los datos, y los que sean mayores que la mediana representarán el otro 50% del total de datos de la muestra.

De esta forma, siendo $\{a_1, a_2, \dots, a_n\}$ los datos de una muestra ordenada en orden creciente y designando la mediana como *Me*, distinguimos dos casos:

Si n es impar, la mediana es el valor que ocupa la posición (n + 1) / 2 una vez que los datos han sido ordenados (en orden creciente o decreciente), porque éste es el valor central. Es decir:

$$Me = x_{n+1/2}$$

Si n es par, la mediana es la media aritmética de las dos observaciones centrales. Cuando n es par, los dos datos que están en el centro de la muestra ocupan las posiciones n / 2 y n / 2 + 1. Es decir:

$$Me_v = \frac{(x_{n/2} + x_{n+1/2})}{2}$$

así tenemos Me_x, Me_y, Me_z y Me_M para las componentes *x*, *y*, *z* y para el módulo de acelerometría respectivamente.

- *Desviación estándar* σ

Se define como la raíz cuadrada de la varianza. Junto con este valor, la desviación típica es una medida (cuadrática) que informa de la media de distancias que tienen los datos respecto de su media aritmética, expresada en las mismas unidades que la variable.

Para conocer con detalle un conjunto de datos, no basta con conocer las medidas de tendencia central, sino que necesitamos conocer también la desviación que representan los

datos en su distribución respecto de la media aritmética de dicha distribución, con objeto de tener una visión de los mismos más acorde con la realidad a la hora de describirlos e interpretarlos para la toma de decisiones. La varianza viene dada por la siguiente igualdad:

$$S_v = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^2}$$

se calculan de esta forma S_x, S_y, S_z y S_M para las componentes *x*, *y*, *z* y para el módulo de acelerometría respectivamente.

- *Desviación media*

La desviación media es la media de las diferencias en valor absoluto de los valores a la media y viene dada por la siguiente expresión:

$$D_v^m = \frac{1}{N}\sum_{i=1}^{n}|a_i - \bar{a}|$$

así se calculan D_x^m, D_y^m, D_z^m y D_M^m para las componentes *x*, *y*, *z* y para el módulo.

- *Rango intercuartílico*

Se denomina rango intercuartílico, a la diferencia entre el tercer y el primer cuartil de la distribución de datos. Es una medida de la dispersión de la muestra que dará una franja en la que se encuentra el 50% de la población.

El rango intercuartílico (*iqr*) viene dado por la siguiente expresión:

$$iqr_v = Q_3 - Q_1$$

de esta forma se calcularán iqr_x, iqr_y, iqr_z e iqr_M.

- *Media geométrica*

Se define la media geométrica de una distribución compuesta por los *n* valores de acelerometría $\{a_1, a_2, \dots, a_n\}$ correspondientes a la ventana temporal *t* como:

$$G_v = \sqrt[n]{\prod_{i=1}^{n} a_i}$$

siendo G_x, G_y, G_z y G_M las medias geométricas correspondientes a las distribuciones de valores para los ejes *x*, *y*, *z* de acelerometría, así como para el módulo de la misma.

Este estadístico, a diferencia de la media aritmética se ve muy influenciado por valores cercanos a cero de la distribución. Se debe tener en cuenta que los valores igual a cero deben ser filtrados previamente para dotar de cierto grado de información al valor de este estadístico.

- *Amplitud máxima frecuencial*

Se denomina amplitud máxima frecuencial a la amplitud máxima de los datos de acelerometría representados en el dominio frecuencial. Esta variable así como el resto que se exponen a continuación será presentada en detalle en la sección titulada *Dominio temporal de la señal de* acelerometría.

- *Amplitud mínima frecuencial*

Se denomina amplitud mínima frecuencial a la amplitud mínima de los datos de acelerometría representados en el dominio frecuencial.

- *Frecuencia de la amplitud máxima*

Se denomina frecuencia de la amplitud máxima a aquella frecuencia en la que se produce la amplitud máxima de los datos acelerométricos. Este dato, como se explicará en la siguiente sección será útil para obtener la velocidad o la frecuencia a la que se realiza la actividad física. Un sencillo ejemplo podría ser el cálculo de los pasos por minutos o las pedaladas por minuto que realiza un determinado usuario de la plataforma de reconocimiento de actividades.

- *Frecuencia de la amplitud mínima*

Se denomina frecuencia de la amplitud mínima a aquella frecuencia en la que se produce la amplitud mínima de los datos acelerométricos.

DOMINIO TEMPORAL DE LA SEÑAL DE ACELEROMETRÍA

El físico francés, Joseph Fourier (1768-1830), desarrolló una representación de funciones basada en la frecuencia, que ha tenido una gran importancia en numerosos campos de las matemáticas y ciencias actuales.

Mediante la transformada rápida de Fourier (FFT) podemos trabajar sobre el dominio frecuencial a partir de los datos de acelerometría de una ventana temporal. Gracias a este proceso podemos observar las frecuencias en lugar de los valores del módulo de acelerometría, dando un nuevo punto de vista sobre la actividad realizada y posibilitando la generación de nuevas variables para el reconocimiento de la misma. Además de esto, gracias al dominio temporal seremos capaces de concretar ciertas características sobre un patrón de actividades genérico, por ejemplo la discriminación entre andar rápido o andar lento, siendo andar el patrón general de actividad. *Fs* será la frecuencia de muestreo (en Hz) de la señal original. Es muy importante ajustar adecuadamente este valor, ya que dependerá del dispositivo y de la configuración concreta del sensor de acelerometría a partir del cual se obtienen los datos.

Además de las limitaciones propias de la velocidad de muestro del dispositivo, en los sensores de acelerometría integrados en los dispositivos móviles, la frecuencia de muestreo puede no ser constante.

Esto influirá por tanto a la hora de calcular los valores derivados del dominio temporal, por lo que *Fs* deberá variar en función de la frecuencia de muestro media obtenida en cada ventana temporal concreta. Con este procedimiento será posible reducir al máximo el error de ajuste.

Antes de proceder con el cálculo de la FFT, es necesario eliminar la componente continua (DC) de la señal original con el fin de evitar irregularidades en la frecuencia 0Hz de la FFT, que corresponde a la intensidad de señal continua. Para ello eliminamos la componente DC mediante la resta de la señal original correspondiente a la ventana temporal y la media de las intensidades de la misma. De esta forma, cada dato acelerométrico a_i de la ventana temporal actual se transformará en otro valor (δ_i) resultante de eliminar la componente de continua:

$$\delta_i = a_i - \overline{a}$$

Posteriormente generamos los valores correspondientes al eje de abscisas, es decir, los valores temporales concretos correspondientes a las diferentes frecuencias identificadas en la señal. Para ello comenzaremos desde el segundo 1/*Fs*, es decir, desde el tiempo en el que fue leído el primer dato hasta llegar al segundo correspondiente al último dato leído.

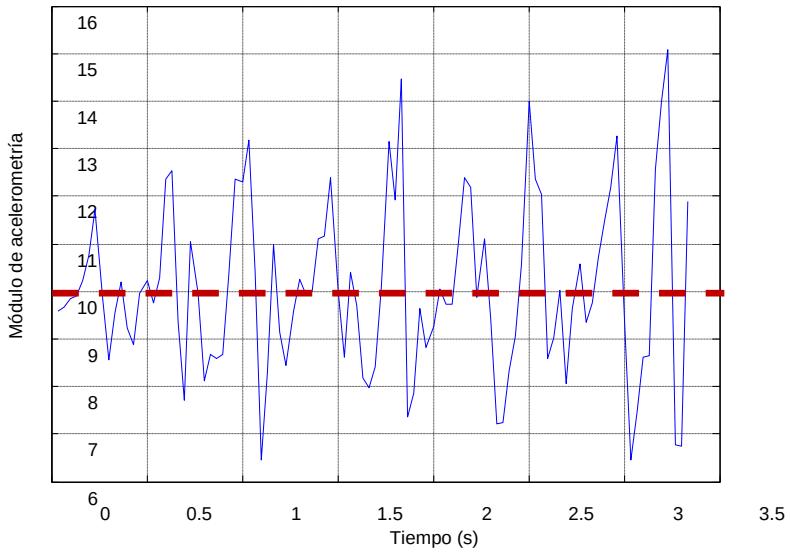


Figura 39: Módulo de acelerometría con componente continua

Una vez eliminada la componente continua de la señal, el resultado se muestra en la Figura 40. Como se puede observar, la única diferencia en el dominio temporal es el nivel en torno al cual se suceden los datos. Mientras que en el caso anterior el módulo medio se situaba en torno a 10 (1g), en este caso el valor baja a 0.

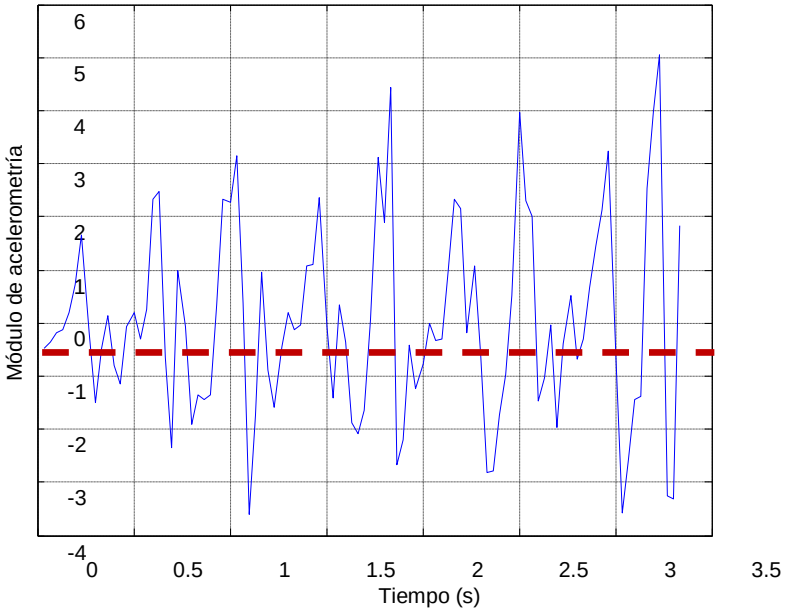


Figura 40: Módulo de acelerometría sin componentes de continua

Debemos tener en cuenta que siempre es necesario trabajar con la mitad de elementos de la FFT, ya que la frecuencia de muestreo debe ser al menos, el doble de la máxima frecuencia a muestrear según el teorema de Nyquist. De esta forma, durante el proceso de reconocimiento se trabajará con una *Fs* en torno a 30Hz, por lo que la frecuencia máxima que podrá ser medida con exactitud en la FFT será de 15Hz.

Llegados a este punto se puede ver claramente qué influencia tiene la componente continua en la transformada de Fourier para la ventana temporal que se está estudiando.

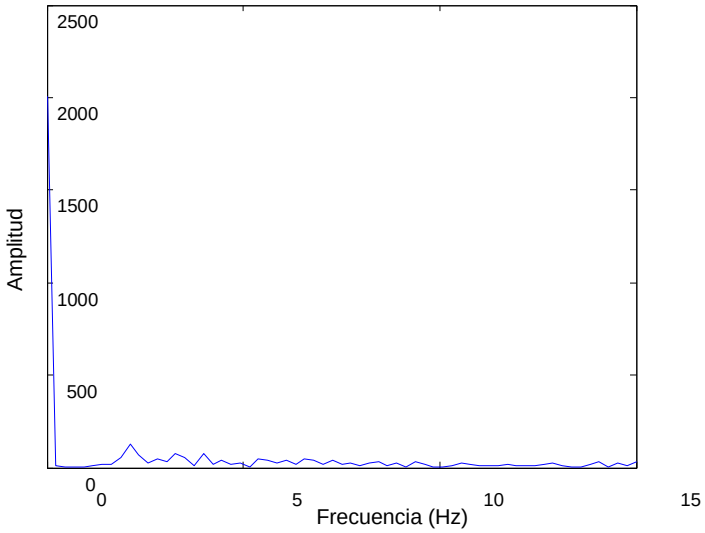


Figura 41: FFT con componente de continua

En la Figura 41 se puede observar claramente que la amplitud de la componente cuya frecuencia está próxima a 0 Hz es muy elevada. Esto se debe a que la componente continua está presente, por lo que la primera componente de la FFT es la propia DC. Sin embargo, en la Figura 42 siguiente

podemos ver como eliminando la DC de la señal original, la irregularidad del extremo en la frecuencia de 0Hz desaparece, dando como resultado una señal mucho más clara y sin irregularidades.

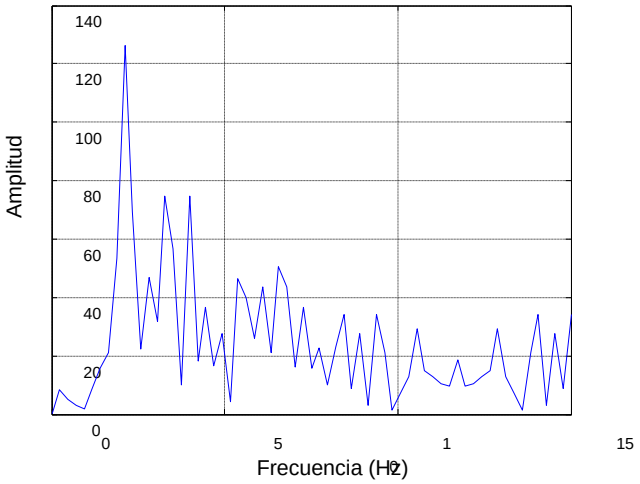


Figura 42: FFT sin componente de continua

Este proceso será realizado para cada una de las ventanas temporales, dando como resultado una señal a partir de la cual será posible obtener una serie de variables que dotarán al sistema de información adicional a la proporcionada por las variables estadísticas.

En primer lugar, mediante el cálculo de máximos de los valores de las señal en el dominio frecuencial proporcionados por la FFT para diversas actividades, podríamos obtener la frecuencia de realización de la actividad. Por ejemplo, los picos de la FFT correspondiente a la actividad caminar o correr, se correspondería con la frecuencia (Hz) con que el usuario está realizando los pasos o las zancadas. Para otras actividades como ir en bicicleta, la frecuencia se asociaría a las pedaladas que el usuario está realizando por segundo.

Si aplicamos lo anterior la ventana temporal de muestra con la que estamos trabajando, podemos ver en la Figura 43 (línea en trazo discontinuo) que representa la señal en el dominio frecuencial, que se produce un claro máximo en torno a los 2Hz.

Con este fin se ha desarrollado una función que permitirá calcular el máximo frecuencial sobre una determinada ventana temporal mediante el uso de un algoritmo basado en la técnica *Divide y Vencerás* con la que obtendremos de manera precisa el valor frecuencial máximo dentro del vector generado por la DFT. Esta técnica, además del resto de operaciones que serán necesarias realizar sobre el sistema, deberán ser implementadas de una forma eficiente, debido a que su aplicación será continua para cada una de las ventanas temporales, por lo que podrían producirse riesgos de bloqueos en el servidor o en el propio dispositivo móvil si no fuera así.

Paralelamente al dato anterior, también se calculará la frecuencia asociada a la amplitud máxima. De esta forma también será posible reconocer cierto tipo de actividades basándonos en este valor. Para obtener dicha frecuencia debemos hacer uso de la función de frecuencia de espaciamiento *fs*. Esta es una función lineal estrictamente creciente a partir de la cual podemos obtener la frecuencia asociada a cada índice del vector de datos de componentes frecuenciales. De esta forma, *fs(ind)* será la frecuencia para la cual se obtiene una mayor amplitud en la componente frecuencial de la señal original.

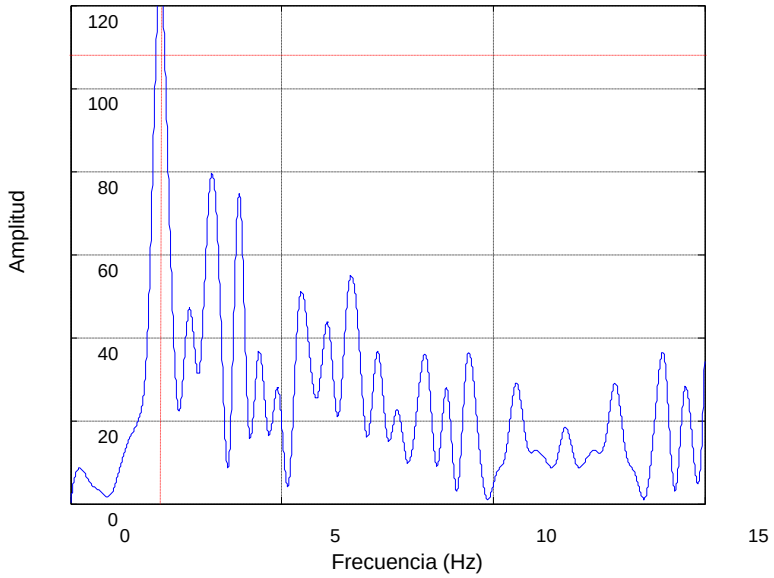


Figura 43: Máximos de una ventana temporal de datos de acelerometría en el dominio temporal

A continuación se expone un ejemplo sobre el uso de la componente frecuencial para la determinación de una determinada actividad o su frecuencia de realización. En primer lugar calcularemos la frecuencia máxima y el índice para la ventana temporal que estamos usando, lo cual da como resultado:

- *Amplitud máxima: 123.2057Hz*
- *Índice del máximo de amplitud = 10;*

Así sabemos que la amplitud máxima de la componente frecuencial de la señal cuyo valor es 123 Hz ocupa el décimo lugar en la función de espaciamiento. Mediante dicha función de espaciamiento será posible calcular la frecuencia en la que se produce la amplitud máxima.

$$fsMax = f(10) = 2.1094$$

De esta forma, el valor calculado anteriormente indica que el usuario está realizando una actividad cuya amplitud máxima se encuentra en torno a 2.1 Hz. Gracias al uso de otras variables y mediante los algoritmos de aprendizaje y clasificación que se verán a lo largo de este documento, podremos saber de qué actividad se trata. En este caso supondremos que la actividad que estaba realizado el usuario es 'Andando', por lo que puede ser inferido a partir de los datos anteriores que el usuario se está desplazando a una frecuencia de 2.1 pasos por segundo, o lo que es lo mismo, a un paso cada 0.47 segundos. Por lo tanto, a partir de estos valores no sólo será posible determinar la actividad que se está realizando sino la intensidad.

Derivado de esta técnica, posteriormente se desarrollará un sistema capaz de determinar el consumo calórico de un individuo a través del dispositivo móvil.

En esta sección se justificará la necesidad de realizar una identificación de variables relevantes en el problema del reconocimiento de actividades para posteriormente, eliminar aquellas variables que aporten menos cantidad de información al sistema. Con ello conseguiremos reducir la complejidad del problema tanto desde el punto de vista temporal como del consumo de recursos del dispositivo.

NECESIDAD DE REDUCCIÓN DE VARIABLES

Para llevar a cabo el sistema de reconocimiento de actividades, se han seleccionado un total de 12 variables en el caso de trabajar con el módulo de los datos de acelerometría y 48 variables si se opta por trabajar con las componentes X, Y, Z de la aceleración independientemente. Como vemos, se trata de un elevado número de variables que el sistema deberá procesar, clasificar y reconocer posteriormente. De esta forma los principales inconvenientes que aparecen en el proceso de reconocimiento de actividades a la hora de trabajar con todo el abanico de variables son los siguientes:

- Coste computacional derivado del cálculo de todos los estadísticos por parte del dispositivo móvil
- Tiempo de entrenamiento del sistema en caso de emplear entrenamiento basado en redes neuronales
- Posibilidad de tratamiento de información redundante en varias variables del sistema, lo que llevaría a un modelo poco confiable
- Espacio necesario para almacenar la información correspondiente al proceso de aprendizaje para todas las variables involucradas en el sistema de clasificación

En ocasiones las variables manejadas por el sistema contienen la misma información o existen relaciones entre ellas. Las variables correlacionadas pueden afectar negativamente al rendimiento del sistema, por lo que es necesario eliminarlas. Existen métodos estadísticos que aportan soluciones a este problema mediante diferentes técnicas de transformación de variables originales del sistema a nuevos subconjuntos reducidos de variables, de manera que la independencia entre ellas es total y cada una aporta una mayor cantidad de información al sistema.

En general los métodos de eliminación de variables relacionadas persiguen los siguientes objetivos (Stratified/PCA: un método de preprocesamiento de datos y variables para la construcción de modelos de redes neuronales, 2003):

- La contribución de la varianza debe ser alta en el conjunto de las nuevas variables
- La información redundante entre variables altamente correlacionadas pueden ser eliminadas sin riesgo a perder información necesaria
- Menos variables independientes hacen más fácil el análisis de los resultados y simplifica la construcción de los modelos de actividad

Concretamente, en este apartado se expondrán dos métodos destinados a reducir el número de variables usadas en el sistema. En primer lugar será presentado el método de reducción basado en el análisis de componentes principales (PCA) y posteriormente un nuevo método basado en el coeficiente de correlación de Pearson que permite detectar y eliminar variables altamente correlacionadas de una forma poco costosa computacionalmente. Precisamente esta característica lo hace especialmente interesante para el procesamiento en dispositivos móviles. Sin embargo, como será

estudiado más adelante, este reducido coste computacional provoca que la eficacia del método sea menor en comparación con el ACP.

ANÁLISIS DE COMPONENTES PRINCIPALES

El análisis de componentes principales, es una técnica estadística propuesta a principios del siglo pasado por Karl Pearson como parte del análisis de factores. Sin embargo la complejidad de los cálculos retrasó su desarrollo hasta la aparición de los computadores y fue ampliamente utilizado durante toda la segunda mitad del siglo XX. El relativamente reciente florecimiento de los métodos basados en componentes principales hace esta técnica sea muy empleada por una gran cantidad de sistemas reales.

Podría decirse que el objetivo principal que persigue el ACP es la representación de las medidas numéricas de varias variables en un espacio de menor dimensionalidad, donde el estudio a simple vista o mediante métodos más elaborados sea capaz de percibir relaciones entre las diferentes variables que de otra manera permanecerían ocultas en dimensiones superiores. Dicha representación debe asegurar que la pérdida de información debido a la elección de ciertas combinaciones lineales de las diferentes características o variables del sistema sea progresiva. De esta manera será posible elegir la pérdida de información que el sistema está dispuesto a asumir en pos de la simplificación del problema y la reducción de la dimensionalidad.

Un símil que podría ilustrar el funcionamiento del ACP sería el siguiente: imaginemos una gran porción de chapa de forma rectangular (objeto de tres dimensiones) de 4m de larga, 1m de ancha y 2 cm de espesor. Para efectos prácticos, dicha lámina puede ser considerara como un objeto plano (de dos dimensiones) de 4m de largo por 1m de ancho. Al realizar esta abstracción, en realidad se está reduciendo la dimensionalidad del problema, haciéndolo más sencillo de comprender (en este caso de dibujar, por ejemplo). Sin embargo, se pierde cierta cantidad de información ya que, por ejemplo, puntos opuestos situados en las dos caras de la lámina aparecerán confundidos en uno solo en el modelo bidimensional obtenido, debido a la pérdida de la tercera dimensión. En general, en el símil de la chapa, se pierden todas las distancias perpendiculares a las caras. Sin embargo, la pérdida de información se ve ampliamente compensada con la simplificación realizada, ya que muchas relaciones, como la vecindad entre puntos, la distancia o el cálculo de la superficie, son más evidentes cuando el sistema es dibujado sobre un plano en lugar de realizar la representación mediante una figura tridimensional que necesariamente debe ser dibujada en perspectiva.

El ACP permite reducir la dimensionalidad de los datos, transformando el conjunto de p variables originales en otro conjunto de q variables no correlacionadas ($q \leq p$) llamadas componentes principales. Las p variables son medidas sobre cada uno de los n individuos, obteniéndose una matriz de datos de orden np ($p < n$).

A la hora de realizar el ACP existen dos alternativas que pueden ser empleadas para el cálculo de las distintas componentes que describirán el sistema posteriormente. La primera opción es el uso de la matriz de correlaciones. En ella se dará la misma importancia a todas y a cada una de las variables, de manera que a priori las cantidades de información aportadas por cada una de ellas se presuponen idénticas; esto puede ser conveniente si consideramos que todas las variables son igualmente relevantes. La segunda opción es el uso de la matriz de covarianzas como método para la obtención de la cantidad de información asociada a cada variable. De esta forma, este método podrá ser utilizado siempre y cuando todas las variables tengan las mismas unidades de medida y además, cuando sea conveniente destacar cada una de las variables en función de su grado de variabilidad.

Las q nuevas variables (componentes principales) son obtenidas como resultado de un conjunto de combinaciones lineales de las variables originales. De esta forma, dichas componentes serán ordenas en función del porcentaje de varianza explicada, teniendo en primer lugar aquellas componentes que explicar o contienen la mayor información, dejando en las últimas posiciones aquellas combinaciones que menos información aporten al sistema. En este sentido, el primer componente será el más importante por ser el que explica un mayor porcentaje de la varianza de los datos.

Para que pueda ser aplicado un análisis de componentes principales en el problema del reconocimiento de actividades, se debe cumplir que todas las variables del sistema deben ser continuas y que el número de elementos observados debe ser mayor que el número de variables originales. Ambos requisitos se cumplen en nuestro caso, por lo que el ACP puede ser aplicado sin ningún problema.

El primer paso para calcular las componentes principales del sistema de reconocimiento de actividades es generar una matriz en la cual las filas representan las distintas variables originales del sistema, mientras que las columnas representan las ventanas temporales obtenidas durante el proceso de aprendizaje. Posteriormente calcularemos la matriz de covarianza de dicha matriz, es decir, la posición [i,j] de la nueva matriz representará el valor de la covarianza entre la variable i y la variable j.

Seguidamente calcularemos los valores y vectores propios de la matriz. Los vectores propios, autovectores o eigenvectores de un operador lineal son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección. Este escalar λ recibe el nombre valor propio, autovalor, valor característico o eigenvalor. A menudo, una transformación queda completamente determinada por sus vectores propios y valores propios. Un espacio propio, autoespacio o eigenespacio es el conjunto de vectores propios con un valor propio común.

Los vectores pueden visualizarse como flechas de una cierta longitud apuntando en una dirección y sentido determinados.

- Los vectores propios de las transformaciones lineales son vectores que, o no se ven afectados por la transformación o se ven multiplicados por un escalar, y por tanto no varían su dirección.
- El valor propio de un vector propio es el factor de escala por el que ha sido multiplicado.

Por ejemplo, un vector propio de una rotación en tres dimensiones es un vector situado en el eje de rotación sobre el cual se realiza la rotación, y el valor propio correspondiente es 1. Formalmente, se definen los vectores propios y valores propios de la siguiente manera: Si **A**: $V \rightarrow V$ es un operador lineal en un cierto espacio vectorial V, v es un vector diferente de cero en V y c es un escalar tales que

$$Av = cv,$$

entonces decimos que v es un vector propio del operador A, y su valor propio asociado es c. Observe que si v es un vector propio con el valor propio c entonces para cualquier v es también un vector propio con el valor propio c. De hecho, todos los vectores propios con el valor propio asociado c junto con **0**, forman un subespacio de V, el **espacio propio** para el valor propio c.

Para realizar la elección de las componentes principales durante el proceso de experimentación se parte de un conjunto de 180 ventanas temporales que pueden ser clasificadas en cinco clases distintas

en función de la actividad que realizara el usuario en el instante de la captura de datos. Dichas cinco clases son parado, sin dispositivo, andando, saltando y bicicleta. Cabe destacar que el método de ACP no es un método de clasificación, por lo que el valor de la clase asociado a cada elemento del conjunto de pruebas es totalmente indiferente. Cada uno de los conjuntos de prueba está formado por 48 variables distintas en el caso del sistema de componentes acelerométricas independientes y 12 en el caso del sistema modular.

En primer lugar comenzaremos aplicando el análisis de componentes principales a un conjunto de datos tridimensional, es decir, contiene lectura de estadísticos para las componentes X, Y y Z de manera independiente para cada una de las ventanas temporales. Para permitir observar los beneficios y el proceso realizado mediante el ACP, se ha reducido el conjunto de entrenamiento para que sólo sea posible reconocer cuatro actividades.

Evidentemente es imposible obtener una representación gráfica para un problema de 48 variables, así que comenzaremos observando la reducción de la dimensionalidad del problema del reconocimiento de actividades a dos dimensiones. Dichas dimensiones se corresponden con los autovectores asociados a la combinación lineal de las variables originales que poseen una mayor representación de la varianza de la muestra, dicho en otras palabras, aquellas componentes que poseen mayor información. La Figura 44 representa la posición de los valores del conjunto de test en base a las dimensiones determinadas por las dos primeras componentes principales. En ella se puede observar cómo los datos (representados por puntos rojos) se agrupan formando una serie de conjuntos o clusters asociados a cada una de las actividades que se pretenden reconocer, representados en la figura por cuadros de distinto tamaño. Cuanto mayor sea la discriminación o la distancia entre los centros de dichos clusters, mayor será la varianza recogida por las componentes y en consecuencia, mejores resultados se conseguirán a la hora de realizar la clasificación.

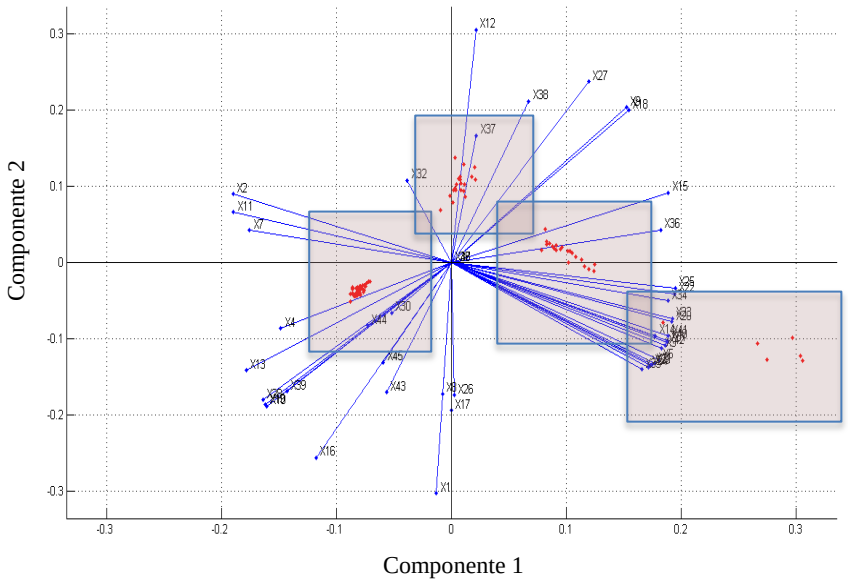


Figura 44: ACP con acelerometría tridimensional y cuatro actividades de entrenamiento. Selección de 2 componentes

En la misma figura podemos observar una serie de trazos azules que se corresponden con la recta que une el origen de los ejes de referencias con cada uno de los autovectores generados por la aplicación del método de análisis de componentes principales sobre los datos y las variables originales. Estos autovectores proporcionan un sistema para proyectar cada uno de los datos originales en el punto adecuado en base al nuevo cambio de ejes realizado. Por último cada una de las 415

observaciones de ventanas temporales del conjunto de prueba está representada por un punto, y su ubicación indica la puntuación de cada observación a partir de las dos componentes principales mostradas. Por ejemplo, los puntos cerca del borde izquierdo de esta gráfica tiene la puntuación más baja para el primer componente principal.

Sin embargo, a pesar de que es posible reconocer cada uno de los conjuntos formados por los datos asociados a una determinada actividad, así como diferenciarlo del resto de conjuntos, aún existen determinadas zonas en las que se produce un solape de la información, indicado en la figura como aquellas áreas en las que se superponen dos rectángulos. Esto se traduce en una incógnita a la hora de realizar el reconocimiento, por lo que deberá ser resuelta con el fin de poder aislar de una manera más eficaz las diferentes clases del sistema. Para ello, será preciso dotar al sistema de más información a partir de la inclusión de una nueva componente para representar al sistema. Esta nueva componente proveerá de una mayor cobertura de la varianza, añadiendo nueva información a las dos componentes usadas en el primer caso. En cambio, podría suceder que incluso incrementando el número de dimensiones del problema, sea imposible determinar o aislar los clusters asociados a dos actividades diferentes. En tal caso la falta de información no habría sido provocada por la reducción del número de variables, sino porque las variables tenidas en cuenta inicialmente no aportan suficiente información para discriminar ambas actividades. Este caso será tenido en cuenta posteriormente durante el proceso de generación de intervalos y discretización, mediante el uso de lógica difusa para el reconocimiento.

Como se ha comentado anteriormente, la evolución natural para buscar una mayor independencia entre las clases con el fin de obtener una mejor clasificación mediante los algoritmos que serán propuestos en la siguiente sección, es introducir una nueva componente que enriquezca la información proporcionada al sistema.

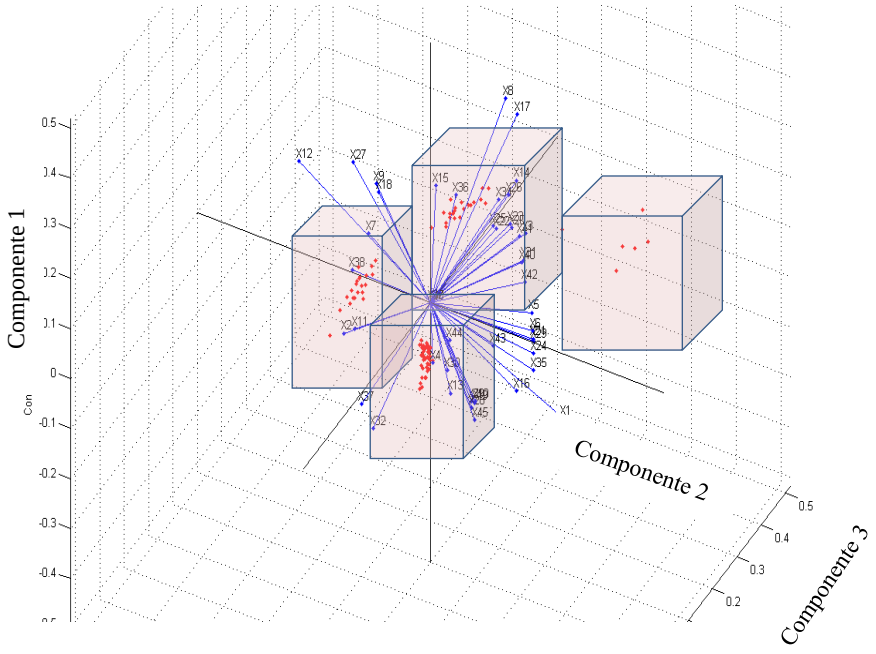


Figura 45: ACP con acelerometría tridimensional y cuatro actividades de entrenamiento. Selección de 3 componentes

En esta línea, la Figura 45 presenta la distribución de los datos de la batería de aprendizaje en un espacio tridimensional, en el que cada una de las dimensiones se corresponde con la información aportada por cada de las una componentes resultantes del análisis. Como se puede observar en dicha

figura, los datos presentan una mayor independencia, gracias a que los clusters gozan de un grado más de libertad con la inclusión de la nueva dimensión.

Este aumento de la dimensionalidad del problema y por consiguiente de la información contenida favorecerá la clasificación realizada posteriormente, aunque la gran ventaja es que con tan sólo tres dimensiones es posible representar con un cierto grado de independencia las diferentes clases de clasificación. Evidentemente esta reducción de 48 variable a tan sólo 3 conllevará una disminución de las necesidades computacionales a la hora de realizar el reconocimiento, hecho que justifica sobradamente el tiempo necesario para llevar a cabo el análisis de componentes principales en la fase de entrenamiento.

Paralelamente al estudio llevado a cabo para el análisis de componentes principales en un entorno de acelerometría tridimensional, en la Figura 46 así como en la Figura 47 se puede observar el resultado para el mismo conjunto de datos de acelerometría, con la diferencia de que en este caso se trabaja con el módulo. En este caso el número de variables iniciales eran doce, por lo que en principio la capacidad global de información del sistema para identificar las distintas actividades sería menor que en el caso anterior. A pesar de ello, bajo determinadas circunstancias podría ser útil este método debido a que la cantidad información que debería enviarse a través de internet, en caso de que el reconocimiento tenga lugar en el servidor, es mucho menor. De igual forma, el número de estadísticos que es necesario calcular se reduce significativamente, pasando de 48 a 12.

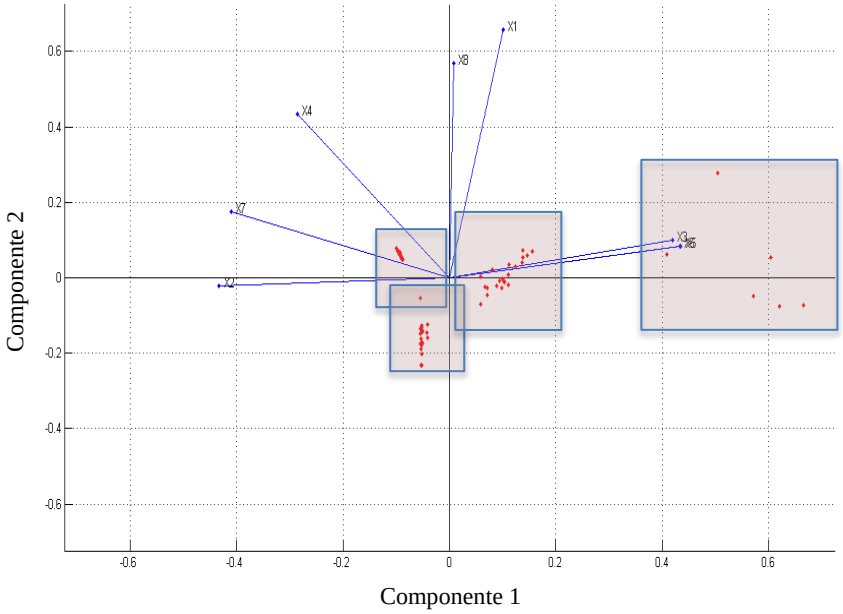


Figura 46: ACP con acelerometría modular y cuatro actividades de entrenamiento. Selección de 2 componentes

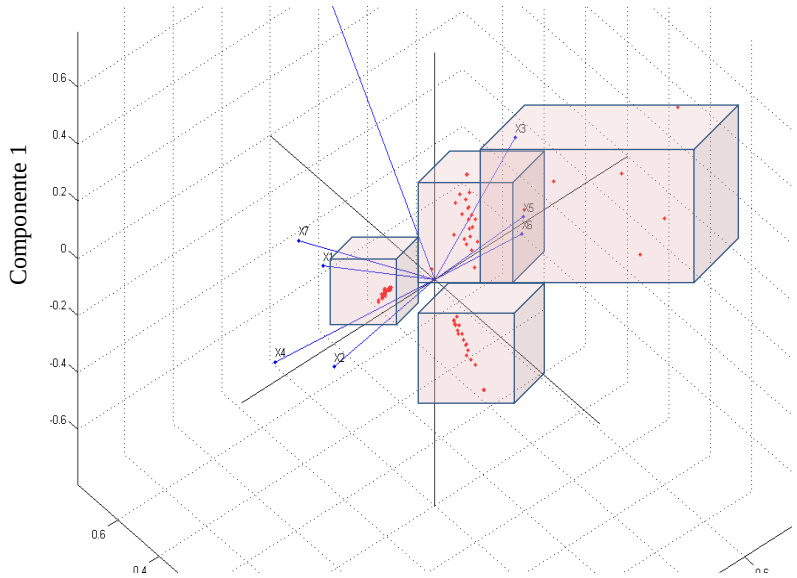


Figura 47: ACP con acelerometría modular y cuatro actividades de entrenamiento. Selección de 3 componentes

Debe ser tenido en cuenta que independientemente de que sea aplicado o no el análisis de componentes principales, es necesario calcular íntegramente el conjunto de estadísticos identificados, debido a que para la generación de los autovectores serán necesarios dichos valores. Por tanto, el principal problema del método de reducción de variables basado en ACP es que, a pesar de que el número de variables que identifican al sistema se ve reducido de manera drástica mediante la creación de nuevas variables, para calcular el valor de las nuevas variables debemos hacer uso de todas las variables iniciales identificadas, puesto que el nuevo conjunto será una combinación lineal de todas las variables iniciales.

Para visualizar la información proporcionada al sistema por la adición de una nueva componente principal al conjunto de componentes que representan el sistema, se incluyen la Figura 48 y en la Figura 49. En ellas se puede observar el porcentaje de varianza total cubierta a medida que se incrementan el número de componentes que describen el sistema. Además, bajo cada una de las figuras se incluyen sendas tablas que indicar numéricamente el porcentaje total de la varianza cubierta pada cada uno de los dos conjuntos de datos.

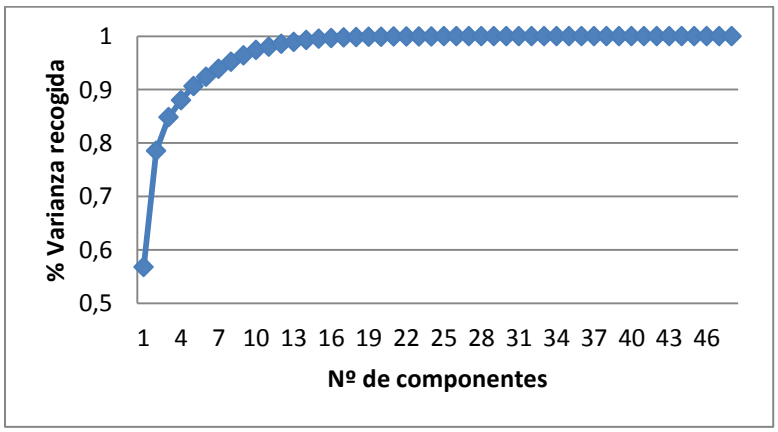


Figura 48: Porcentaje de varianza recogida para las componentes en acelerometría tri-axial

Nº Componentes	1	2	3	4	5	6	7	8	9	10	11	12	13
Varianza	0,57	0,79	0,85	0,88	0,91	0,92	0,94	0,95	0,96	0,97	0,98	0,99	0,99

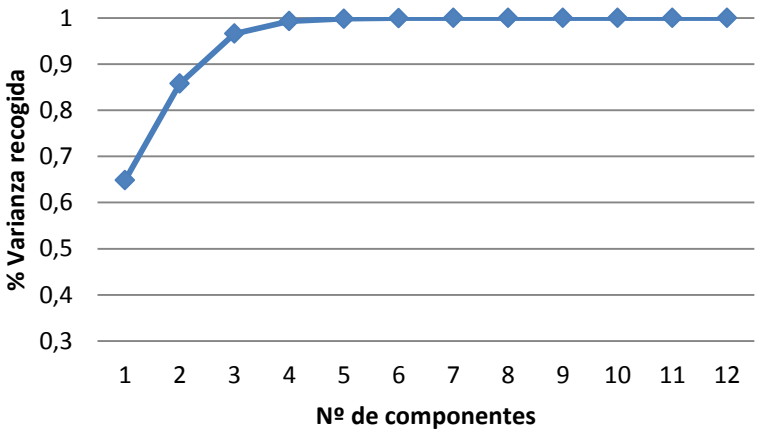


Figura 49: Porcentaje de varianza recogida para las componentes en acelerometría modular

Nº Componentes	1	2	3	4	5	6	7	8	9	10	11	12
Varianza	0,65	0,86	0,97	0,99	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

El resultado del análisis es el siguiente: para la acelerometría modular, empleando tan sólo 5 componentes el sistema es capaz de cubrir el 100% de la varianza del problema, mientras que este número asciende a 19 variables para el caso del método triaxial. Es decir, de 12 y 48 variables podríamos seleccionar tan sólo 5 y 19 respectivamente sin perder ninguna cantidad de información.

En un entorno real de aplicación del método propuesto, el número de actividades capaces de ser reconocidas por el sistema de clasificación es mayor que cuatro, por lo que el análisis realizado en esta sección es meramente ilustrativo y por tanto, durante cada uno de los procesos de aprendizaje llevados a cabo por los usuarios de la plataforma, el número de componentes seleccionadas variará como consecuencia de la aplicación del ACP sobre los datos concretos de aprendizaje. De esta forma la selección de características singulares es un proceso dinámico, estrechamente ligado a las actividades aprendidas así como a la manera en que cada individuo lleva a cabo dichas actividades.

COEFICIENTE DE CORRELACIÓN PONDERADO

Si dos variables poseen un alto grado de correlación, obviamente el comportamiento de una será predecible a partir del de otra, por lo que eliminaremos esta redundancia mediante la eliminación de una de esas variables. En función de cada actividad concreta, la correlación entre los estadísticos variará, de manera que si dos variables están altamente correlacionadas para la actividad *Walk* podrían no estarlo para *Jump*. Debido a esto, se deberá estudiar la correlación entre dos variables en global, es decir, teniendo en cuenta todas las actividades, pero sin olvidar que es posible que un determinado estadístico sea representativo para una determinada actividad, mientras que para el resto posee un alto grado de correlación con otro. Esto provoca que la eliminación de un determinado estadístico altamente correlacionada para una actividad, influya negativamente en el reconocimiento de otra

actividad para la cual no existía ninguna correlación. El proceso de eliminación de variables en este caso, se hará enfrentando los estadísticos uno a uno, obteniendo su correlación para cada una de las actividades entrenadas, aplicar un coeficiente de amplificación a cada uno de los coeficientes de correlación de una variable para las distintas actividades y, por último, seleccionar aquellos coeficientes cuyo índice de correlación sea más elevado. El valor obtenido será una representación numérica de la correlación global de un estadístico para el conjunto de actividades.

Esta eliminación de variables irrelevantes se basará en un primer nivel en el cálculo del índice de correlación de Pearson. Este índice se trata de una medida de correlación entre dos o más variables continuas, como es el caso de los estadísticos que estamos empleando. El coeficiente de correlación de Pearson se define como el cociente entre la covarianza de X e Y, siendo X e Y las dos variables comparadas, y el producto de las deviaciones típicas de las dos variables, es decir:

$$r = \frac{\sigma_{XY}}{\sigma_X . \sigma_X}$$

El índice elegido varía entre -1 y 1, de forma que cuanto más próximo esté r al valor 0, la relación lineal entre los estadísticos será menor.

En las siguientes figuras podemos ver gráficamente la relación existente entre el estadístico *Media aritmética* y cada una de las variables que queremos estudiar en forma de diagrama de dispersión. En dichos diagramas se representará en el eje de abscisas los valores de las *medias aritméticas* obtenidas para un conjunto de ventanas temporales capturadas mientras el usuario realizaba la actividad *Walk*. Por otro lado, en el eje de ordenadas se mostrarán los valores de los distintos estadísticos presentes en el conjunto original de variables para cada una de las ventanas citadas anteriormente. Con ellos podremos ver claramente la relación existente entre cada una de las variables del sistema.

Además, en las figuras se puede observar la correlación existente entre la variable *Media Aritmética* y el resto de variables correspondientes sistema de acelerometría modular. En ellas se refleja claramente cómo determinadas variables presentan una correlación elevada con la media aritmética, aunque como veremos a continuación, esto puede enmascarar una mínima correlación si se estudia dicha correlación independientemente para cada una de las actividades reconocidas.

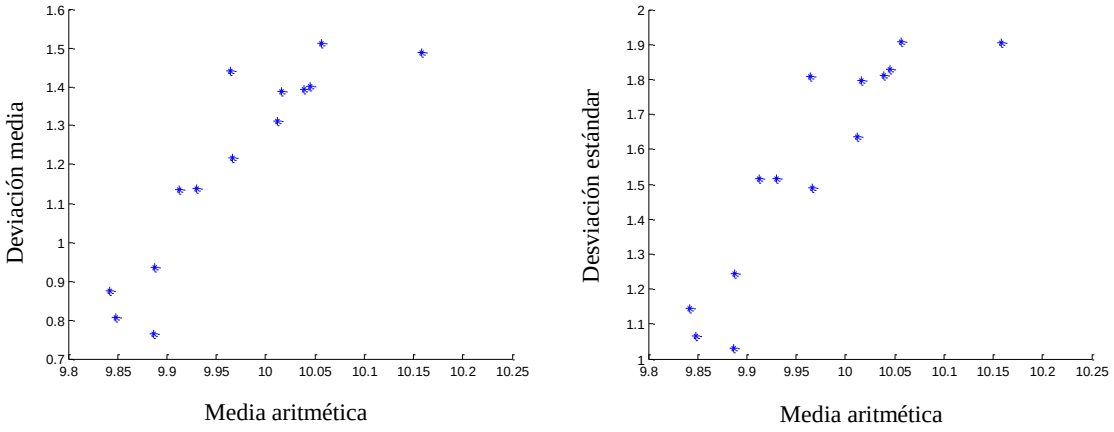


Figura 50: A) Dispersión de la media aritmética frente a la desviación media B) Dispersión de la media aritmética frente a la desviación estándar

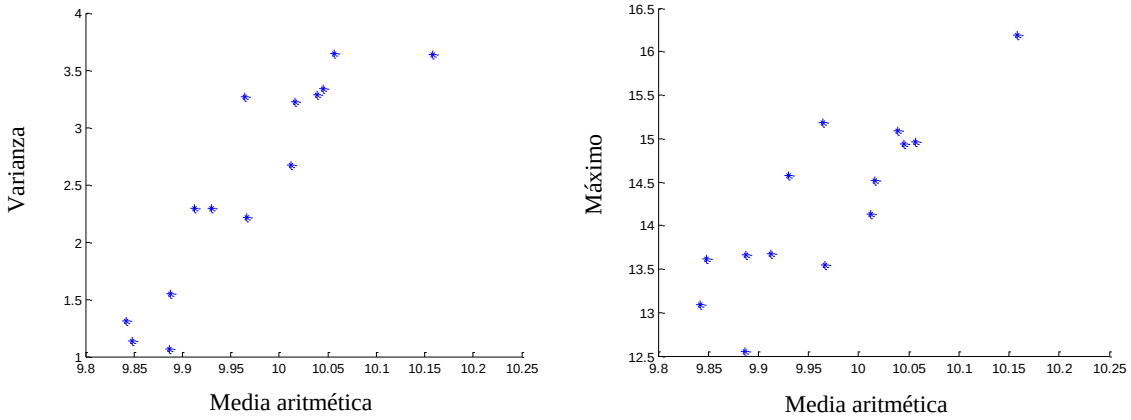


Figura 51: A) Dispersión de la media aritmética frente a la varianza B) Dispersión de la media aritmética frente al máximo

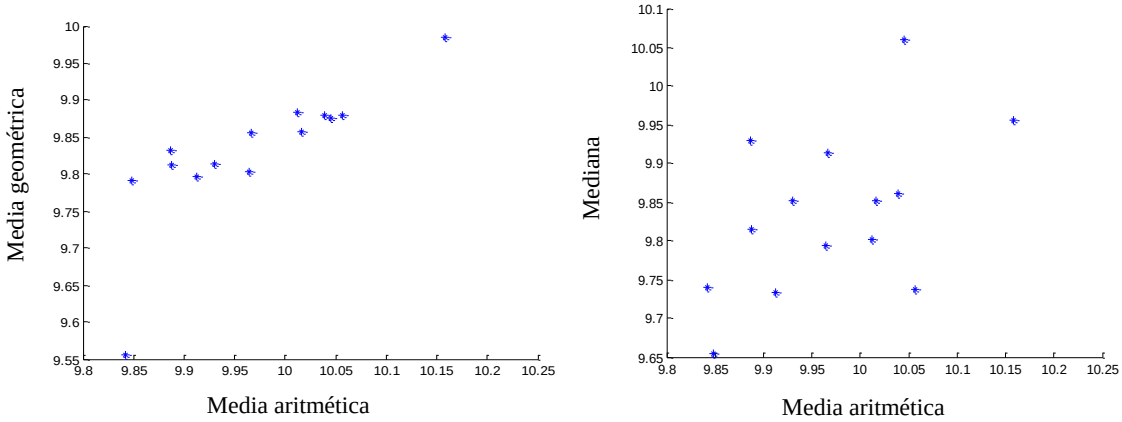


Figura 52: A) Dispersión de la media aritmética frente a la media geométrica B) Dispersión de la media aritmética frente a la mediana

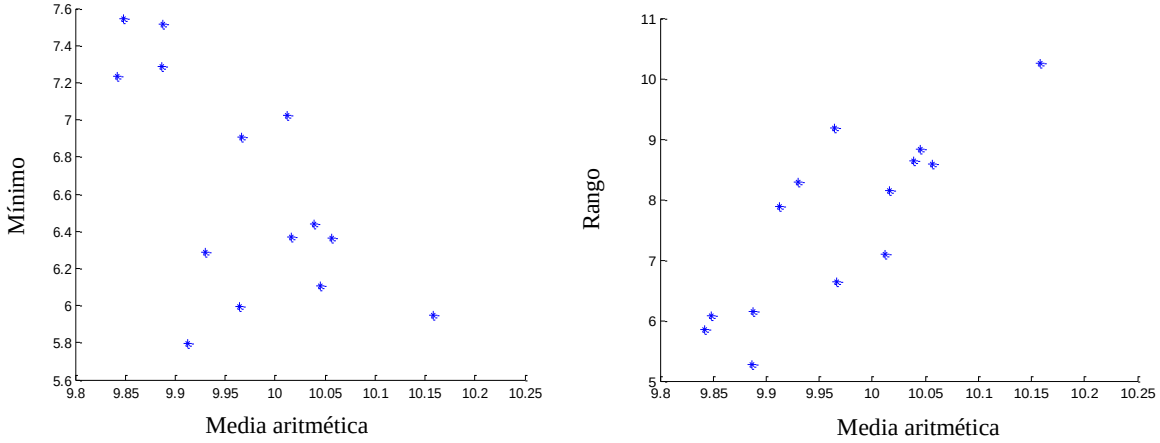


Figura 53: A) Dispersión de la media aritmética frente al mínimo B) Dispersión de la media aritmética frente al rango

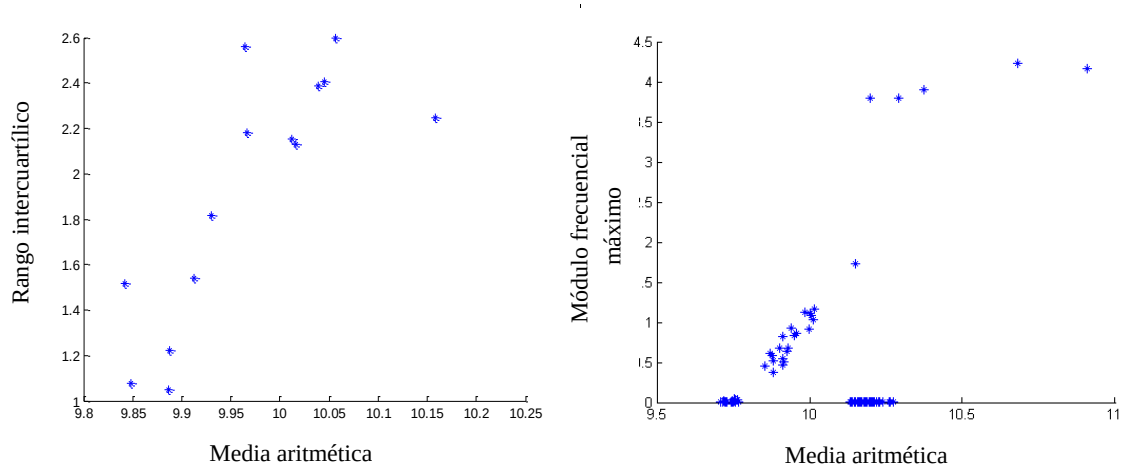


Figura 54: A) Dispersión de la media aritmética frente al rango intercuartílico B) Dispersión de la media aritmética frente al módulo frecuencial máximo

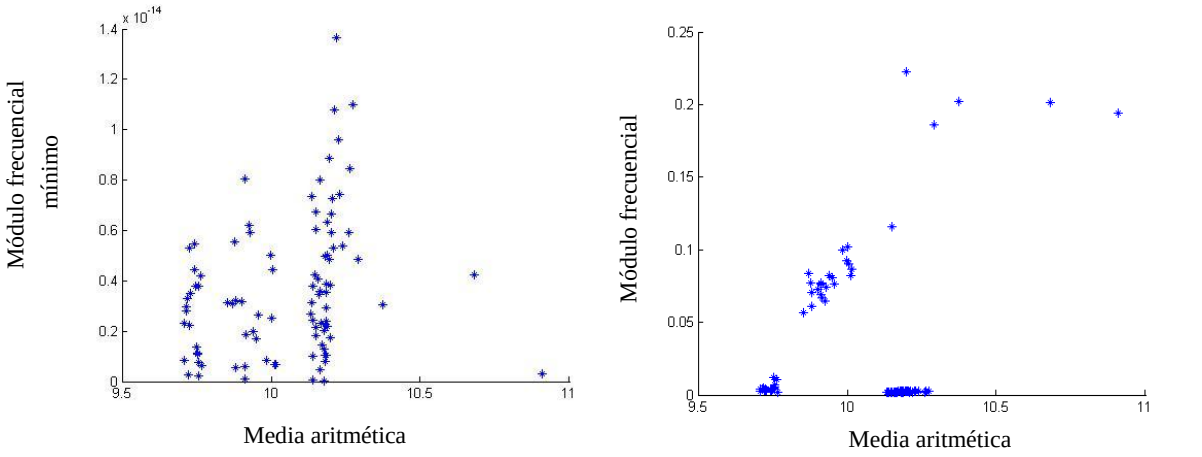


Figura 55: A) Dispersión de la media aritmética frente al módulo frecuencial mínimo B) Dispersión de la media aritmética frente a la mediana del módulo frecuencial

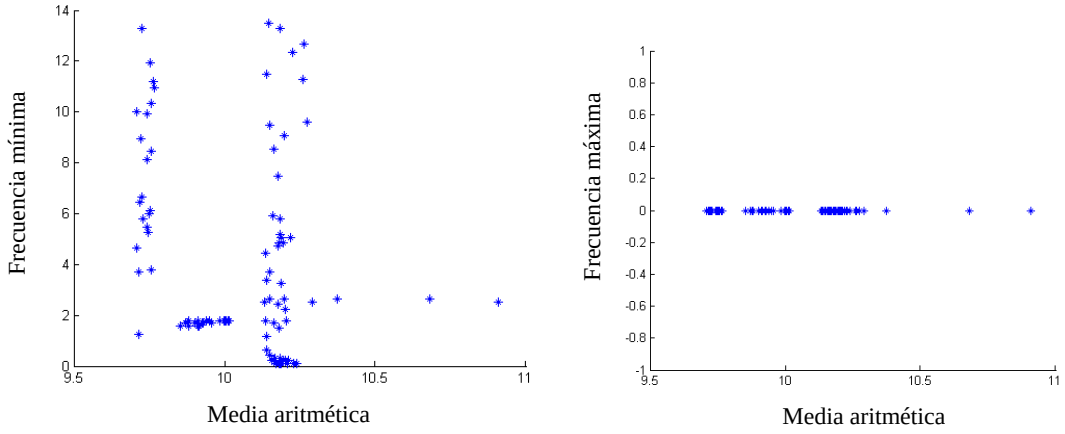


Figura 56: A) Dispersión de la media aritmética frente a la frecuencia mínima B) Dispersión de la media aritmética frente a la frecuencia máxima

Con el fin de determinar la correlación de una determinada variable respecto al conjunto restante de variables para cada una de las actividades reconocidas por el sistema, se introducirá el concepto de *coeficiente de correlación total de la actividad*. Este coeficiente reflejará la correlación existente entre cada par de variables para una determinada actividad, siendo 0 el valor que indica un aporte de información máximo (menos correlación) y 1 un aporte de información mínimo (mayor correlación).

Para obtener el coeficiente de correlación de la actividad (ρ_L), una primera aproximación sería sumar los índices de correlación para cada actividad y normalizar, es decir:

$$\rho_L = \frac{\sum_{i=0}^{n} |\rho_i|}{n},$$

donde ρ_i es el coeficiente de correlación del primer estadístico para la actividad i-ésima, y n es el número total de actividades que pueden ser reconocidas por el sistema. Como puede observarse, la ecuación anterior para el cálculo del coeficiente de correlación total de la actividad obtiene un resultado normalizado, es decir $\rho_L \in [0,1]$ por lo que puede ser comparado con un coeficiente de correlación total de la actividad correspondiente a otro estadístico ($\rho_L{}'$). Sin embargo, uno de los principales problemas de esta solución es que se recogen todos los índices de cada actividad de manera conjunta, por lo que si el estadístico *media aritmética* posee un alto grado de correlación con el estadístico *rango* en la actividad *Walk*, y un coeficiente muy bajo para la actividad *Run*, el coeficiente alto de correlación enmascarará al coeficiente menor, por lo que el resultado ρ_L podría llevarnos a la conclusión de que ambas variables están altamente correlacionadas y una de ellas podría ser eliminada. Pero nada más lejos de la realidad, ya que si al menos para una de las actividades el estadísticos "media aritmética" y "rango" poseen un bajo coeficiente de correlación, el ρ_L debería verse afectado, de manera que se "premiará" aquellos pares de estadísticos para los que en al menos una actividad su coeficiente de correlación sea bajo, mientras que será penalizado el hecho de que posean una alta correlación para todas las clases de la muestra.

Por otro lado, el principal problema que aparece en el coeficiente de correlación general de Pearson es la influencia entre las diversas variables involucradas en el sistema. Con el fin de eliminar la influencia del resto de variables, se aplicará el coeficiente de correlación total de cada actividad. En primer lugar, será necesario calcular el coeficiente de correlación parcial entre cada par de variables para una determinada actividad. Para ello emplearemos el concepto de regresión lineal múltiple para obtener los coeficientes de ajuste de los datos de las dos variables a comparar independientemente del resto de variables. De esta forma, definimos dos nuevas variables ${X_1}^*$ y ${X_2}^*$ de la siguiente forma:

$${X_1}^* = f(X_3, X_4, X_5 \dots X_N)$$

$${X_2}^* = g(X_3, X_4, X_5 \dots X_N)$$

donde $f(X_3, X_4, X_5 \dots X_N)$ y $g(X_3, X_4, X_5 \dots X_N)$ son la regresión lineal múltiple mínimo cuadrática que explica el comportamiento de X_1 y X_2 a partir de $X_3, \dots, X_N$ respectivamente.

De esta forma, con objeto de eliminar la influencia de todas las variables del sistema sobre las dos a estudiar, definimos los siguientes valores:

$$Z_1 = X_1 - X_1^*$$

$$Z_2 = X_2 - X_2^*$$

Tras realizar el proceso de eliminación de la influencia del resto de variables sobre las dos procesadas, es posible analizar la correlación entre ambas. Para ello se empleará el coeficiente de correlación de Pearson para el par de variables sobre cada actividad entrenada de manera independiente.

Con el fin de disminuir el coste computacional del procesamiento, el proceso de eliminación de influencias sobre cada par de variables será obtenido y almacenado globalmente a partir de todos los datos del conjunto de aprendizaje, para posteriormente realizar un filtrado de éste en base a cada una de las actividades.

De esta forma, se define el grado de relación lineal entre las variables X_i y X_j para la actividad L de la siguiente forma:

$$\rho_{ij}{}^{L} = \frac{\sigma_{Z_i{}^{L} Z_j{}^{L}}}{\sigma_{Z_i{}^{L}} . \sigma_{Z_j{}^{L}}}$$

donde $Z_i{}^{L}$ denotan a la variable Z_i pero tan sólo considerada la actividad L.

Como puede observarse, la ecuación anterior representa el coeficiente de correlación de Pearson sobre las variables X_i y X_j eliminando la influencia del resto de variables para los valores asociados a la actividad L. Una vez obtenido el coeficiente de correlación para el par de variables X_i y X_j, será necesario calcular el coeficiente de correlación total de la actividad para el estadístico i. Para ello se define el índice de correlación parcial de la variable i sobre la clase L de la siguiente forma:

$$\rho_i{}^{L} = \frac{1}{n} \sum_{j=1}^{n} |\rho_{ij}{}^{L}| \qquad 0 \leq \rho_i{}^{L} \leq 1$$

En último lugar, para obtener el factor de correlación general de la variable i aplicaremos un factor de penalización a cada coeficiente de correlación total de la actividad con el fin de atenuar los grados de correlación menores a un umbral, mientras que dicho coeficiente parcial será amplificado a partir de éste. Con este procesamiento será posible reducir la correlación general para aquellas variables que posean una correlación parcial reducida a la hora de identificar una determinada actividad.

Para llevar a cabo este método basado en el coeficiente de penalización haremos uso de un factor de peso de correlación para cada actividad L al que denominamos γ^L y se define de la siguiente forma:

$$\gamma_i{}^{L} = exp\left(-e^{(\alpha \cdot (\beta + 10 \cdot \rho_i{}^{L}))}\right)$$

La ecuación anterior se basa en una de las denominadas funciones Gompertz (Prediction model based on Gompertz function, 2009), la cual se aplica de manera habitual en modelos matemáticos para series temporales donde el crecimiento de los valores son más lentos al comienzo y al fin del periodo. Otra peculiaridad de esta función es que el crecimiento se produce más lentamente a la derecha que a la izquierda, a diferencia de lo que ocurre en las funciones logísticas. En el caso concreto del reconocimiento de actividades y del proceso de eliminación de variables, la función anterior permitirá ajustar el coeficiente de correlación parcial de la variable $\rho_i{}^L$ de manera que su crecimiento sea reducido para valores cercanos a 0, mientras que para valores a partir de un umbral el crecimiento será más acusado. Con esto es posible reducir el coeficiente original para valores inferiores al umbral de forma que su influencia total sea menor, evitando de esta forma el enmascaramiento de coeficientes de correlación elevados para una determinada actividad, con coeficientes de correlación menores para otras.

En la función anterior existen dos parámetros que nos permitirán ajustar el perfil de la función, α es el factor de amplificación y β el factor de ajuste. El parámetro α permite aumentar o disminuir la pendiente de la curva de ajuste, de manera que la transición entre los valores atenuados y los simplificados sea más o menos acusada. En cambio, el factor de ajuste β será el encargado de desplazar hacia la izquierda o hacia la derecha el área de transición generada.

Ahora debemos ajustar el factor de amplificación, que será el encargado de dotar a la función de ajuste de una forma adecuada, es decir, para valores pequeños del índice de correlación $\rho_i{}^L$ la función debe devolver valores muy pequeños de $\gamma_i{}^L$ y viceversa. Este valor es totalmente empírico y los resultados óptimos se han dado para $\alpha = -1.5$. En las siguientes ilustraciones se puede observar los diferentes perfiles de la función $\gamma_i{}^L$ en base a los valores de α y β elegidos.

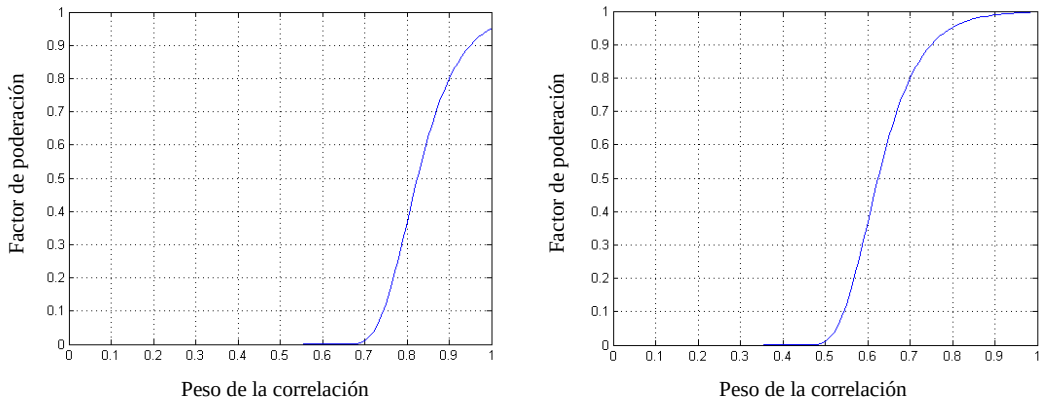


Figura 57: A) Peso de la correlación con α = -1.5, β = -2 B) Peso de la correlación con α = -1.5, β = -4

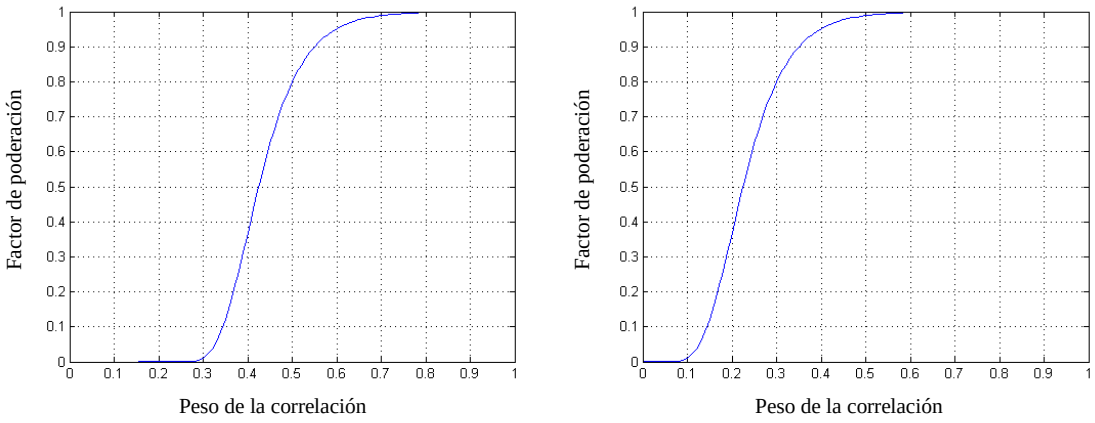


Figura 58: A) Peso de la correlación con α = -1.5, β = -6 B) Peso de la correlación con α = -1.5, β = -8

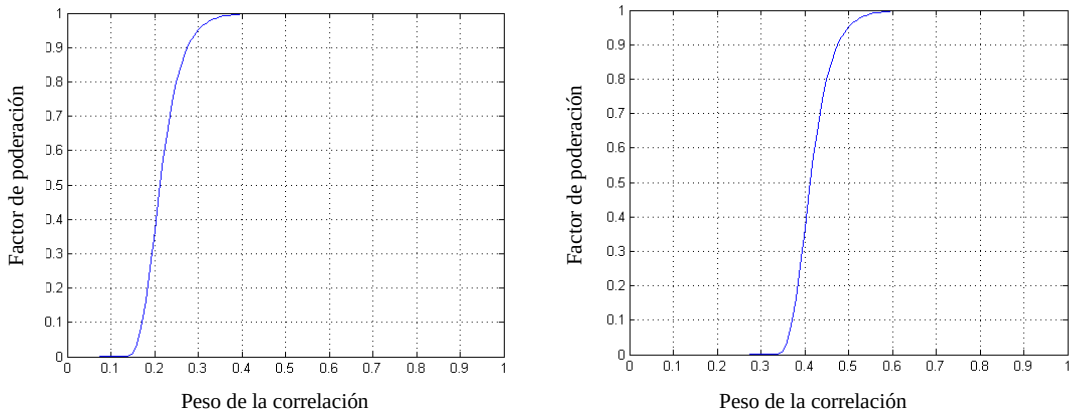


Figura 59: A) Peso de la correlación con α = -3, β = -2 B) Peso de la correlación con α = -3, β = -4

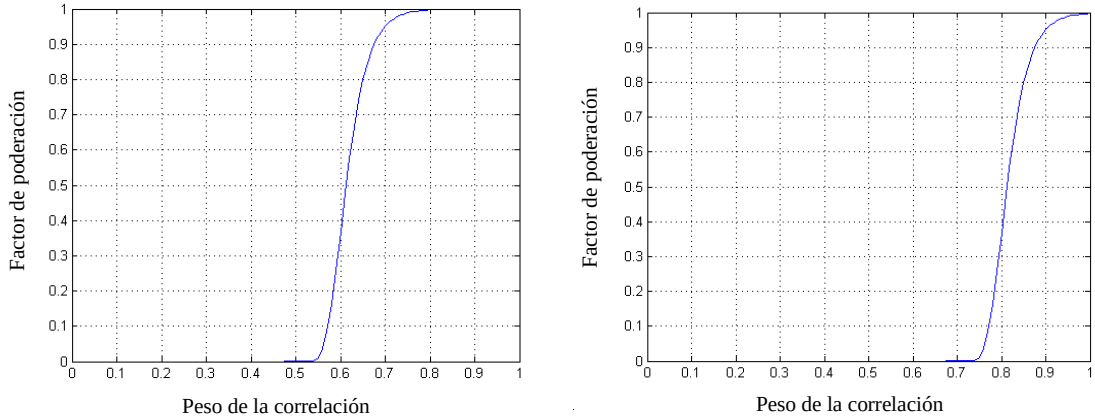


Figura 60: A) Peso de la correlación con α = -3, β = -6 B) Peso de la correlación con α = -3, β = -8

Es precisamente para $\alpha = -1.5$ y $\beta = -4$ cuando se alcanza un equilibrio adecuado entre la penalización por una alta correlación y la reducción del índice de correlación cuando éste es menor a un umbral cercano a 0.7. Los perfiles de la función anteriores han sido generados de manera general, pero éstos deben ser aplicados para cada una de las actividades para las cuales ha sido entrenado el

sistema. De esta forma, será obtenido un valor para el factor de peso de correlación para cada actividad y estadístico, como por ejemplo para el estadístico media aritmética y la actividad *Walk*, se obtendrá el factor de peso $\gamma^{walk}_{MediaAritmética}$.

El proceso anterior debe ser realizado para cada una de las actividades, obteniendo por tanto un factor de peso de correlación para cada una de las actividades capaces de ser reconocidas por el sistema. Finalmente, se obtendrá el coeficiente de correlación general para la variable *i*, el cual indica el grado de correlación de dicha variable con el resto del sistema, teniendo en cuenta las diferentes influencias sobre cada una de las actividades. Dicho coeficiente se denota por Ic_i (índice de correlación de la variable i) y viene dado por la siguiente expresión:

$$Ic_i = \frac{\sum_{i=1}^{n} {\gamma_i}^L}{|L|}$$

donde |L| indica el cardinal de actividades capaces de ser reconocidas por el sistema. Además se tiene que $0 \leq Ic_i \leq 1$, por lo tanto, los diferentes coeficientes de correlación general pueden ser comparados entre las diferentes variables, pudiendo así eliminar aquellas variables cuyo índice de correlación sea mayor, lo cual indica que su aporte de información autónoma, es decir, información no aportada por el resto de variables del sistema, es bajo.

Para ilustrar lo expuesto anteriormente, seguiremos el proceso para el cálculo del índice de correlación general para el problema del reconocimiento de actividades. En este caso se aplicará el factor a un conjunto de ventanas temporales obtenidas por el sistema a partir de actividades realizadas por el usuario, concretamente para los estadísticos simples recogidos en la tabla inferior. Dichos estadísticos se corresponden con los valores obtenidos en el reconocimiento modular basado en el módulo de la señal de acelerometría. Tras obtener todos y cada uno de los factores de correlación general para cada variable, se han obtenido los siguientes resultados:

Estadístico	Valor Ic_i
Módulo máx frecuencial	0.1000
Frecuencia máxima	0.1108
Mediana mod. frecuencial	0.1825
Frecuencia mínima	0.1825
Mediana	0.2111
Máximo	0.2957
Desv. Media	0.3023
Media aritmética	0.3172
Media geométrica	0.3272
Módulo mín frecuencial	0.3783
Desv. Típica	0.3846
Mínimo	0.4250

Tabla 1: Valores del índice de correlación para el sistema modular con un total de 12 variables

Del estudio anterior podemos concluir que los estadísticos que mejor representan a la muestra y con una mayor independencia, tomando como umbral un valor de 0.3, son *Máximo, Mediana, Desviación Media, Módulo máx frecuencial* y *Frecuencias máxima y mínima*. De esta forma hemos pasado de emplear 12 estadísticos para el proceso del reconocimiento de las actividades que el usuario está llevando a cabo, a emplear tan sólo 6, reduciendo por tanto un 50% el coste de cálculo de estadísticos en el dispositivo móvil y el almacenamiento de dicha información, además de conseguir un aumento de fiabilidad en el modelado del problema.

COMPARATIVA DE LOS MÉTODOS DE SELECCIÓN DE VARIABLES

En esta sección será realizada una evaluación y comparación de los métodos de selección y eliminación de variables expuestos. Los resultados se basarán en la precisión y la eficacia de la clasificación en base a un conjunto de aprendizaje y siguiendo un técnica de clasificación común para ambos. En todos los casos se seguirá una técnica basada en cross-validation para evitar dependencias del conjunto de datos. Concretamente se empleará el método de k-fold cross-validation, mediante el cual el conjunto de datos inicial será dividido en 10 subconjuntos (10-fold). El método de eliminación de variables y clasificación de la actividad será ejecutado 10 veces, en las cuales se seleccionarán 9 subconjuntos para realizar el aprendizaje, mientras que uno de ellos se reservará para la validación. Dicho subconjunto de validación será diferente para cada una de las ejecuciones del método.

A la hora de comparar los métodos se tendrán en cuenta cuatro indicadores:

- *Tiempo de ejecución del proceso de selección y eliminación de variables en el servidor*: este proceso será llevado a cabo una sola vez, cuando el usuario finalice la sesión de aprendizaje. Sin embargo, la reducción de este tiempo liberará al servidor de una carga computacional que podría ser necesaria si la plataforma da servicio a un elevado número de usuarios. Como se ha comentado anteriormente, los métodos de eliminación de variables deben ser ejecutados para cada uno de los usuarios de manera independiente, debido a las características propias de cada usuario a la hora de realizar las diferentes actividades que pueden ser reconocidas por el sistema.
- *Tiempo de ejecución del proceso de reconocimiento en el dispositivo*: el hecho de poder eliminar una serie de variables del conjunto de datos reduciendo el número de ellas implicadas en el reconocimiento, tendrá un efecto positivo en el rendimiento del dispositivo. Esto se debe a que las limitaciones de los dispositivos móviles hacen que el procesamiento para la obtención de datos procesados a partir de los datos RAW procedentes del sensor sea más lenta que en el caso de los dispositivos de escritorio o los servidores. Por este motivo, si es posible conseguir una reducción del número de variables a partir de las cuales es posible realizar el reconocimiento, la eficiencia del dispositivo será mayor debido al menor cálculo que es necesario llevar a cabo. Esto tendrá una influencia positiva en el rendimiento general del dispositivo así como en el tiempo de vida de sus baterías.
- *Precisión del reconocimiento*: la eliminación de variables implicará en la mayoría de los casos una pérdida de información debida al porcentaje de varianza perdida por el proceso de eliminación. Por este motivo, es extremadamente importante controlar y gestionar dicha pérdida de información con el fin de evitar que la precisión del sistema descienda por debajo de un determinado nivel. A dicho nivel le denominaremos umbral de error asumible y se calculará en base a la diferencia entre el error de clasificación previo al proceso de eliminación de variables y el error a posteriori tras este proceso.
- *Impacto de la inclusión de actividades:* es importante recordar en este punto que el sistema desarrollado debe ser totalmente versátil y tener la capacidad de incluir de una manera

sencilla y transparente la capacidad de reconocimiento de nuevas actividades. La inclusión de una nueva actividad trae como consecuencia un reentrenamiento del sistema, para lo cual podrá ser necesario calcular de nuevo los coeficientes que nos permitirán determinar la posible eliminación de ciertas variables. En este punto de control se evaluará el impacto sobre el sistema en general y sobre el subsistema de eliminación de variables en concreto, que provoca la inclusión de nuevas clases en el sistema.

Análisis del tiempo de ejecución en el servidor

Comenzaremos comparando el tiempo de procesamiento en el servidor para la obtención de las variables que pueden ser eliminadas del sistema. Para ello se realizarán varias pruebas modificando el número de datos en el conjunto de entrenamiento, el número de actividades que pueden ser reconocidas por el sistema así como el conjunto de variables iniciales del sistema a partir de las cuales se realizará el filtrado de las mismas.

- *Relación con el número de datos del conjunto de entrenamiento*

Para llevar a cabo esta prueba, se proporcionan 6 conjuntos de entrenamiento distintos en cuanto al número de datos de aprendizaje. El primero de ellos contendrá 196 datos correspondientes al mismo número de ventanas temporales, posteriormente se aplicarán los criterios de reducción de variables expuestos anteriormente a unos conjuntos de 150, 120, 96, 70 y 50 datos consecutivamente. Con esta batería de pruebas comprobaremos la influencia del número de datos en cada uno de los métodos.

Con el fin de evitar desviaciones imprevisibles y poder comparar los resultados obtenidos en la aplicación de los métodos anteriores sobre los conjuntos de pruebas descritos, todas las pruebas se han realizado en el mismo equipo, eliminando todas los procesos que pudieran influir en el rendimiento del algoritmo y sobre un único núcleo dedicado. Evidentemente, en producción no puede ser replicado el mismo entorno y las restricciones descritas, por lo que los valores expuestos a continuación tienen un fin meramente orientativo y sobre todo, brindan la posibilidad de comparar los métodos de una manera clara y transparente.

En las Figura 61 y 68 se muestra el tiempo de ejecución de los métodos basados en el coeficiente de correlación y en el análisis de componentes principales respectivamente para un tamaño del conjunto de aprendizaje de 50 a 1500 ventanas de entrenamiento. Según se puede observar en dichas gráficas, en tiempo de ejecución para el método basado en ACP es muy inferior al basado en el coeficiente de correlación, concretamente el primero es unas 100 veces más rápido que el segundo. Esto se debe principalmente al coste computacional provocado por la generación de las regresiones para eliminar la influencia de las diferentes variables durante el proceso de obtención del coeficiente de correlación general de cada actividad.

Otra característica que puede ser observada en las figuras citadas es la tendencia de los tiempos de procesado. Mientras que en el método basado en el coeficiente de correlación el crecimiento temporal es logarítmico, en el método basado en el ACP dicho tiempo de ejecución es prácticamente constante.

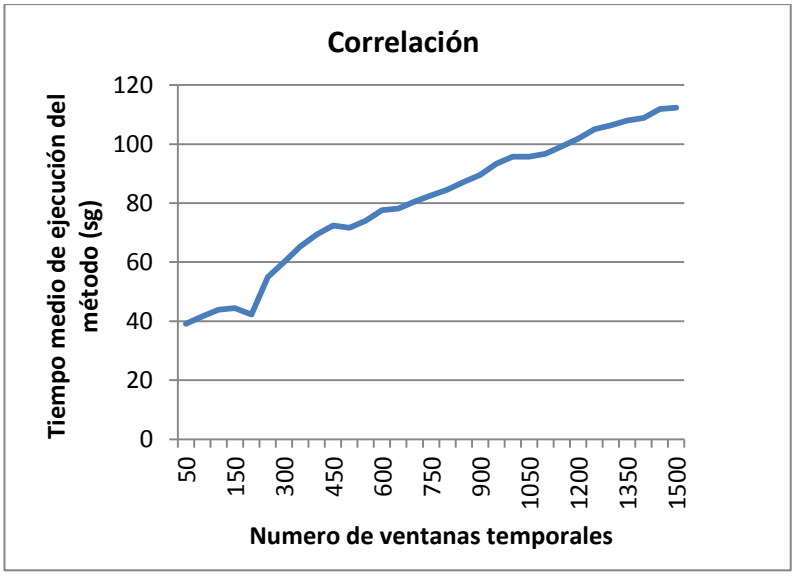


Figura 61: Tiempo de ejecución del método basado en el coeficiente de correlación en función del número de ventanas temporales

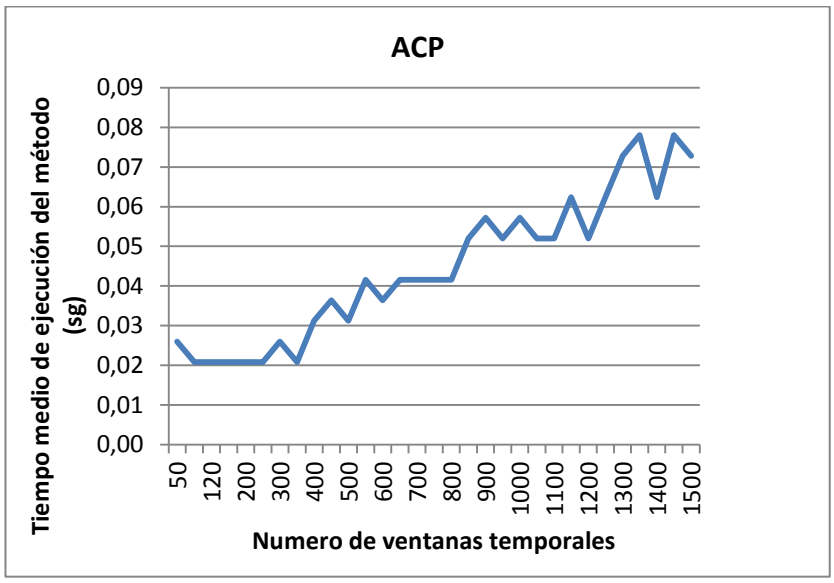


Figura 62: Tiempo de ejecución del método basado en el ACP en función del número de ventanas temporales

Como conclusión podemos destacar que evidentemente el método basado en el ACP es mucho más rápido que el basado en el coeficiente de correlación. Sin embargo, tanto en esta comparativa como en el resto de comparativas sobre el tiempo de ejecución en el servidor, debemos tener en cuenta que dicho procesamiento será realizado tan sólo una vez a la finalización del proceso de entrenamiento del sistema o, en general, cuando el usuario desee reentrenar al sistema manualmente. Por este motivo, un tiempo excesivo podría haber sido crítico, sin embargo los tiempos manejados en el método del coeficiente de correlación no son excesivamente grandes, por lo que el coste temporal estaría dentro de los límites aceptables.

Por otro lado, en un sistema real donde el entrenamiento es llevado a cabo en un servidor con múltiples núcleos y una potencia de cálculo superior, el tiempo general tendería a bajar, disminuyendo en cierta medida el elevado tiempo de procesamiento.

- *Relación con el conjunto de actividades reconocidas*

En este caso comprobaremos la dependencia del tiempo de ejecución del proceso de eliminación de variables en el servidor en función del conjunto de actividades que pueden ser reconocidas por el sistema. Una primera reflexión nos haría pensar que un mayor número de actividades posibles de ser reconocidas por el sistema conllevará un mayor tiempo de procesado, aunque comprobaremos que esto no siempre es cierto.

Para realizar las pruebas se ha empleado un conjunto de aprendizaje de 1500 ventanas temporales en todos los casos. Sin embargo los conjuntos han sido modificados para que puedan reconocerse entre 1 y 10 actividades distintas. El resultado de realizar el procesamiento para los dos métodos expuestos sobre el conjunto de actividades se puede ver en la Figura 63 y en la Figura 64. En la primera, se recoge la evolución temporal para el caso del método basado en el coeficiente de correlación, siendo la segunda la asociada al método de ACP. En dichas figuras se puede observar la evolución del tiempo de procesamiento en función del número de actividades reconocidas.

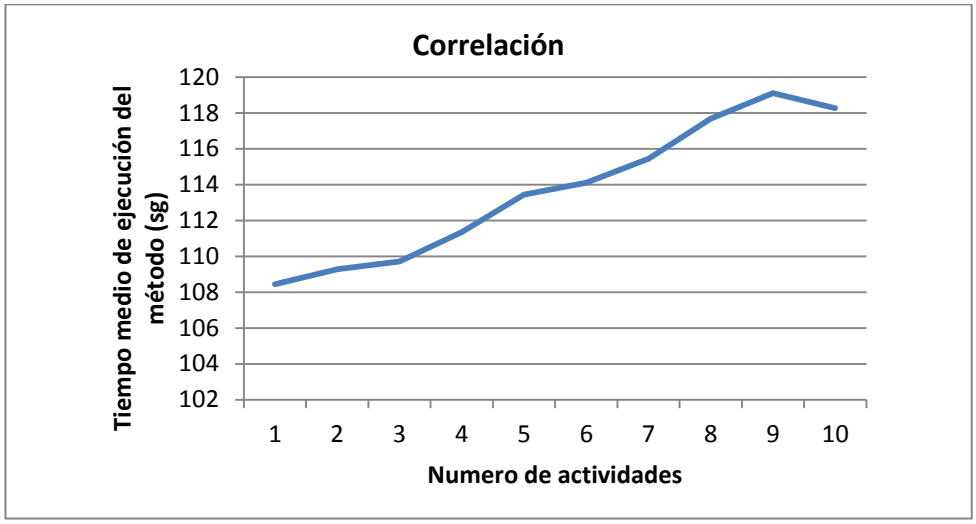


Figura 63: Tiempo de ejecución del método basado en el coeficiente de correlación en función del número de actividades reconocidas

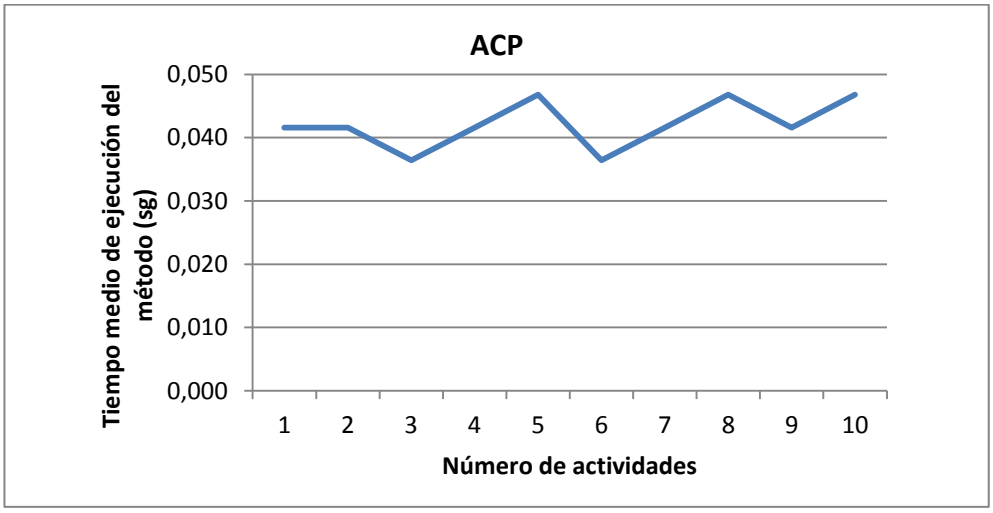


Figura 64: Tiempo de ejecución del método basado en el ACP en función del número de actividades reconocidas

En el caso del ACP se observa una clara independencia del número de actividades reconocidas, puesto que los diferentes valores temporales no siguen una tendencia significativa hacia ningún valor, sino que los valores son totalmente irregulares. Esto era de esperar, ya que dicho método no tiene en cuenta las actividades concretas asociadas a los diferentes datos del conjunto de entrenamiento, sino que simplemente procesa las diferentes variables del sistema en busca de una serie de combinaciones lineales (componentes) que maximice el porcentaje de la varianza expresado en cada una de ellas.

Sin embargo, en el primer de los métodos comparados sí puede observarse una clara evolución ascendente a medida que aumenta el número de actividades posibles de ser reconocidas por el sistema. Como era de esperar, esto se debe a que el método desarrollado sí tiene en cuenta las diferentes actividades a la hora de obtener el coeficiente de correlación general. Sin embargo el aumento del tiempo de procesamiento es muy moderado, por lo que no compromete la funcionalidad del sistema. Observando la Figura 63 se concluye que la diferencia entre realizar el proceso de eliminación de variables con 1 sola actividad posible y realizarla con 10 actividades es de apenas 10 segundos. Cabe destacar que dicho tiempo disminuirá a su vez en función del número de datos en el conjunto de aprendizaje.

- *Relación con el número de variables iniciales*

Concluiremos el análisis del tiempo de procesamiento en el servidor comprobando la influencia del número de variables iniciales de las que se parte para realizar el ACP o la obtención del coeficiente de correlación general, en función del método empleado. Durante todo el proceso de diseño y pruebas del sistema, el número de variables iniciales se ha ajustado a 48 para el caso del método basado en acelerometría triaxial y tan sólo 12 para el caso de acelerometría modular. En los resultados anteriores se mostraron los resultados para el caso de acelerometría triaxial.

En este apartado reduciremos el número de variables base y comprobaremos el funcionamiento del sistema en base a ello. Para eliminar influencias externas, el tamaño del conjunto de aprendizaje así como el número de actividades reconocidas permanecerán constantes e iguales a 780 y 10 respectivamente.

En la Figura 65 y en la Figura 66 se puede visualizar la tendencia de ambos métodos cuando el número de variables base a partir de las cuales se realizará el proceso de eliminación es modificado. A diferencia del caso anterior, tras estudiar el impacto del número de variables en el proceso se puede llegar a la conclusión que para ambos métodos se produce un crecimiento en el coste computacional necesario para la eliminación. Sin embargo este crecimiento es más significativo en el caso del método basado en el coeficiente de correlación.

Merece ser mencionado el efecto que se produce en torno a 28 variables base, en dicho punto el tiempo computacional experimenta un crecimiento excesivo que rompe con la dinámica anterior de crecimiento leve. Este efecto se produce en ambos métodos, pero es más acusando en el basado en el coeficiente de correlación. El motivo del escalón que se produce en el citado punto es el hecho de que existen demasiadas variables para ser inferidas a partir de los datos presentes en el conjunto de aprendizaje, produciéndose un error probablemente excesivo en la clasificación o en la obtención de los parámetro necesario para el análisis de la regresión en el caso del método basado en coeficientes de correlación o para generar las componentes adecuadas para el caso del ACP. Este problema puede ser resuelto mediante la inclusión de nuevos datos al conjunto de aprendizaje, aunque esta solución no es viable debido al elevado número de entradas que es necesario introducir. Por este motivo a la hora de solucionar este problema en el sistema real se ha optado por reducir el número de variables base,

eliminando aquellas que mediante un análisis de correlación independiente se ha llegado a la conclusión que es menos significativo.

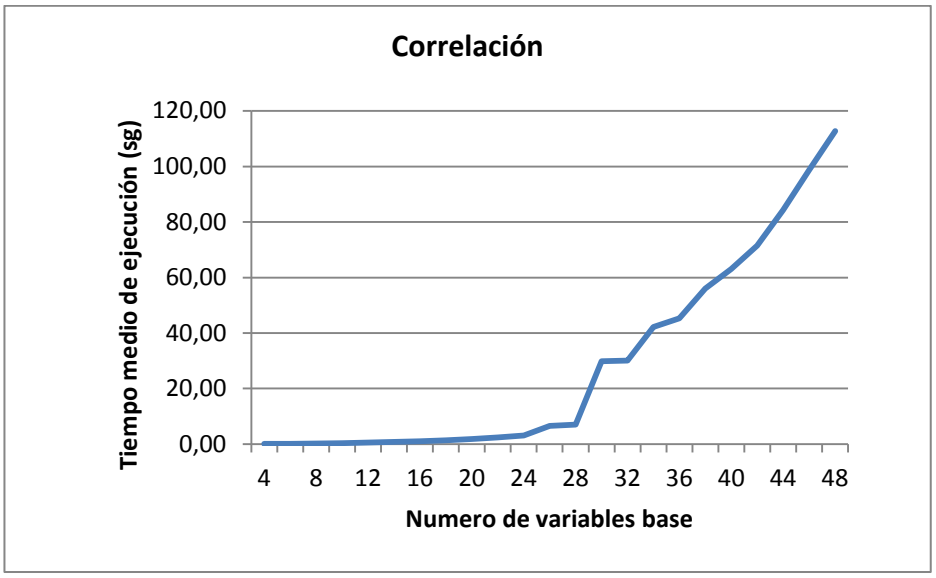


Figura 65: Tiempo de ejecución del método basado en el coeficiente de correlación general en función del número de variables base

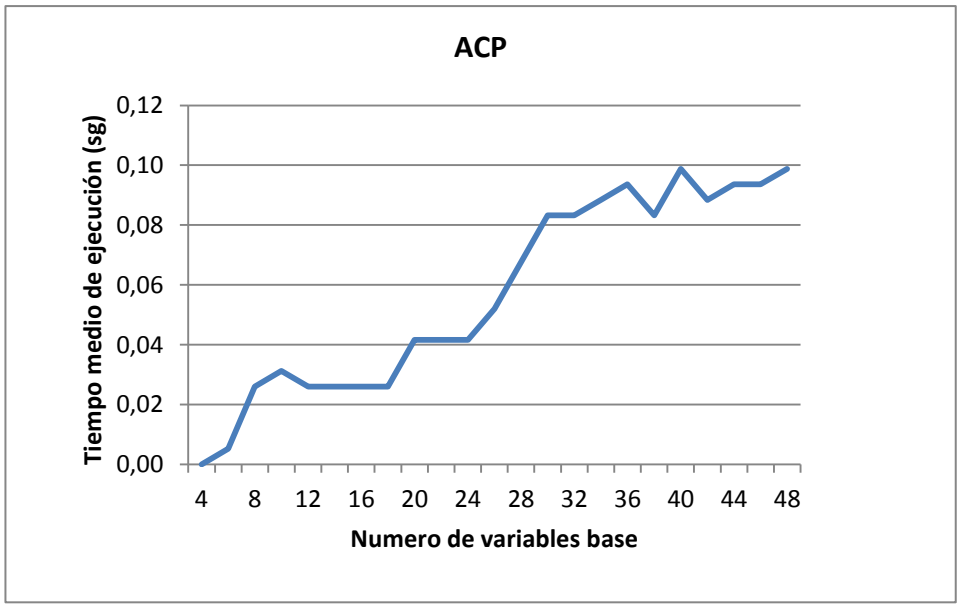


Figura 66: Tiempo de ejecución del método basado en el ACP en función del número de variables base

En esta comparativo podemos llegar a la conclusión que para un número reducido de variables iniciales (menor a 28), el comportamiento de ambos algoritmos es computacionalmente correcto. Sin embargo, el aumento de las variables iniciales por encima de este umbral conlleva un acusado aumento en el tiempo necesario para el procesamiento de los datos, siendo éste más acusado en el método de coeficientes de correlación general, puesto que se precisa ejecutar repetidas veces los procesos de regresión a los que afecta este aumento de variables.

Continuando con la comparativa entre los métodos de selección de características propuestos para la reducción del número de variables del sistema, estudiaremos en este caso el tiempo de ejecución sobre el dispositivo móvil. A diferencia del caso anterior, en esta sección no analizaremos directamente el tiempo de ejecución del método, ya que carece de sentido realizar la reducción de variables en el propio dispositivo debido al elevado coste computacional que requiere y las características limitadas de los sistemas móviles. En cambio, sí es posible determinar el impacto que tiene el aplicar la reducción de variables propuesta por cada uno de los métodos sobre el tiempo de clasificación y/o generación de variables útiles para el reconocimiento.

Llegados a este punto debemos definir el concepto de *variable útil*. Decimos que una determinada variable X_i es útil para el sistema si dicha variable es empleada, bien directamente o a través de otra variable Y_j que es una combinación lineal de X_i entre otras, durante el reconocimiento de actividades llevado a cabo por el sistema. En este sentido, existe una clara diferencia entre los métodos de eliminación basados en ACP y en coeficientes de correlación. En el primero de ellos, como se comentó cuando se detallaba el método, se basa en la generación de un conjunto de variables $Y_1, Y_2, Y_3, \dots, Y_N$ que son cada una de ellas una combinación lineal de las variables base $X_1, X_2, X_3, \dots, X_N$. De esta forma, el número de variables útiles del sistema seguirá siendo N, ya que es necesario obtener las N variables base para generar a partir de ellas la combinación lineal propuesta por el método de ACP. La relación entre las variables iniciales y las útiles puede observarse en la siguiente función:

$$f(X_1, X_2, X_3, \dots, X_N) \rightarrow g(Y_1, Y_2, Y_3, \dots, Y_N)$$

En contraposición con el anterior, el método de coeficientes de correlación parte de las N variables base para determinar cuáles de ellas son candidatas a ser eliminadas del conjunto de variables debido a la correlación entre ellas. Una vez seleccionadas las variables a eliminar, el conjunto de variables útiles será un subconjunto de las variables base. De esta forma, partimos de la obtención de N variables para poder llevar a cabo el proceso de aprendizaje y eliminación de variables, aunque una vez realizado éste, el número de variables útiles se reducirá a J, siendo evidentemente J menor que N. En este caso se puede plantear el proceso mediante la siguiente formulación:

$$f(X_1, X_2, X_3, \dots, X_N) \rightarrow f(Y_1, Y_2, Y_3, \dots, Y_J), \quad J < N$$

Esta reducción del número de variables útiles respecto al conjunto de variables iniciales tiene una implicación directa en el rendimiento del reconocimiento de las actividades en el dispositivo móvil. Dicha implicación se basa en que es necesario calcular un menor número de estadísticos (variables) para llevar a cabo el reconocimiento, por lo que los siguientes aspectos son mejorados:

- *Incremento de la estabilidad de la aplicación cliente:* de cara a la realización de un sistema que permita llevar a cabo el reconocimiento de actividad en el dispositivo, la reducción de variables que deben ser calculadas liberará al procesador de una carga computacional considerable, especialmente teniendo en cuenta las limitaciones de los dispositivos móviles.
- *Aumento del tiempo de baterías:* derivado del beneficio anterior, la reducción de los procesos que deben llevarse a cabo en el dispositivo conlleva una reducción a su vez del consumo energético. Como se citaba cuando introducíamos el desarrollo de aplicaciones ubicuas en sistema móviles, la batería es un aspecto trascendental que debe ser tenido en cuenta para conseguir un sistema usable.

- *Reducción del tráfico de datos:* determinados aspectos de la plataforma necesitan que los datos del usuario sean enviados al servidor, lo cual se verá en profundidad cuando describamos la plataforma de reconocimiento. Por este motivo, es necesario enviar con cierta frecuencia datos relativos a las ventanas temporales generadas a partir de la realización de una actividad, concretamente las variables asociadas a estos datos. Evidentemente, cuando menor sea el número de variables necesitadas por el sistema para conseguir completar el reconocimiento, menor será las variables que deben ser transferidas al servidor. Teniendo en cuenta las limitaciones que hoy en día posee el tráfico de datos en móviles, es muy interesante la reducción del volumen de datos a transferir.

Para realizar el análisis del sistema de reconocimiento en base a los métodos estudiados, se llevará a cabo un aprendizaje del sistema para 5 actividades diferentes, con un total de 96 ventanas temporales distribuidas entre estas actividades. Para corroborar de manera más evidente la necesidad de la reducción de variables, se empleará un sistema de acelerometría triaxial. En consecuencia se tendrá un total de 48 variables base, a partir de las cuales se reducirán en función del método concreto hasta obtener el conjunto de variables útiles.

El análisis se realizará en base al tiempo necesario para que el dispositivo procese las distintas ventanas temporales y obtenga los estadísticos asociados. En este sentido se harán tres casos de prueba, el primero de ellos consistirá en la generación de las variables sin ningún tipo de reducción, seguidamente se realizará otra prueba empleando una reducción basada en ACP y por último, una proceso de eliminación de variables basado en coeficientes de correlación.

Debemos tener en cuenta que cada prueba consiste en la lectura de 100 ventanas temporales, obteniendo en cada una de ellas el tiempo necesario para el procesamiento de las variables. No se tendrá en cuenta el periodo de recopilación de datos de la ventana (en torno a 5 segundos) ni el tiempo necesario para enviar los datos al servidor de la aplicación si fuera necesario.

En la Figura 67 se presenta una comparativa del tiempo de cómputo para la obtención de los estadísticos necesarios en función del método de reducción de variables seleccionado. Dicho tiempo será calculado para cada ventana temporal, con el fin de obtener un tiempo medio para generar dichas variables útiles del sistema.

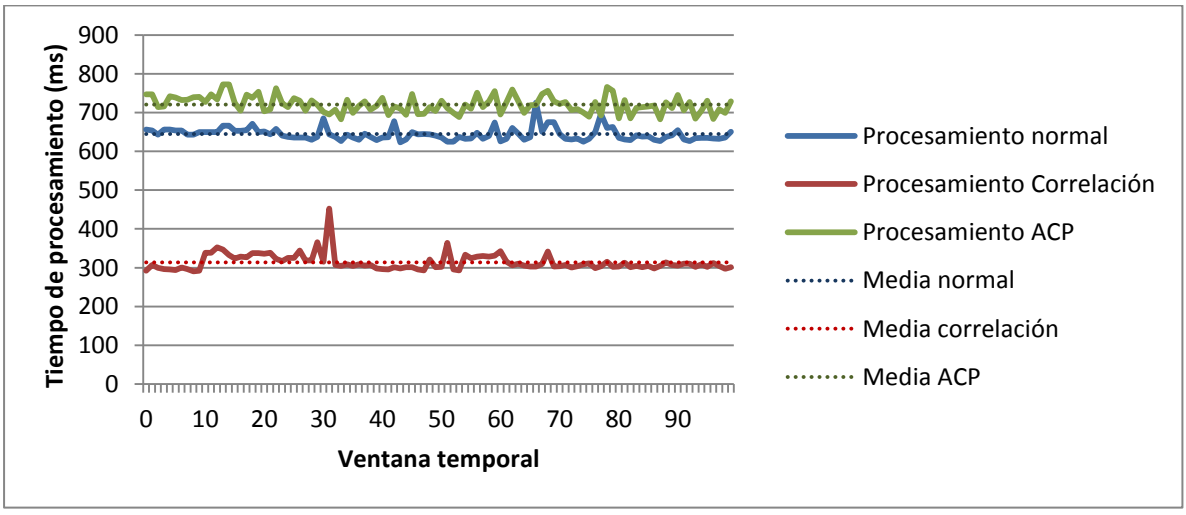


Figura 67: Comparativa del tiempo empleado en obtener las variables necesarias en un dispositivo móvil

Comenzaremos en este caso observando el tiempo de procesamiento para el proceso normal, es decir, para un sistema de reconocimiento de actividades que no emplea ningún método de reducción de variables para simplificar la complejidad del problema. La media del tiempo de generación de estadísticos es 644ms, es decir, algo más de medio segundo. Este tiempo no es excesivamente elevado teniendo en cuenta que se están generando 48 estadísticos por cada ventana temporal. Dado que la frecuencia a la que trabajan los sensores de acelerometría sobre los que han sido realizadas las pruebas está en torno a 25Hz y las ventanas temporales seleccionadas tienen una duración de 5 segundos, el total de datos que debe procesar el dispositivo es de 125 elementos. Muchas de las variables seleccionadas deben obtenerse a partir del conjunto total de datos, sin que puedan ser calculadas en base a otras variables, por lo que en la mayoría de las 48 variables se deberán procesar 125 datos para obtener el valor de éstas.

Siguiendo con los datos obtenidos a partir del sistema sin eliminación de variables, podemos observar que el tiempo de ejecución no es constante en todas las ventanas temporales. Esta diferencia temporal entre ventanas se debe en la mayoría de los casos al tiempo dedicado a la tarea de generación de estadísticas por parte del sistema operativo. En otros casos sin embargo, podría deberse a que el conjunto de datos posee más o menos elementos, debido en gran medida a la frecuencia oscilante del sensor de acelerometría, influyendo por tanto en el número de datos a procesar. En general, a la hora de comparar los valores para cada uno de los métodos, recurriremos a la media.

Continuaremos analizando el sistema de reconocimiento basado en el método de ACP. En este caso podemos observar como el tiempo es mayor que el anterior, a pesar de haber realizado un proceso de selección de variables. Este resultado que a priori podría parecer poco intuitivo, se debe a la propia manera de trabajar de las componentes principales. Anteriormente se presentó la manera en la que el ACP genera las nuevas variables, así como que las nuevas variables eran una combinación lineal de las primarias. Debido a esto último, el sistema trabajará con varias combinaciones de todas las variables iniciales, por lo que es necesario generar dichas combinaciones. De esta forma, el tiempo total en el que se procesarán las variables de una ventana será igual a la suma del tiempo empleado en obtener todas las variables base más el tiempo añadido de generar la combinación lineal de éstas para crear las diferentes componentes con las que trabajará el sistema.

En este sentido, podemos observar como la media del método ACP asciende a 720ms, o lo que es lo mismo, este método provoca un aumento del tiempo de computación de un 10% respecto al sistema anterior en el que no se empleaba reducción de variables. En cambio, aunque en este punto se aumente el tiempo de ejecución, más adelante observaremos como a pesar de ello, la selección variables mejora le precisión del sistema.

Por último estudiaremos el comportamiento del método de reducción de variables basado en coeficientes de correlación. A primera vista, observando la Figura 67 es posible llegar a la conclusión que este último método reduce significativamente el tiempo de ejecución en el dispositivo para la generación de las variables útiles del sistema. Esto se consigue gracias a que el método basado en correlación, como ha sido comentado anteriormente, permite la eliminación completa de variables, evitando así el coste de calcularlas a partir de los datos de la ventana temporal. Concretamente, en el sistema cuyo rendimiento se muestra en la Figura 67, se ha reducido el número de variables de las 48 iniciales a 15, lo cual representa una disminución del 68,75%.

Si obtenemos la media de la distribución de tiempos para la técnica de correlación, muestra que el tiempo medio de procesamiento es de 314 ms, lo que reduce el tiempo necesario para el cálculo de

variables en el sistema sin selección de variables y en el sistema basado en ACP en un 51,24% y en un 56,39% respectivamente.

Para dar una visión más amplia del tiempo acumulado por los distintos métodos a lo largo de un proceso de reconocimiento, en la Figura 68 se presenta el tiempo empleado en realizar el cálculo de variables en el dispositivo durante un proceso de reconocimiento de 60 minutos (700 ventanas temporales). Como se podía esperar a partir de los resultados anteriores, el tiempo acumulado para la obtención de variables en el caso de la reducción por ACP es de 93 segundos (1,5 minutos) cada hora de reconocimiento, mientras que el tiempo acumulado para el caso de la reducción por correlación es de tan sólo 13 segundos.

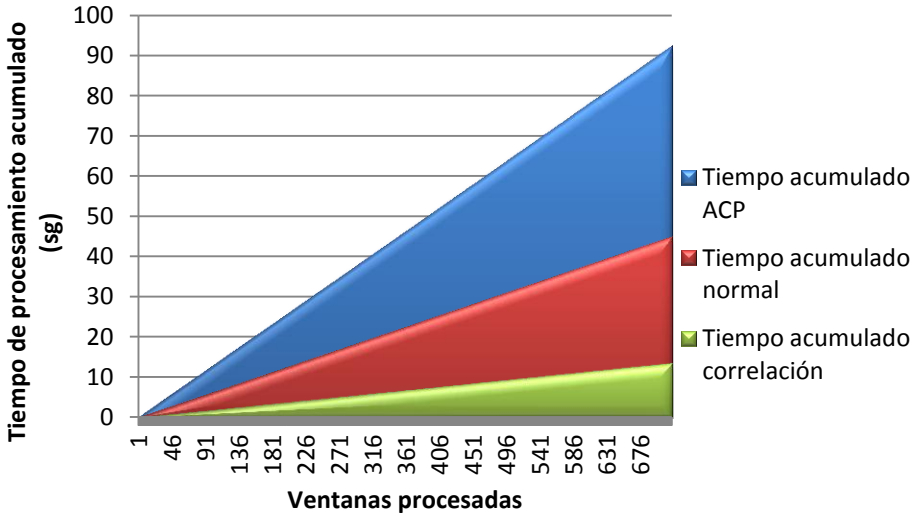


Figura 68: Tiempo acumulado de procesamiento en el dispositivo por los distintos métodos en un periodo de 60 minutos

El tiempo de obtención de las variables útiles del sistema en el dispositivo no es un proceso crítico en cuanto a estabilidad, ya que en general si el tiempo empleado en obtener dichas variables es menor al tamaño de la ventana temporal, el proceso de generación de variables se verá enmascarado por el proceso de recolección de datos. Sin embargo, si el tiempo de generación supera al tamaño de la ventana, el sistema podría volverse inestable y podrían producirse casos de consumo excesivo de memoria. En todo caso, tanto los dos métodos de reducción de variables como el método tradicional no superan dicho umbral, por lo que no existe peligro. En cambio, sí que es aconsejable reducir al máximo el procesamiento debido a las inherentes limitaciones de los dispositivos móviles, por lo que la conclusión en esta sección de la comparativa sería elegir el método de reducción de variables mediante el análisis de la correlación.

Análisis de la precisión del reconocimiento

La precisión del proceso de reconocimiento es un factor que debe ser tenido muy en cuenta a la hora de decidir las diferentes técnicas que serán empleadas en el sistema de reconocimiento de actividades. Entre este conjunto de técnicas se encuentran los métodos de reducción de variables expuestos anteriormente. En este caso analizaremos el rendimiento de los diferentes métodos en base a la precisión que alcanzan los sistemas mediante una técnica concreta de aprendizaje tras ser aplicados dichos métodos.

Para simplificar el análisis y mostrar unos resultados comparables entre los distintos procesos de reducción de variables, que es el objetivo de esta sección, se aplicará una técnica común tanto para el aprendizaje como para el reconocimiento de las actividades que llevan a cabo los usuarios. Concretamente el proceso que llevaremos a cabo a lo largo de los distintos casos de prueba de la comparativa que se expone son los siguientes:

- Obtener un conjunto de ventanas temporales a partir del proceso de aprendizaje que el usuario debe llevar a cabo en el dispositivo.
- En base a las ventanas temporales anteriores, aplicar el método de reducción de variables seleccionado y eliminar o generar las nuevas variables útiles del sistema.
- Ejecutar el proceso de aprendizaje mediante una técnica concreta. En este caso se empleará una técnica basada en redes neuronales. Concretamente se generará una red neuronal de 10 neuronas en la capa de salida y tantas neuronas en la capa de entrada como sea necesario, dependiendo del caso. Dicha red poseerá además una capa oculta. El aprendizaje se realizará mediante un método supervisado, evaluando el error en cada iteración mediante el error cuadrático media, aplicando una técnica de aprendizaje basada en el algoritmo de Levenberg-Marquardt.
- Comenzar el proceso de reconocimiento en el dispositivo, a través del cual se capturará una ventana temporal cada 5 segundos con los valores de acelerometría obtenidos.
- A partir de la ventana temporal anterior se aplicará el algoritmo de clasificación específico. En el caso que nos ocupa, para realizar la comparativa, se aplicará el algoritmo de clasificación tradicional dada la red neuronal, es decir, introduciendo los diferentes valores a la entrada de la red y seleccionando aquella actividad asociada a la neurona de la capa de salida con mayor nivel de activación.

Una vez realizados los pasos anteriores, se aplicará la clasificación a cada nueva ventana temporal generada, clasificando como correcto el reconocimiento si la etiqueta generada por el sistema de reconocimiento coincide con la actividad real que el usuario marcó en el proceso de reconocimiento. Por tanto, todo el proceso de reconocimiento empleado para la comparativa, se realizará de manera semi-asistida, puesto que es necesario saber la actividad real con el fin de determinar la corrección del reconocimiento respecto a dicha actividad de referencia.

Siguiendo lo anteriormente expuesto, se ha obtenido la Tabla 2, sin embargo, antes de estudiarla será necesario determinar el significado y el motivo de la elección para los diferentes índices de precisión seleccionados. En primer lugar aparece el índice *Epochs*, el cual hace referencia al número de iteraciones que han sido necesarias para obtener un factor de precisión adecuando durante el entrenamiento empleando la técnica de redes neuronales. Este elemento refleja un indicio del nivel de dificultad asociado al proceso de entrenamiento para el conjunto de variables determinado. Seguidamente encontramos el tiempo en segundos que ha tardado el método de aprendizaje en encontrar un peso adecuado para las conexiones sinápticas de la red neuronal que satisfaga un índice de precisión mínimo. Tras esto aparecen cuatro porcentajes que nos indican la precisión del método en los conjuntos de entrenamiento, validación y test, además de un porcentaje de acierto global calculado como una media aritmética simple de los tres anteriores. El acierto en el entrenamiento generalmente será 1 o cercano a este valor, ya que el algoritmo de aprendizaje de la red neuronal ajustará los pesos en base a este conjunto. El porcentaje de acierto de validación indicará las muestras bien clasificadas en un grupo de datos empleado para comprobar la precisión del aprendizaje. Por último encontramos el porcentaje de aciertos del conjunto de test, el cual nos aporta información de cómo se comportaría en un entorno real el sistema desarrollado.

Para cada uno de los métodos de eliminación de variables se han obtenido dichos parámetros en base a la media de dichos valores sobre 20 entrenamientos con distintos conjuntos de test, lo cual reduce la probabilidad de aparición de valores anómalos.

Tras detallar la manera en la que se llevará a cabo el proceso de reconocimiento para la generación de las estadísticas de precisión, expondremos el resultado de las mismas para el caso en el que no se lleve a cabo ningún método de eliminación de variables. Los resultados del conjunto de análisis pueden verse en la Tabla 2, donde se recogen los valores obtenidos para cada uno de los índices elegidos para obtener la precisión del sistema.

Como podemos observar, para el caso del método sin reducción, el número de iteraciones del algoritmo fueron 10, realizadas en un tiempo total de más de 5 segundos que, como podemos comparar con el resto de métodos de reducción, es un tiempo relativamente elevado. Esto se debe al número de parámetros con los que debe trabajar, puesto que la red se volverá más compleja debido al mayor número de neuronas en la capa de entrada. En cambio, el tiempo de procesamiento en el resto de métodos es del orden de décimas de segundo es decir, prácticamente despreciable.

Por otro lado deberemos centrarnos en el análisis de la precisión, ya que un tiempo de entrenamiento elevado puede llegar a ser asumible, puesto que sólo se realizará cada vez que se deseen añadir nuevas actividades, lo cual no es habitual. Sin embargo, no es admisible poseer una reducida tasa de aciertos, puesto que afectará en general al uso del sistema. Por tanto, nos centraremos en el análisis del porcentaje de acierto de test, ya que es representativo de la forma en la que se comportará el sistema durante el proceso de reconocimiento. En el caso concreto del sistema sin reducción, el porcentaje de acierto es cercano al 91%, mientras que en los sistemas que emplean reducción de variables, la tasa sube al 99%. Evidentemente esta tasa de clasificación correcta dependerá en gran medida de la calidad del proceso de aprendizaje así como del número de actividades que el sistema sea capaz de reconocer.

Índice	Sin reducción	ACP	Correlación
Epochs	9,33	10,67	6,67
Tiempo (sg)	5,33	0,00	0,33
% acierto entrenamiento	1,00000	1,00000	1,00000
% acierto validación	0,90783	0,97899	0,99648
% acierto test	0,91181	0,99807	0,99599
% acierto global	0,93988	0,99452	0,99851

Tabla 2: Resultado de los parámetros de comparación medidos para cada uno de los métodos de reducción de variables

En resumen, podemos decir que tanto el tiempo de entrenamiento como el porcentaje de clasificación correcta de los sistemas que emplean reducción de variables son significativamente mejores que para el caso del sistema sin método de reducción. Concretamente el porcentaje de aciertos en el conjunto de test que podemos alcanzar mediante los métodos de eliminación de variables son de un 9’53% y un 9’23% con ACP y correlación respectivamente, mayor que empleando la totalidad de variables base.

En esta sección se presentarán una serie de métodos que permitirán crear intervalos que caractericen cada una de las actividades en base a las distintas variables tenidas en cuenta. De esta forma podrá ser posible identificar de una manera rápida y sencilla cada una de las actividades con tan sólo la obtención de las distintas variables, la caracterización de cada una en un intervalo determinado y por último la comparación con los intervalos característicos de una determinada actividad.

La principal ventaja de los métodos basados en la generación de intervalos es que es posible transformar el problema cuantitativo que nos ocupa al dominio cualitativo, simplificando de esa forma los algoritmos y procesos que deben realizarse sobre el conjunto de datos para obtener en el dispositivo la actividad que el sistema estime se está realizando.

Para ello se expondrán tres soluciones al problema de la transformación al dominio cualitativo. La primera de ellas se basa en la generación estática de intervalos gracias al análisis de los datos de las distintas variables del sistema. El principal inconveniente de esta solución es la adaptación a cada usuario, ya que los intervalos dependen en gran medida de las características de las actividades que realice el usuario. A continuación se expondrá un método basado en el anterior, en el que los límites de los intervalos serán generados automáticamente a través de un algoritmo de segmentación. Por otro lado presentamos la aproximación basada en el algoritmo de discretización Ameva (Ameva: an Autonomous Discretization Algorithm, 2009), el cual permite obtener intervalos a partir del conjunto de variables de entrada y maximizando el coeficiente χ^2 de independencia del sistema para cada una de las clases en las que pueden ser clasificados los datos. Este algoritmo, a diferencia del primero, puede ser ejecutado de manera dinámica, por lo que los intervalos pueden ser generados de manera automática tras el aprendizaje realizado por el propio usuario, adaptándose así perfectamente a las características de las actividades llevadas a cabo.

GENERACIÓN ESTÁTICA DE INTERVALOS

La necesidad de generar una serie de intervalos viene impuesta por el paso del ámbito continuo del sistema original, en el que cada una de las variables podía tomar cualquier valor, al ámbito discreto, en el que las variables tan sólo podrán tomar un valor determinado de entre un conjunto de valores específico para cada una de ellas. Aunque más adelante se verán las ventajas de trabajar en el dominio discreto, en esta sección se determinarán una serie de métodos que permiten discretizar los datos provenientes del proceso de obtención de variables a partir de los datos acelerométricos.

El hecho de obtener una serie de intervalos para cada una de las variables, permitirá determinar mediante un criterio de pertenencia a qué intervalo corresponde un determinado dato con tan sólo comprobar los límites de los distintos intervalos y comprobar entre cuáles de dichos límites está incluido el valor. De esta forma, a partir de ese momento el nuevo dato pasará a ser referenciado por la etiqueta del intervalo al que pertenece, realizando así una discretización del mismo. Si aplicamos este procedimiento en cada uno de los datos de las distintas variables, conseguimos crear un conjunto de datos discretizados en un conjunto finito de valores caracterizados por una etiqueta asociada al intervalo.

En este sentido, el primer método que estudiaremos será la generación estática de intervalo, el cual consiste en la determinación de los valores máximos y mínimos que deben tener los intervalos generados para cada una de las variables del sistema, de manera que el intervalo sea lo más

representativo posible de la clase a la que simboliza y que los datos presentes en él posean una menor dispersión. Cuando hablamos de diseminación hacemos referencia a que cada uno de los intervalos debe caracterizar de la manera más precisa posible una actividad para cada variable seleccionada.

Para realizar esta tarea debemos estudiar las diversas variables que posee el sistema, e identificar una serie de patrones dados para los datos que puedan ser decisivos a la hora de determinar una actividad. Estos patrones se buscarán analizando el rango de valores de cada una de las variables del sistema para todas las actividades posibles de ser reconocidas. En las Figura 69, 76, 77, 78 y 79 se presentan una serie de diagramas de cajas que serán los que se tomen como base para la generación estática de los intervalos. Dichos diagramas han sido el resultado de estudiar la consecución de las diversas actividades por un grupo de 30 usuarios distintos. Tras la obtención de estos datos se llevó a cabo un análisis de los mismos procediendo a la eliminación de datos extraños y valores anómalos de la distribución.

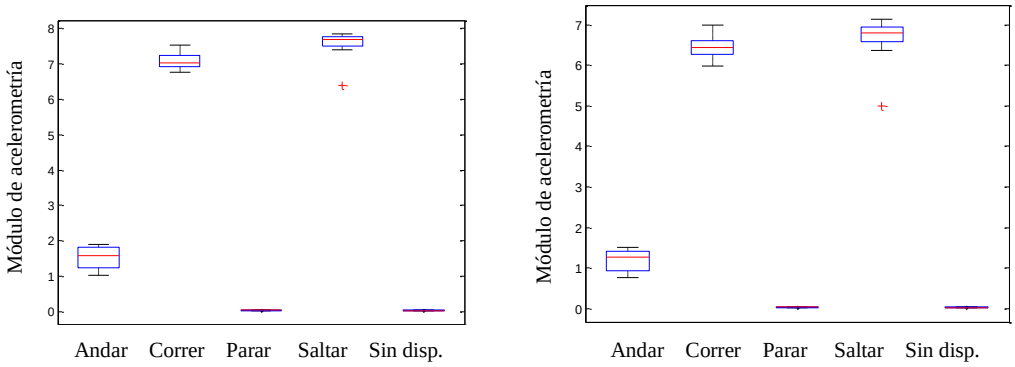


Figura 69: A) Diagrama de cajas de la variable desviación típica B) Diagrama de cajas de la variable desviación media

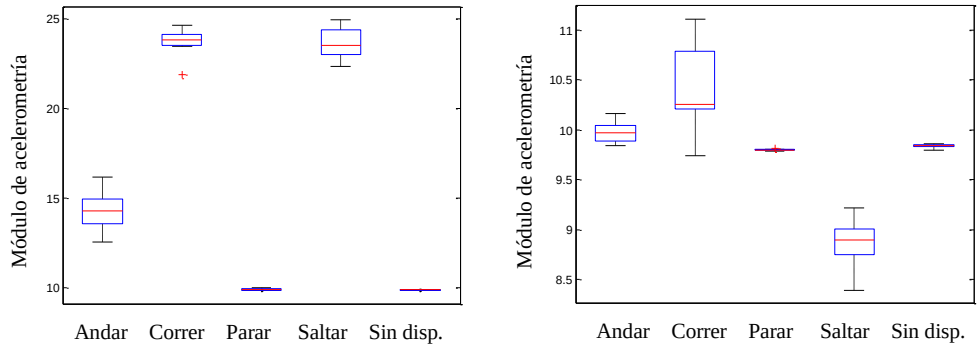


Figura 70: A) Diagrama de cajas de la variable máximo B) Diagrama de cajas de la variable media aritmética

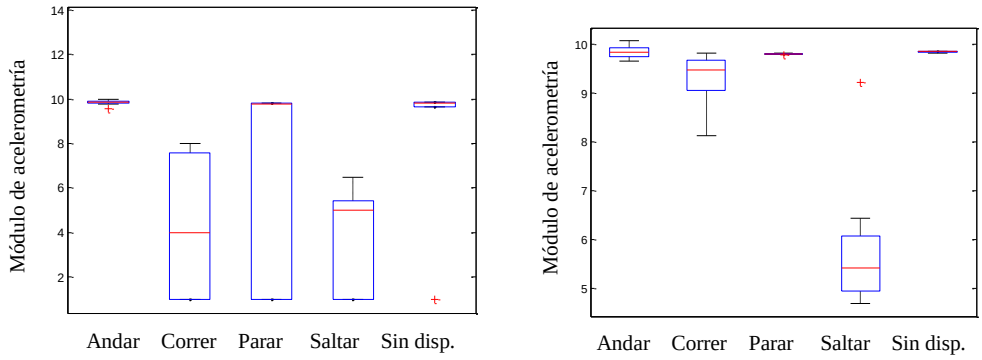


Figura 71: A) Diagrama de cajas de la variable media geométrica B) Diagrama de cajas de la variable mediana

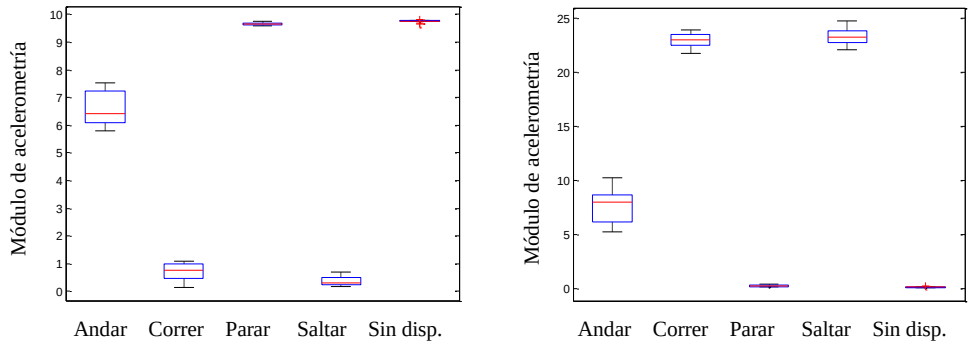


Figura 72: A) Diagrama de cajas de la variable mínimo B) Diagrama de cajas de la variable rango

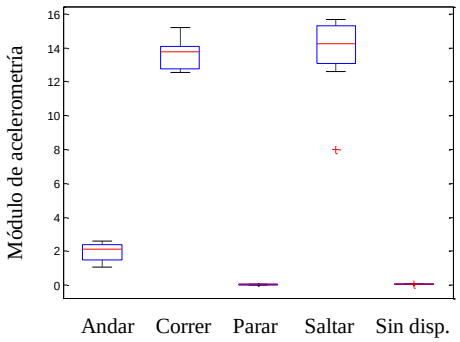


Figura 73: Diagrama de cajas de la variable rango intercuartílico

Mediante el proceso de análisis de los datos e identificando los rangos en los que se presentan los diferentes valores de las variables, es posible reconocer unos límites que permitan dividir el recorrido total de una determinada variable en varios intervalos. El tamaño de los intervalos se calculará en base al número de muestras que contenga cada uno de estos intervalos para los datos presentes en el conjunto de aprendizaje y minimizando en todo momento la heterogeneidad de las actividades asociadas a los datos del intervalo. De esta forma, en el mejor de los casos, un determinado intervalo contendrá un conjunto de muestras cuyas etiquetas se asocien únicamente a una determinada actividad.

En todo momento se buscará que los datos estén repartidos equitativamente entre todos los intervalos creados, de manera que no existan intervalos con excesivas muestras, mientras que otros están completamente vacíos. En este sentido, como se detallará en el apartado Métodos de aprendizaje, es posible incluir un determinado valor de una variable en uno de los intervalos generados, de manera que posteriormente se asociará al intervalo una etiqueta que se corresponderá con la actividad que posea un mayor número de representantes en dicho intervalo.

Por su parte, la generación de los intervalos se hará de manera estática, obteniendo una serie de intervalos *generales* para la clasificación de los distintos estadísticos. De esta forma, aunque como ha sido referido anteriormente, los valores de los intervalos han sido generados a partir de un estudio sobre 30 personas, no es posible asegurar que sean válidos para todos los usuarios. Esto se debe a que una determinada actividad llevada a cabo por cada dos usuarios puede ser radicalmente distinta desde el punto de vista de la acelerometría, entre otras cosas debido a los diferentes patrones a la hora de realizar la actividad (andar, correr, etc.), a la localización donde se encuentre el dispositivo (cadera, brazo, bolsillo, etc.) y a otros factores como la velocidad de las actividades. Todos estos factores hacen que una actividad posea valores distintos para cada estadístico del conjunto de variables.

Una vez expuestas las limitaciones del método actual, debemos decir que este tipo de generación es muy eficiente a la hora de la generación, pues los límites de los intervalos ya están definidos a priori. Por otro lado, la implementación de un sistema que emplee este método como método de discretización es extremadamente sencilla, puesto que tan sólo deberá comprobar la permanencia de un determinado valor de una variable concreta a cada uno de los conjuntos generados, seleccionando aquel conjunto en el que el valor esté entre los límites inferior y superior del intervalo.

El conjunto de intervalos obtenidos mediante esta técnica para cada una de las variables, definen un rango para cada estadístico. Dichos rangos para cada una de los estadísticos empleados por el sistema basados en acelerometría modular se presentan en la Tabla 3. De esta forma, cada uno de los datos del conjunto de aprendizaje podría asociarse a uno de los intervalos de discretización generados estáticamente.

Estadístico	Intervalo 1	Intervalo 2	Intervalo 3	Intervalo 4
Media aritmética	(-∞, 9.3)	[9.3, 9.7)	[9.7, 10.25)	[10.25, ∞)
Mínimo	(-∞, 0.3)	[0.3, 4)	[4, 8)	[8, ∞)
Máximo	(-∞, 12)	[12, 16)	[16, 23)	[23, ∞)
Mediana	(-∞, 5)	[5, 15)	[15, 22)	[22, ∞)
Desviación estándar	(-∞,0.5)	[0.5,4)	[4,7.35)	[7.35, ∞)
Desviación media	(-∞, 1)	[1, 5)	[5, 6.6)	[6.6, ∞)
Rango intercuartílico	(-∞, 0.2)	[0.2, 8)	[8, 14)	[14, ∞)
Media geométrica	(-∞, 4)	[4, 5.7)	[5.7, 7)	[7, ∞)
Amplitud máxima frecuencial	(-∞, 0.8)	[0.8, 2)	[2, 10)	[10, ∞)
Amplitud mínima frecuencial	(-∞, 0.4)	[0.4, 1)	[1, 2.3)	[2.3, ∞)
Frecuencia de la amplitud máxima	(-∞,0.1)	[0.1,3)	[0.3, 1.2)	[1.2, ∞)
Frecuencia de la amplitud mínima	(-∞,0.4)	[0.4, 0.8)	[0.8, 1.3)	[1.3, ∞)

Tabla 3: Intervalos estáticos generados para cada estadístico en un sistema de acelerometría modular

El hecho de que los intervalos hayan sido generados estáticamente hace que este sistema no sea válido en general, además de tener que dedicar un profundo esfuerzo a estudiar los datos obtenidos a partir del conjunto de entrenamiento para generar los intervalos debido al elevado número de usuarios y a la heterogeneidad de los datos. Por tanto, el primer objetivo de mejora de este sistema es generar un método capaz de adaptarse a las actividades del usuario, permitiendo definir una serie de intervalos a partir de las variables del sistema en los datos de entrenamiento. Dicho método será definido a continuación.

GENERACIÓN DINÁMICA DE INTERVALOS

En esta sección se describirá una evolución del método anterior que permite obtener los intervalos de una manera automática a partir de un conjunto de aprendizaje con las diferentes muestras etiquetadas con las actividades correspondientes. Para ello se definirá un algoritmo basado en el reconocimiento de los límites de los diferentes rangos en las variables presentes para las actividades capaces de ser reconocidas por el sistema.

Dados los valores $x_1, x_2, x_3, \dots, x_N$ para una variable X concreta y las etiquetas $L_1, L_2, L_3, \dots, L_P$ correspondientes a las distintas actividades reconocidas por el sistema, se denotan por $C_1, C_2, C_3, \dots, C_P$ los intervalos resultantes de agrupar los valores de la variable X en función del valor de su etiqueta. El límite inferior del intervalo C_i vendrá dado por la siguiente expresión:

$$C_i^m = mín\{x_j | x_j \in L_i \wedge j = 1,2, \dots, N\}$$

Por otra parte, la expresión a partir de la cual se obtiene el límite máximo del intervalo generado para la actividad i-ésima vendrá dado por:

$$C_i^M = máx\{x_j | x_j \in L_i \wedge j = 1,2, \dots, N\}$$

Una vez definidos los límites inferior y superior de los intervalos para cada una de las actividades de un determinado estadísticos, es posible determinar los denominados intervalos de discretización. Estos intervalos permitirán asignar una etiqueta a los nuevos datos provenientes del procesamiento de la información acelerométrica, pudiendo simplificar dicha información obteniendo tan sólo un número discreto de valores para las variables de clasificación.

Los distintos intervalos de discretización se calcularán en base a los intervalos C_i generados anteriormente, para lo cual tendremos en cuenta las siguientes casuísticas que se exponen a continuación.

Si dado un determinado intervalo C_j no existe ningún solape con otro intervalo generado para la misma variable y distinta etiqueta, es decir si se cumple la siguiente restricción,

$$\nexists C_i, \qquad 0 \leq i \leq p \ \wedge \ i \neq j \mid C_j^m \leq C_i^M \leq \ C_j^M \vee C_j^m \leq C_i^m \leq \ C_j^M \vee (C_i^m \leq C_j^m \wedge C_j^M \leq \ C_i^M)$$

el nuevo intervalo poseerá un límite máximo igual al valor medio entre el límite superior del propio intervalo y el límite inferior del siguiente intervalo. En caso de no existir dicho intervalo, el límite

superior se ajusta a ∞. Es decir, si denotamos como $I_j^{e,M}$ al límite inferior del intervalo de clasificación para la actividad j-ésima sobre el estadístico e,

$$I_j^{e,M} = \overline{C_j^M, C_r^m},\ \ C_r = mín\left(|C_t^m - C_j^M|\right)|C_t^m \geq C_j^M$$

Para obtener el límite inferior del intervalo de clasificación, aplicaremos la siguiente condición,

$$I_j^{e,m} = \overline{\overline{C_r^M, C_r^m}},\ \ C_r = máx\left(|C_t^m - C_j^M|\right)|C_t^m \leq C_j^M$$

De esta forma, el intervalo de clasificación generado a partir de un estadístico y una actividad sobre la cual no existe ningún solapamiento de datos procedente de otra actividad, estaría formado por los valores calculados anteriormente,

$$I_j^e = [I_j^{e,m}, I_j^{e,M})$$

Si por el contrario, existe algún solapamiento de actividad tanto en el límite del intervalo superior como en el límite del intervalo inferior, ambos límites vendrían definidos de la manera en la que se expondrá a continuación. De esta forma si se da la siguiente condición,

$$\exists C_i, \qquad 0 \leq i \leq p\ \wedge\ i \neq j \mid C_j^m \leq C_i^M \leq\ C_j^M$$

indicará que existe un conjunto de datos correspondiente a la actividad i-ésima cuyo límite superior entra en conflicto con el límite inferior de la actividad que está siendo procesada (j-ésima). En este caso, se dice que existe un solapamiento parcial inferior y el intervalo inferior de la actividad j se calculará de manera similar a la expuesta anteriormente.

Paralelamente al caso anterior, podría producirse un solapamiento parcial superior de la actividad clasificada. En este caso se cumpliría la siguiente condición,

$$\exists C_i, \qquad 0 \leq i \leq p\ \wedge\ i \neq j \mid C_j^m \leq C_i^m \leq\ C_j^M$$

En tal caso, el límite superior del intervalo se calcularía de manera similar. Sin embargo, podría existe un doble solapamiento, es decir, que tanto el límite superior como el inferior se encuentren solapados por otros intervalos. En dicho caso se debería cumplir la siguiente condición,

$$\exists C_i, \qquad 0 \leq i \leq p\ \wedge\ i \neq j \mid C_j^m \leq C_i^m \leq\ C_j^M \wedge C_j^m \leq C_i^M \leq\ C_j^M$$

En este caso el límite superior del intervalo de clasificación sería igual al límite superior del intervalo que procesado (j-ésimo), así como el límite superior de la clasificación sería igual al límite superior del mismo intervalo,

$$I_j^e = [C_j^m, C_j^M)$$

En base a las expresiones anteriores, el proceso se puede resumir en los siguientes puntos expresados en leguaje natural, los cuales deberán ser aplicados a cada una de las variables y actividades presentes en el sistema:

- Identificar el máximo y el mínimo de la variable para cada una de las actividades.
    - Si no existen solapamientos entre los máximos y los mínimos, los intervalos son calculados de la siguiente manera:
        - El mínimo de la actividad que posee el límite inferior más bajo se establecerá a 0. En otro caso el límite inferior será calculado como la media aritmética entre el valor máximo de la variable para la actividad anterior y el mínimo para la actividad actual, sin tener en cuenta los valores espurios.
        - El máximo de la actividad que posee los valores máximos de todo el conjunto para la variable procesada, se establecerá a ∞. En otro caso el límite superior será calculado como la media aritmética entre el valor mínimo de la variable para la actividad siguiente y el máximo para la actividad actual, sin tener en cuenta los outliers.

En el caso de que no existan solapes entre los intervalos de las diferentes actividades, en la Figura 74 podemos observar la generación de los valores límite. Como se puede observar, dichos límites se encuentran localizados en el valor intermedio del mínimo y el máximo de las distintas actividades excluyendo valores extraños.

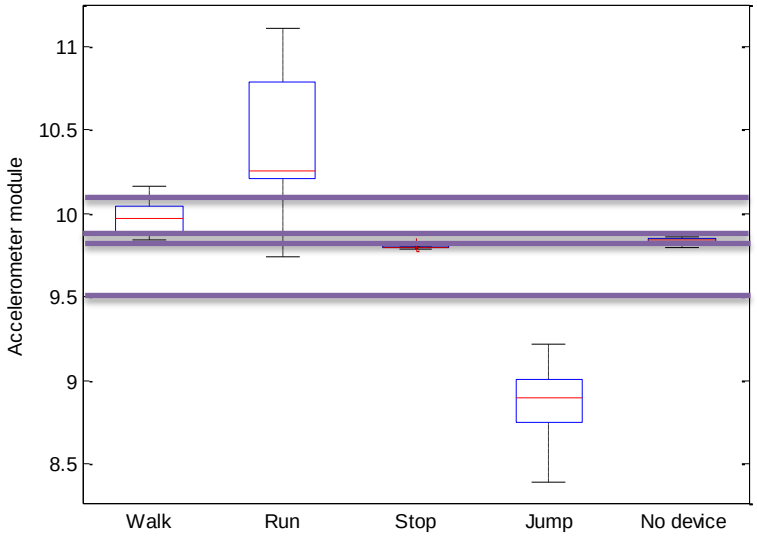


Figura 74: Generación estática de intervalos sin solapamiento

- En cambio, si existen solapamientos entre los valores máximo y mínimo de la variable bajo estudio para las actividades, los intervalos para estos casos se calcularán de la siguiente manera:
    - Si existe una inclusión total de una actividad dentro de otra, se selecciona la actividad cuyos valores de la variable se incluyen en la actividad más general y ese rango será identificado como un intervalo. En este caso los valores de ambas actividades serán englobados en el mismo intervalo de discretización, por lo que se produce una pérdida máxima de información para una de las actividades.
    - Si existe una inclusión parcial, se identifican dos intervalos, el primero se corresponde con aquel cuyo mínimo se establece en la actividad que posee un menor valor para la variable bajo estudio, tal y como se realizaría en el caso

de que no exista solapamiento, y cuyo máximo se establece en el punto correspondiente al máximo del mismo intervalo. El otro intervalo transcurre desde el máximo obtenido anteriormente, hasta el máximo del intervalo superior, que se calcula según lo presentado en el caso de que no exista solapamiento.

En la Figura 75 se representa un caso en el que existe solapamiento entre las actividades *Correr* y *Saltar*. En este caso el solapamiento no es total, por lo que haremos dos intervalos. El primer correspondiente al mínimo de la actividad cuyos valores poseen el mínimo más elevado (*Saltar*), y el límite superior estará localizado en el máximo de la actividad con menor valor para el máximo (*Run).*

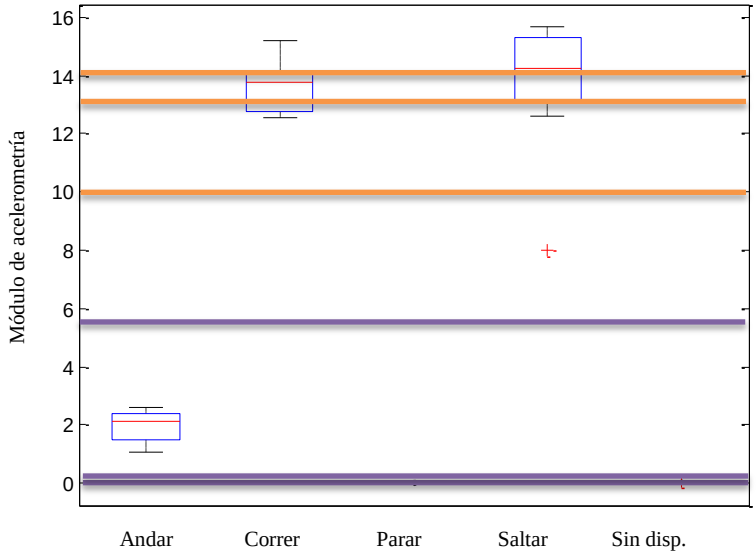


Figura 75: Generación estática de intervalos con solapamiento

Este método aporta una mejora importante al expuesto anteriormente, ya que los intervalos vendrán dados en función del conjunto específico de valores de aprendizaje que reciba el sistema. Por otro lado el coste computacional del método es muy reducido, por lo que puede ser implementado sin ningún problema en el propio dispositivo móvil, sin necesidad de realizar la obtención de los valores límites en el servidor.

Como se comprobará en el apartado de resultados, este método es muy eficaz si el catálogo de actividades elegidas posee un conjunto de estadísticos para los que no existe un nivel de solapamiento elevado. En cambio, si las actividades son fácilmente confundibles a nivel de análisis intervalar de las variables, el método de generación dinámica de intervalos no asegura la optimización de los intervalos generados, pudiendo perder excesiva cantidad de información en el proceso de discretización.

Para solventar el problema de optimización del método actual, se expondrá a continuación una nueva propuesta basada en un novedoso algoritmo de discretización que, posteriormente, adaptaremos en este documento para ser empleado como algoritmo de clasificación.

Ameva (Ameva: An autonomous discretization algorithm, 2009) es un potente algoritmo de discretización, el cual proporciona un reducido número de intervalos que es capaz de obtener a partir de las muestras, lo cual implica que se obtendrán un reducido número de clases tras el proceso de discretización. El hecho de obtener un menor número de clases no sólo ayuda a hacer más intuitivo e interpretable el resultado, sino a minimizar el error de discretización debido al sobreajuste de los datos, así como una menor cantidad de recursos y tiempo para realizar tareas de post-procesamiento sobre los intervalos obtenidos.

El algoritmo Ameva es totalmente autónomo, de forma que es posible obtener los diferentes intervalos directamente a partir de las muestras etiquetadas. Este sistema se basa en la medida estadística Chi cuadrado ($\chi 2$). Existen otras aproximaciones basadas en Chi cuadrado, ChiMerge (Rough set based on modified ChiMerge algorithm and its application, 2003), ChiSplit and Chi2 (An extended Chi2 algorithm for discretization of real value attributes, 2005) (A modified Chi2 algorithm for discretization, 2002). En concreto, el algoritmo ChiMerge ha sido ampliamente usado y referenciado en trabajos de aprendizaje automático. Sin embargo, estos algoritmos necesitan que un experto proporcione algunos parámetros para su mejor uso, dado que deben ser configurados adecuadamente según el conjunto de muestras empleadas para el proceso de discretización.

Por un lado la especificación de los valores de estos parámetros puede ser difícil por múltiples razones: la existencia de ruido importante en los datos, errores en las propias medidas o fallos en los datos como valores anormales; incluso la no disponibilidad de un experto, por ser un problema sobre el que no existe un conocimiento previo suficiente, o que, existiendo el experto, no sea capaz de plasmar de forma numérica su propio conocimiento. Por otro lado, como se indica en (Ilia Mitov, 2009), estos parámetros que normalmente se presentan como una posibilidad de mejorar los resultados suelen, en la mayoría de los casos, ser determinantes y por tanto obligatorios de ser tenidos en consideración, requiriendo una optimización. Es más, se indica que los métodos automáticos para el ajuste de ´estos parámetros tienen, a su vez, parámetros iniciales.

Por otra parte, en los últimos algoritmos supervisados, el algoritmo debería maximizar la interdependencia entre los valores del atributo discreto y las etiquetas de clase, minimizando la pérdida de información debida a la discretización. Es más, la discretización en si misma puede verse como un proceso de descubrimiento de conocimiento en el que los valores críticos dentro de un dominio continuo deben ser identificados.

Volviendo al funcionamiento del algoritmo de Ameva, éste parte de un conjunto de atributos continuos cada cual asociado a una única etiqueta que los representa. A partir de ellos, mediante el procedimiento que será explicado a continuación, el algoritmo determinará una serie de intervalos de forma que cada uno de los atributos quede asociado a un solo intervalo. El procedimiento que sigue el algoritmo se basa en el denominado coeficiente de contingencia, el cual viene definido por la siguiente expresión:

$$\chi^2(k) = N\left(-1 + \sum_{i=1}^{l}\sum_{j=1}^{k}\frac{n_{ij}^2}{n_{i.}n_{.j}}\right)$$

donde n_{ij} representa el número total de valores continuos pertenecientes a la clase *i-ésima* que están contenidos dentro del intervalo *j-ésimo*. La variable *k* representa el número del intervalo que se está procesando y *l* el número total de clases del conjunto que se pretende discretizar.

De la expresión anterior se deduce el coeficiente Ameva que viene dado por la siguiente relación:

$$Ameva(k) = \frac{\chi^2(k)}{k(l-1)}$$

De esta forma, el valor máximo del coeficiente Ameva indica la mejor correlación entre las diferentes clases e intervalos, por lo que el valor máximo del coeficiente se dará cuando todos los atributos asociados a cada clase pertenezcan al mismo intervalo.

El hecho de buscar un óptimo global para el problema de la discretización conlleva una complejidad muy elevada, la cual no puede ser asumida en determinados sistemas, como es el caso de los dispositivos móviles. La ventaja en este aspecto del algoritmo de Ameva es que busca un máximo local en un entorno definido del intervalo, de manera que el error de discretización sea el mínimo. Esta aproximación del problema de discretización lo convierte en un problema con un coste computacional asumible.

Para obtener este mínimo error de discretización, el algoritmo buscará mediante la división recursiva en intervalos más pequeños, aquella división que maximice el coeficiente Ameva, partiendo en principio de un solo intervalo que representa a todos los valores del conjunto de entrenamiento.

En el caso específico del reconocimiento de actividades basado en acelerometría modular, el algoritmo Ameva tomará como etiquetas el conjunto de estadísticos definidos anteriormente (Media aritmética, Mínimo, Máximo, Mediana, Desviación estándar σ, Desviación media, σ^2, Rango, etc.)

Como se ha visto anteriormente, algunas de estas variables pueden ser eliminadas siempre y cuando así se haya determinada tras aplicar alguno de los métodos descritos en la sección de Selección de características singulares. Sin embargo, el atributo *Actividad representada* siempre debe estar presente, ya que es la etiqueta que relaciona los valores estadísticos con la actividad realizada.

A partir de los datos mostrados anteriormente, el algoritmo Ameva será capaz de calcular una serie de intervalos característicos para cada actividad, de manera que el error de clasificación del conjunto de prueba sea el mínimo posible. Sin embargo, existen actividades que pueden contener solapamiento entre las diferentes variables, como es el caso de la media geométrica para las actividades *Run, Stop y Jump*. En este caso, los intervalos ofrecidos por Ameva obviamente, tendrán un error de clasificación máximo, ya que existe solapamiento entre actividades. Sin embargo, dado que la discretización se realizará para todas las variables singulares del sistema, en la mayoría de los casos existirá una combinación de intervalos única para cada actividad.

En las siguientes figuras se representan los intervalos generados mediante el algoritmo Ameva. Para ello se ha seleccionado un conjunto de entrenamiento de 180 datos asociados a 6 actividades distintas. Para las representaciones se ha tomado una muestra de todos los estadísticos generados, concretamente de la media aritmética, mínimo, máximo, mediana, desviación típica y desviación media.

En primer lugar podemos observar como el número de intervalos no es constante, sino que depende de la agrupación de los diferentes datos. Como se comentó anteriormente, el número máximo de intervalos es igual al número total de etiquetas, en este caso 6, aunque el mínimo dependerá de los propios datos. La gran ventaja de este método es que sea cual sea el número de intervalos, el algoritmo maximizará el coeficiente Ameva, lo que asegura una minimización del error de discretización.

Como se puede observar en las siguientes figuras, el número de intervalos generados por Ameva dependerá de la distribución de los datos. Así, para la media se obtendrán 4 intervalos (-∞, 8'9], (8'9, 9'7], (9'7, 9'9] y (9'9, +∞), para el mínimo 3 (-∞, 1'2], (1'2, 8'8] y (8'8, +∞), para el máximo otros 4 intervalos (-∞, 10'1], (10'1, 14], (14, 18'9] y (18'9, +∞), en el caso de la mediana el número de intervalos generados será 3 (-∞, 6'7], (6'7, 8'3] y (8'3, +∞), 4 para la deviación típica (-∞, 2'7], (2'7, 6'3], (6'3, 7'9] y (7'9, +∞) y por último 3 para la desviación media (-∞, 0'2], (0'2, 4'1] y (4'1, +∞)

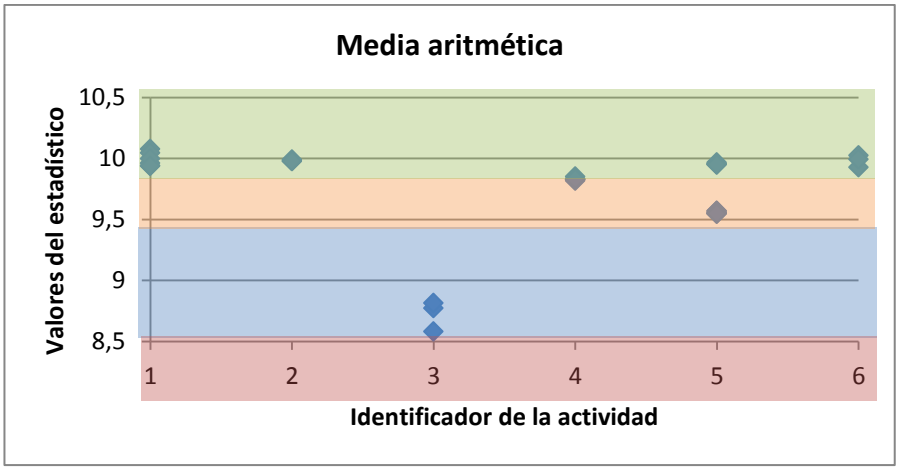


Figura 76: Intervalos generados mediante el algoritmo Ameva para la variable media aritmética con 6 actividades en el conjunto de entrenamiento

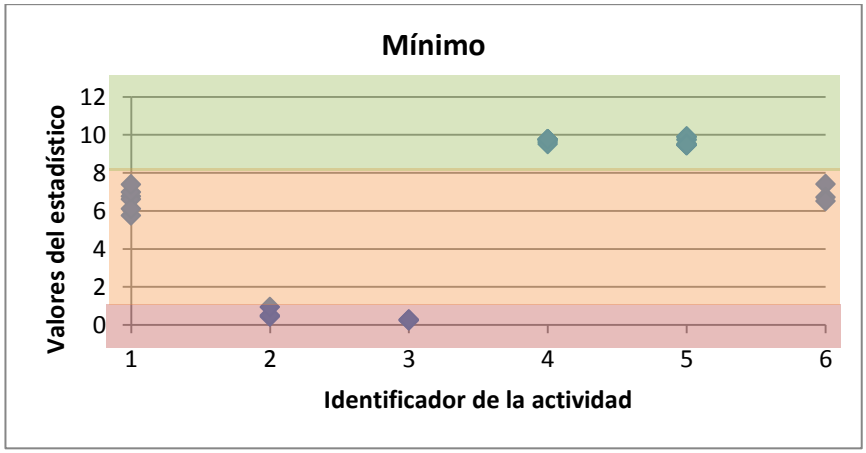


Figura 77: Intervalos generados mediante el algoritmo Ameva para la variable mínimo con 6 actividades en el conjunto de entrenamiento

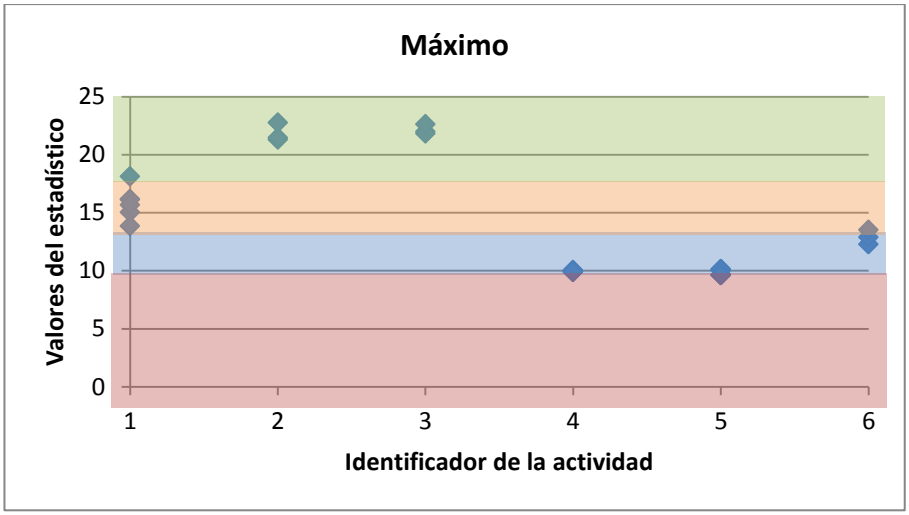


Figura 78: Intervalos generados mediante el algoritmo Ameva para la variable máximo con 6 actividades en el conjunto de entrenamiento

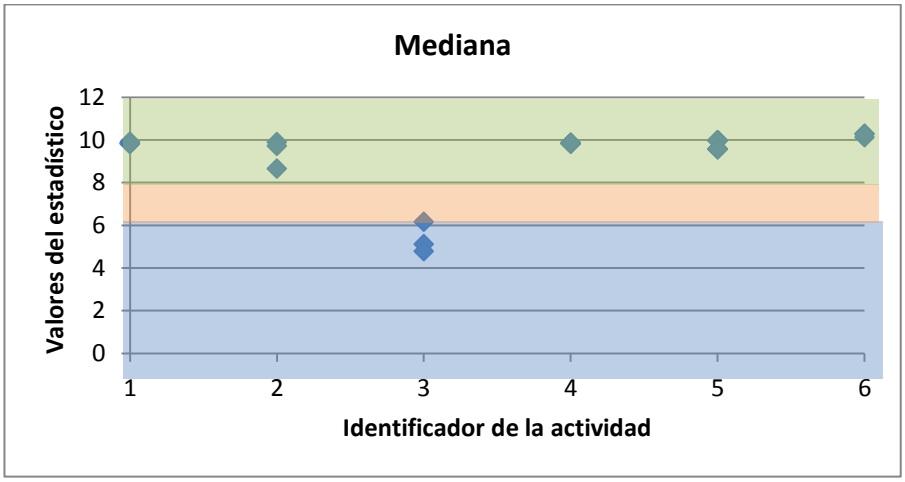


Figura 79: Intervalos generados mediante el algoritmo Ameva para la variable mediana con 6 actividades en el conjunto de entrenamiento

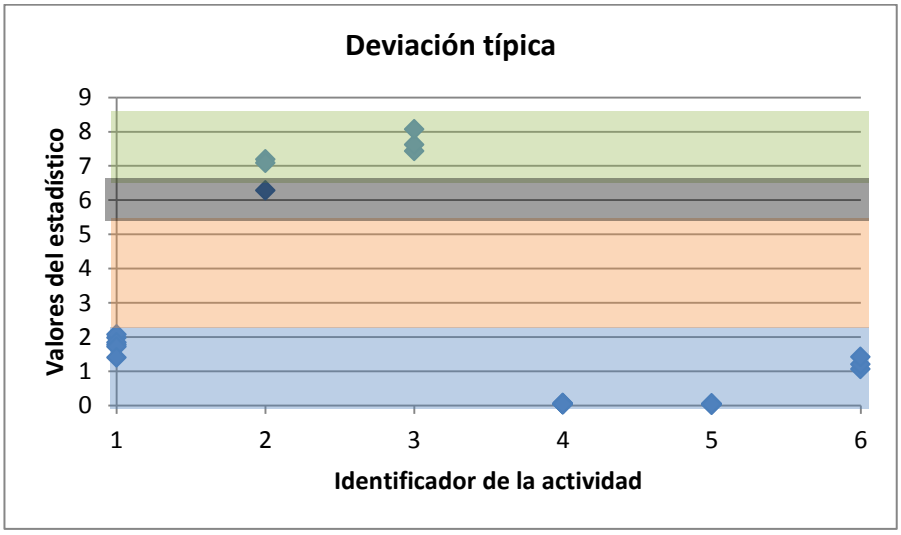


Figura 80: Intervalos generados mediante el algoritmo Ameva para la variable desviación tipica con 6 actividades en el conjunto de entrenamiento

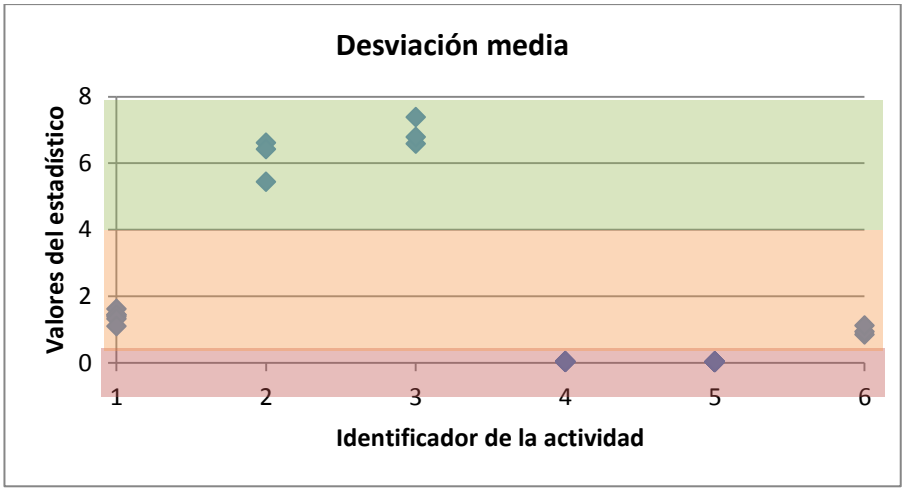


Figura 81: Intervalos generados mediante el algoritmo Ameva para la variable deviación media con 6 actividades en el conjunto de entrenamiento

MÉTODOS DE APRENDIZAJE

En los apartados anteriores han sido expuestas diferentes técnicas para obtener, seleccionar y filtrar datos procedentes del sensor de acelerometría de un determinado dispositivo. Una vez los datos han sido obtenidos y han sido seleccionadas las variables encargadas de representar el sistema es hora de mostrar los métodos empleados para realizar el aprendizaje de las diferentes actividades llevadas a cabo por el usuario. En esta sección se presentarán dos grupos claramente diferenciados de llevar a cabo el aprendizaje; por un lado serán expuestos diferentes métodos para permitir el aprendizaje a partir de una serie de variables discretas, las cuales han sido discretizadas mediante alguno de los métodos presentados en la sección Generación de intervalos y discretización. A continuación se presentará otro conjunto de métodos para llevar a cabo el aprendizaje a partir de variables continuas, las cuales se obtendrán directamente calculando una serie de estadísticos sobre los datos procedentes del acelerómetro.

En general, ambos grupos de métodos de aprendizaje serán llevados a cabo a través de un proceso de supervisión por parte del usuario, lo que implica que se llevará a cabo mediante la colaboración del usuario que está realizando la actividad. De esta forma, el proceso de aprendizaje se podría dividir en dos partes, en primer lugar el sistema le pedirá al usuario que realice una determinada actividad durante al menos un tiempo mínimo de aprendizaje. Durante este tiempo el sistema obtendrá los valores de las diversas variables necesarias para el proceso de clasificación de la actividad. Una vez el usuario ha realizado el proceso de aprendizaje para las actividades que serán reconocidas, se realizará un post-procesamiento de las variables en caso de que fuera necesario. Este procesamiento consistirá en la mayoría de las ocasiones en llevar a cabo el proceso de discretización y/o filtrado. Por último, una vez han sido calculadas las variables útiles del sistema, será posible comenzar el proceso de aprendizaje mediante alguno de los métodos expuestos a continuación.

En última instancia, una vez el sistema ha sido enseñado, comenzará el proceso de reconocimiento, a través del cual el usuario realizará las diferentes actividades y el sistema las clasificará de manera totalmente autónoma. El reconocimiento será tratado también en este apartado como un elemento más del tipo de aprendizaje ya que irremediablemente, el proceso de reconocimiento irá ligado de manera íntima al tipo de aprendizaje realizado.

Dado que todo el proceso de clasificación (y entrenamiento) se llevará a cabo en el propio dispositivo, es indispensable reducir al máximo el coste computacional. Además, el objetivo del diseño del reconocimiento de patrones de actividad es que toda la lógica se realice en el terminal. Por este motivo, desde el aprendizaje hasta la propia clasificación se deben basar en unos pilares estables aunque lo menos pesados posible.

Aunque existen diversas técnicas que pueden llegar a producir un resultado más exacto, como se verá más adelante en los métodos de aprendizaje basados en redes neuronales, son computacionalmente mucho más costosas y además, el aprendizaje es en la mayoría de los casos, impensable llevarlo a cabo en un terminal móvil. El método de aprendizaje y clasificación elegidos para llevar a cabo el método discreto basado en clasificación intervalar están basados en un clasificador de Naive-Bayes.

Para ello, definimos el *vector de rangos* de la variable estadística β como aquel que establece entre sus posiciones un rango de valores en los que pueden agrupar las lecturas tomadas a partir de una ventana temporal.

Dichos vectores pretenden definir un rango para cada una de las variables, y pueden ser obtenido mediante cualquier de los métodos de generación de intervalos expuestos en el apartado Generación de intervalos y discretización. Debemos recordar que las ventanas temporales usadas para analizar la actividad realizada por el usuario son estáticas, es decir, de tamaño constante, y no solapadas. Esto otorga al sistema un menor coste computacional a la hora de realizar el reconocimiento, lo cual es imprescindible para emplear el sistema en dispositivos móviles, cuya potencia de cálculo es menor que en otros equipos.

En el aprendizaje, el usuario debe realizar cada una de las actividades que puede reconocer el sistema. A diferencia de otros trabajos, el número y el tipo de actividades no están determinados a priori, sino que es el propio administrador del sistema o el usuario el que determina las actividades que desea que sean reconocidas. Cuando se realiza el aprendizaje de una actividad concreta, se almacena en los vectores correspondientes a cada uno de los rangos el número de ventanas temporales cuya medida estadística se incluye en dicho rango, así como la actividad que se está desarrollando, a la que denominamos etiqueta. De esta forma, tras finalizar el proceso de recopilación de datos del proceso de aprendizaje, el sistema poseerá una serie de vectores que contendrán en cada posición el número de ventanas temporales de cada actividad que se han detectado en un rango determinado. A la hora de la detección, de la cual hablaremos más adelante, podemos intuir que la actividad elegida irá en función de la frecuencia de ventanas temporales detectadas para un rango y actividad concreta.

De esta forma, el resultado del anterior procesamiento será una matriz con las diferentes ventanas temporales asociadas a cada intervalo, actividad y estadísticos. En la Tabla 4 se puede observar un sencillo ejemplo del contenido de cada uno de estos vectores.

Estadístico	Intervalo	Andando	Saltando	Parado	Subiendo escaleras	Bajando escaleras
Media aritmética	(-∞, 9.3)	10	15	50	15	25
	[9.3, 9.7)	40	30	10	25	15
	[9.7, 10.25)	5	20	6	10	5
	[10.25, ∞)	2	5	4	0	5
Mínimo	(-∞, 0.3)	3	62	0	2	0
	[0.3, 4)	3	5	2	3	5
	[4, 8)	50	3	8	35	12
	[8, ∞)	1	0	60	10	33
Máximo	(-∞, 12)	48	0	65	40	6
	[12, 16)	5	3	3	7	35
	[16, 23)	4	7	2	3	5
	[23, ∞)	1	60	0	0	4
Mediana	(-∞, 5)	3	5	3	8	3
	[5, 15)	51	40	64	37	30
	[15, 22)	2	20	3	6	15
	[22, ∞)	1	5	0	4	2

Tabla 4: Ventanas temporales asociadas a cada intervalo y actividad durante el aprendizaje

Para la determinación de la actividad realizada se han calculado a partir de los datos de acelerometría contenidos en una determinada ventana temporal, los valores estadísticos que se han descrito anteriormente. Una vez transcurrido el tiempo correspondiente a la ventana temporal, tendremos tantos valores estadísticos como variables hayan sido tomadas para representar el sistema, las cuales serán empleadas para determinar la actividad que ha llevado a cabo el usuario durante ese periodo de tiempo. El siguiente paso es acudir a la matriz de frecuencias que se generó durante el proceso de aprendizaje. Existirá una matriz para cada actividad y estará compuesta de N filas, correspondientes a cada una de las medidas estadísticas recopiladas, y P columnas, una para cada rango definido en los vectores expuestos anteriormente.

Definimos la *matriz de frecuencias asociada a la actividad* ∂ como aquella que recoge en la posición [i,j] el número de lecturas que se han recopilado en el proceso de aprendizaje de la actividad ∂ de la variable estadística i en el rango j. Siendo j el rango definido en los vectores de rango. Tras cada proceso de aprendizaje los valores que forman la matriz son normalizados para evitar dependencia con el tiempo de entrenamiento de cada actividad.

Una vez analizada la ventana temporal actual, denotamos $\Omega_t(\partial)$ como el sumatorio del contenido de las posiciones de la matriz de aprendizaje de la actividad ∂ donde el valor obtenido en el análisis de

la ventana temporal t para la medida estadística i coincide con el rango definido en la posición j normalizado.

Por tanto, llegados a este punto asignamos este valor $\Omega_t(\partial)$ como la cuantificación de la posibilidad de que se trate de una actividad u otra en base a la lectura llevada a cabo por el sensor de acelerometría y el estudio de la ventana temporal generada. Un aspecto importante que debemos tener en cuenta es el reducido tiempo de ejecución del algoritmo de probabilidad de detección. Hasta el momento, el algoritmo posee complejidad lineal, por lo que su desarrollo en el terminal móvil es posible y no supondrá un coste computacional excesivo.

Denotamos por $\ddot{\partial}$ *la actividad más probable* en base a las $\Omega_t(\partial)$ obtenidas para cada actividad ∂. De esta forma, tenemos que:

$$\ddot{\partial} = \max_{\partial} \Omega_t(\partial)$$

El reconocimiento de actividad basado en el clasificador de Naive-Bayes junto con el sistema de reducción de ruido basado en el método de Cooley-Tukey ha producido unos resultados relativamente buenos incluso con presencia de altos niveles de ruido. Sin embargo, como veremos más adelante en la sección *Comparativa de los métodos de aprendizaje*, los porcentajes de acierto son bajos cuando el número de actividades a reconocer es elevado. En cambio, como se ha comentado anteriormente, la complejidad del algoritmo es lineal, por lo que se podría emplear como método alternativo en caso de problemas de aprendizaje o reconocimiento de otros métodos más avanzados.

A la hora de llevar a cabo el aprendizaje, debemos tener en cuenta que mediante este método, en contra de lo que se podría pensar, un largo entrenamiento puede afectar de manera negativa al resultado de la clasificación, debido al sobreaprendizaje. Este efecto se da generalmente cuando el número de datos de entrenamiento es muy elevado para determinadas actividades mientras que para otras es excesivamente bajo. Esto produce que las actividades menos entrenadas sean más concretas a la hora de la clasificación mediante el método de Naive-Bayes descrito, mientras que las más entrenadas podrían poseer una mayor probabilidad en rangos de clasificación determinados por los vectores de rango subyacentes al intervalo principal donde se produce la actividad. Este efecto no se produce cuando el usuario realizar un aprendizaje riguroso, sin incluir durante el proceso ninguna actividad ajena a la que se está entrenando. La Figura 82 muestra una comparativa entre dos sistemas, uno de ellos con sobreaprendizaje en la actividad andando respecto al resto de las actividades y el otro sistema con tiempos de entrenamiento semejantes. En dicha ilustración, se puede observar que el número real de veces que se ha llevado a cabo la actividad *andar* está en torno a 15. Cuando el aprendizaje se realiza de manera correcta, el número de veces que el sistema ha reconocido esta actividad es de 12 veces, lo cual está muy cerca de las 15 que se han realizado en realidad. Con el resto de actividades pasa algo semejante, en términos generales, el valor real de número de veces que se ha realizado la actividad está bastante cerca de lo reconocido mediante un aprendizaje normal. Sin embargo, cuando se lleva a cabo un sobreaprendizaje de alguna de las actividades, en este caso *andar*, el número de reconocimientos de esta actividad se dispara, pasando de 12 a 21. Este incremento del número de ventanas identificadas como esta actividad influye en el resto de actividades, disminuyendo el número de reconocimientos.

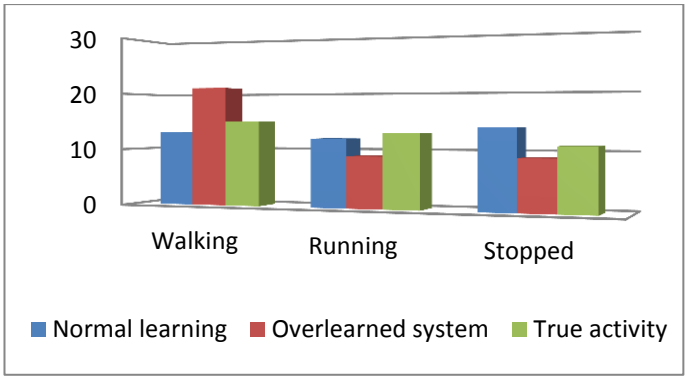


Figura 82: Resultados del parendizaje intervalar sobre tres actividades con sobreaprendizaje

APRENDIZAJE DISCRETO BASADO EN AMEVA

A continuación será expuesto el método de aprendizaje discreto basado en el algoritmo Ameva, el cual será el encargado de generar los diferentes intervalos en el que serán agrupados los datos de las variables de clasificación. De esta forma, el primer paso para realizar el aprendizaje del sistema mediante este algoritmo es aplicar Ameva sobre las variables identificadas en el sistema de reconocimiento de actividades. Recordemos que estas variables serán las resultantes de aplicar uno de los procesos de eliminación de variables correlacionadas expuestos anteriormente.

Una vez aplicado el algoritmo de discretización, el resultado será expresado de forma matricial en la denominada *matriz de clases* y se denota por ϕ. De esta forma, la matriz de clases será un matriz tridimensional, la cual contendrá tantas matrices bidimensionales como variables posea el sistema de reconocimiento. Cada una de las matrices bidimensionales contendrá en cada fila, información referente al identificador de la actividad asociada a la fila, el número de datos etiquetados con esta actividad en el conjunto de aprendizaje y, a continuación, un conjunto de posiciones en las que se almacena el número de datos del conjunto de aprendizaje que corresponden al intervalo *i*-ésimo generado por Ameva, donde *i* viene dado por el índice de la columna concreta donde se almacena el dato.

Para clarificar el concepto de *matriz de clases*, en la Figura 83 representa gráficamente el contenido de las distintas matrices bidimensionales asociadas a cada actividad que componen la propia *matriz de clases*. Según se puede observar en esta figura, se ha llevado a cabo el aprendizaje sobre un total de 6 actividades distintas. Para cada una de las variables, como se ha dicho anteriormente, poseemos una estructura idéntica a la mostrada para la variable *Mediana*.

Actividad	Total	(-∞, 8’82)	[8’82, 9’8)	[9’8, +∞)
Parar	13	1	7	5
Sin disp.	7	1	6	0
Andar	3	3	0	0
Correr	6	0	0	6
Saltar	3	0	0	3
Ciclismo	3	0	0	3

Figura 83: Representación gráfica de una matriz de clases para un conjunto concreto de valores de aprendizaje

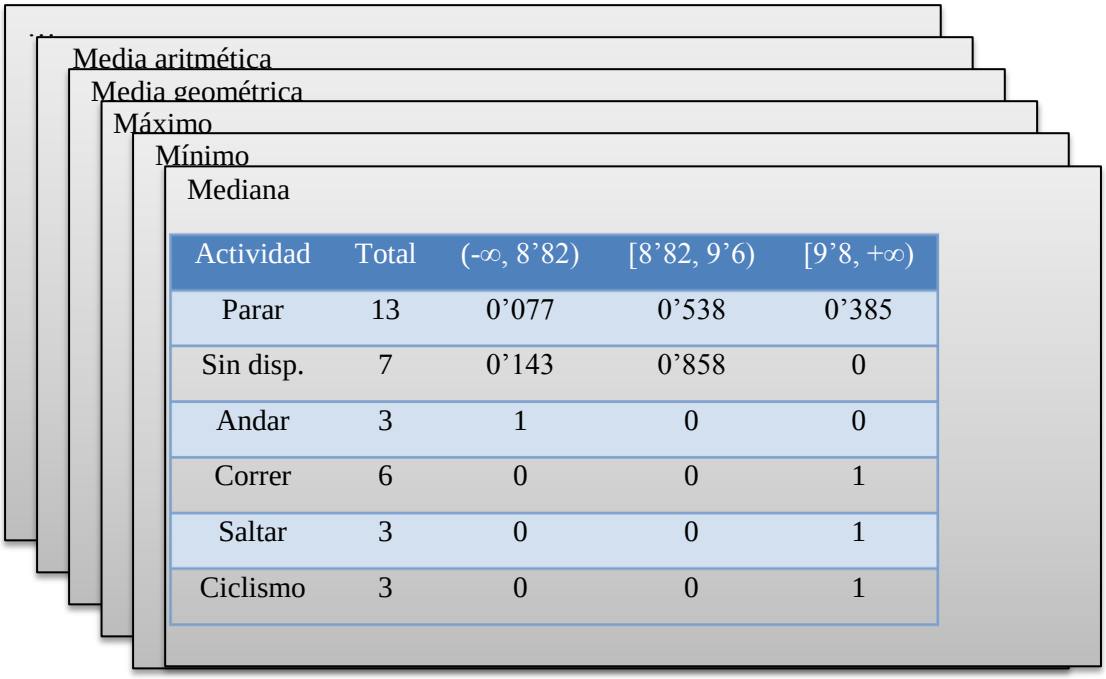
En el caso de la mediana se tiene un total de 3 intervalos, los cuales han sido generados por el algoritmo Ameva a partir de los datos presentes en el conjunto de aprendizaje. Una vez obtenidos los intervalos, se clasificará cada instancia del conjunto de prueba en uno de los intervalos generados, incrementando así en una unidad el número de lecturas contenidas en el intervalo. Esto se hará para todos los datos y se agrupará en función de la etiqueta asociada, es decir, para la actividad que se realizaba cuando fue tomado dicho dato.

De esta forma, la matriz de clases contendrá la frecuencia absoluta de las lecturas agrupadas en cada intervalo para las diferentes actividades y estadísticos. Con el objetivo de lograr un índice de clasificación que determine la probabilidad de que, dado un valor de una determinada variable, pertenezca a una actividad concreta, debemos convertir la tabla anterior de frecuencias en una tabla de frecuencias relativas. En este sentido, cada elemento de la *matriz de frecuencias relativas* denotada por φ, se calculará según la siguiente expresión:

$$\varphi_{e,l,i} = \phi_{e,l,i} \cdot \sum_{\substack{j=1 \\ j \neq i}}^{n} \phi_{e,l,j}$$

donde $\phi_{e,l,i}$ es el número de valores asociados a la actividad *l* en el intervalo *i* para el estadístico *e* en la matriz de clases ϕ. Por tanto, φ al igual que ϕ será una matriz tridimensional.

Aplicado la expresión anterior a la tabla de ejemplo anterior, el resultado sería el que se muestra en la Figura 84.



Actividad	Total	(-∞, 8’82)	[8’82, 9’6)	[9’8, +∞)
Parar	13	0’077	0’538	0’385
Sin disp.	7	0’143	0’858	0
Andar	3	1	0	0
Correr	6	0	0	1
Saltar	3	0	0	1
Ciclismo	3	0	0	1

Figura 84: Matriz de frecuencias relativas de la variable Mediana para un conjunto concreto de valores de aprendizaje

Un valor de $\varphi_{e,l,i}$ cercano a 1 indica que los elementos de la actividad *l* para el estadístico *e* están muy concentrados en el intervalo *i*, mientras que si el valor es cercano a 0, indica que no existen elementos de dicha actividad y estadístico en el intervalo *i*, o bien que el número que existe es muy reducido.

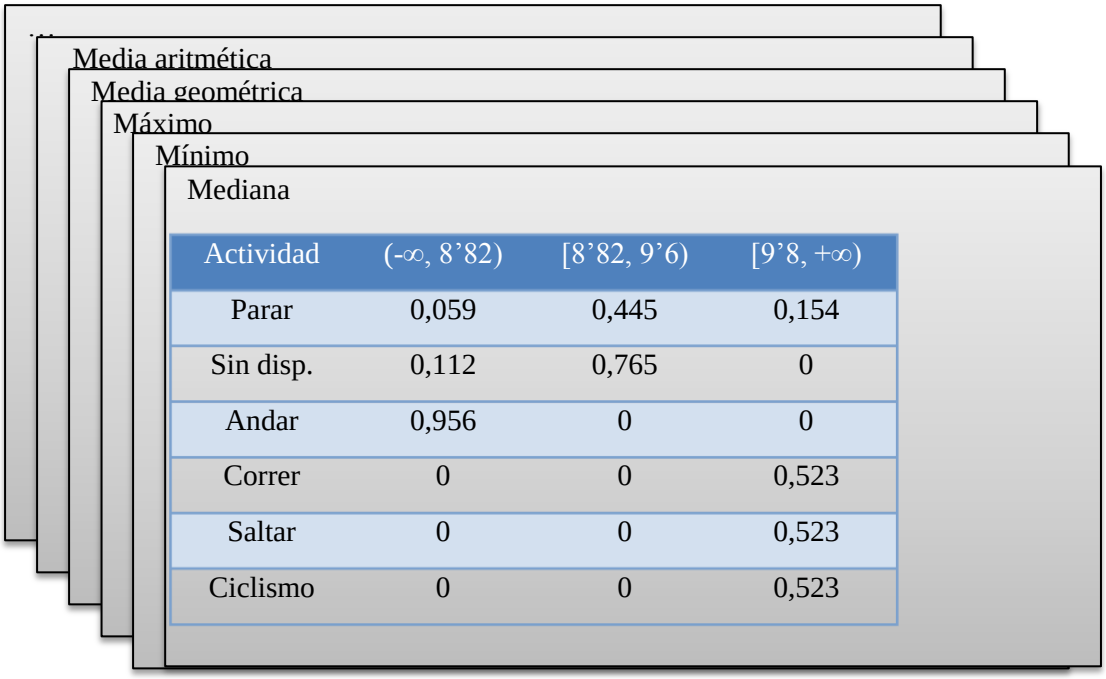
La matriz de frecuencias relativas nos servirá de base para calcular la probabilidad de, dado un dato concreto para una variable determinada, pertenezca a un intervalo. Esta probabilidad se basará en el cálculo de un factor que indique, en función del número de elementos presentes en intervalos distintos al seleccionado para la actividad concreta así como en el número de elementos presentes en el mismo intervalo para otras actividades, la bondad de la clasificación.

Dicho esto, procederemos al cálculo de los diferentes factores que determinarán la bondad de la clasificación de un determinado dato en función del intervalo y la actividad representada. Este *índice de bondad* se calculará en base a la siguiente expresión:

$$P_{e,l,i} = \varphi_{e,l,i} \cdot \frac{1}{n} \sum_{\substack{a=1 \\ a \neq l}}^{n} 1 - \phi_{e,a,i}$$

donde *n* es el número de actividades capaces de ser reconocidas por el sistema, *e* es el estadístico del dato concreto mientras que *l* y *i* son respectivamente la etiqueta y el intervalo para los que se está calculando la bondad.

Si aplicamos la expresión anterior para cada uno de los intervalos de la variable *Mediana*, el resultado es el mostrado en la Figura 85.



Actividad	(-∞, 8’82)	[8’82, 9’6)	[9’8, +∞)
Parar	0,059	0,445	0,154
Sin disp.	0,112	0,765	0
Andar	0,956	0	0
Correr	0	0	0,523
Saltar	0	0	0,523
Ciclismo	0	0	0,523

Figura 85: Factores de bondad de la clasificación para la variable Mediana, un conjunto de 6 actividades y 3 intervalos de discretización

El índice de bondad $P_{e,l,i}$ está normalizado, por lo que emplearemos el valor 0 para denotar que dado un dato concreto que pertenezca al intervalo *i* para la variable *e*, la probabilidad según el método desarrollado de que la actividad llevada a cabo por el usuario sea la actividad asociada a la etiqueta *l* es nula. En cambio, si el valor del índice de bondad es 1, indicara que esta posibilidad es máxima, por lo que con una probabilidad total según el método, la actividad llevada a cabo será la asociada a la etiqueta *l*.

Del modo explicado, mediante el índice de bondad es posible determinar la probabilidad de que dado un valor de una variable concreta, sea reconocida como una actividad determinada. Sin embargo, para que el sistema sea funcional no será posible limitarse a trabajar con una sola variable, sino que deberemos determinar la actividad llevada a cabo en base a todas las actividades del sistema. Con este fin se define el *índice de bondad general* que viene dado por la siguiente expresión:

$$\ddot{P}_l(d) = \frac{1}{m}\sum_{e=1}^{m} P_{e,l,f(d,e)}$$

donde *d* es el dato que se desea clasificar, *m* es el número total de variables del sistema y *f(d,e)* es una función que, dado el dato *d*, obtiene el intervalo concreto al que pertenece para la variable *e*.

De esta forma, $\ddot{P}_l(d)$ proporcionará la probabilidad global de que, dado un dato *d*, la actividad llevada a cabo por el usuario sea la asociada a la etiqueta *l*. Sin embargo, la gran ventaja del método de reconocimiento basado en Ameva es la posibilidad de conocer en todo caso las probabilidades de todas las actividades dado un determinado dato. Actualmente se está empleando esta ventaja para realizar un método capaz de realizar un aprendizaje continuo a través de la retroalimentación del usuario. Dicho sistema se basa en la posibilidad de que, mientras el usuario está realizando las diferentes actividades, se muestre la posibilidad de seleccionar la actividad que está realizando, incluso

aunque no sea la más probable, de manera que al finalizar la sesión de reconocimiento, los datos introducidos serán almacenados como parte del conjunto de entrenamiento.

Además, otra gran ventaja de la visualización de las diferentes probabilidades es que posibilidad la identificación de actividades semejantes, gracias a la igualdad en las probabilidades de detección entre ambas.

APRENDIZAJE CONTINUO BASADO EN REDES NEURONALES

Las redes neuronales proporcionan un método de clasificación muy eficaz. En nuestro caso emplearemos la red neuronal para la clasificación de las actividades en base a los estadísticos obtenidos a partir de las ventanas temporales de las actividades y de la *FFT* generada para cada una de éstas ventanas temporales. En el proceso de reconocimiento de actividades mediante redes neuronales, debemos contemplar dos fases, en primer lugar se deberá realizar el proceso de aprendizaje y en segundo lugar el de reconocimiento.

El perceptrón multicapa (MLP, Multilayer Perceptron) al igual que la mayoría de las redes neuronales artificiales, crean modelos a partir de multiplicadores, sumadores, funciones, etc. El perceptrón multicapa (Rumelhart et al., 1986) es el tipo tradicional de redes neuronales artificiales con aprendizaje supervisado.

Para la primera fase, es decir, la de aprendizaje, se construirá una red neuronal de R entradas, donde R es el número de estadísticos que emplearemos para caracterizar las actividades. El número de salidas será S, es decir, el número de actividades que pueden ser reconocidas. Estas S salidas se activarán de manera independiente, de forma que la salida con un mayor nivel de excitación será la asociada a la actividad que mejor sea representada a partir de los estadísticos. Idealmente, las neuronas de la capa de salida deberían generar S-1 ceros y tan sólo un uno, sin embargo esto no es así debido a la propia naturaleza del aprendizaje para redes neuronales y la ambigüedad de ciertos datos, por lo que todas las salidas se activarán en un cierto grado, aunque será tan sólo una la neurona que mayor valor obtenga a su salida la seleccionada como actividad más probable.

Para realizar el proceso de aprendizaje se empleará una red neuronal con 3 capas (una de entrada, una de salida y una sola capa oculta) y 20 neuronas en la capa oculta. Más adelante será llevado a cabo un estudio a través del cual se determina que un número de neuronas igual a 20 en la capa oculta es adecuado. En general, si este número fuera demasiado elevado podrían presentarse problemas de sobreaprendizaje y especialización de las neuronas, además de incrementarse excesivamente el tiempo de entrenamiento. En cambio, si el número de neuronas en la capa oculta es bajo, cabría el riesgo de crear un cuello de botella en esta capa, influyendo negativamente en los resultados de la capa de salida.

El entrenamiento de estas redes, se basa en la presentación sucesiva y de forma reiterada, de pares de vectores en las capas de entrada y salida (vectores entrada y salida deseada). La red crea un modelo en base al ajuste los pesos en función de los vectores de entrenamiento, de forma que a medida que se pasan estos patrones, para cada vector de entrada la red producirá un valor de salida más similar al vector de salida esperado. Estas redes también se llaman de retropropagación (backpropagation), nombre que viene dado por el tipo de aprendizaje que utilizan.

Una representación de una red con estas características puede verse en la Figura 86.

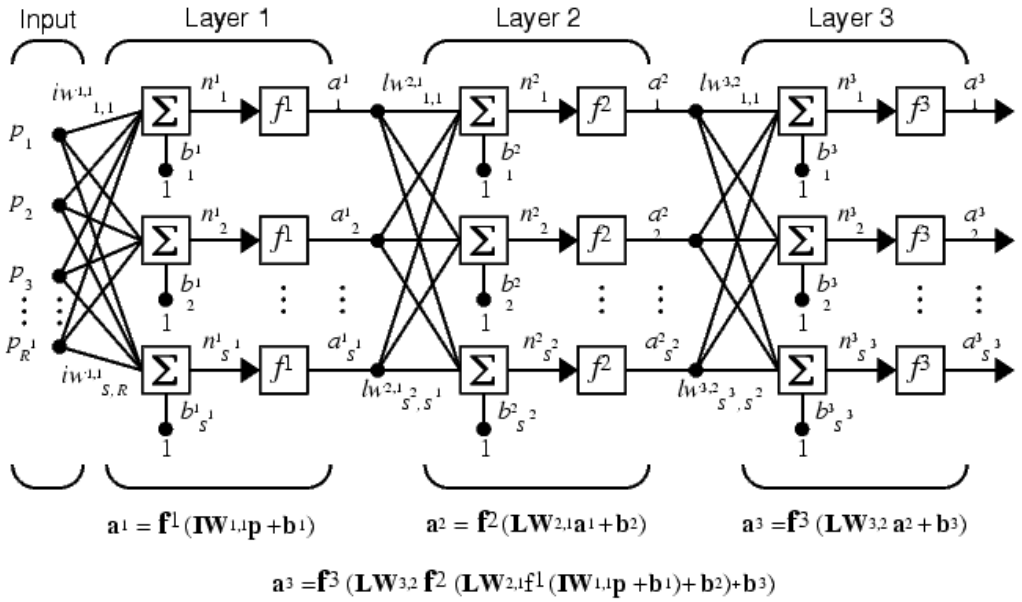


Figura 86: Representación de una red neuronal genérica

Para este caso se han empleado funciones de activación sigmoides para todas las capas de la red. Una función sigmoide es una clase general de funciones suaves que asintóticamente se acercan a un límite inferior a medida que los valores de entrada tienden a menos infinito, y asintóticamente a un límite superior para los valores de entrada que tienden a infinito positivo. La Figura 87 muestra la representación gráfica de la función de activación lógica sigmoide.

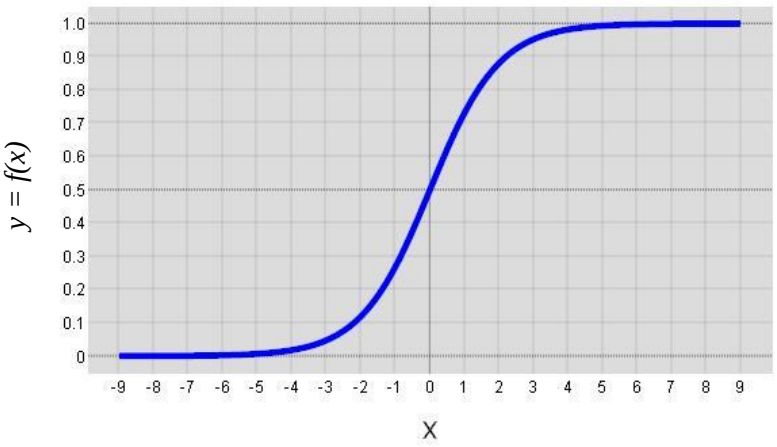


Figura 87: Función de activación sigmoide. Con f(x) = 1/(1+exp(-x))

A la hora de representar la información para ser procesada a través de la red neuronal, se ha optado por un modelo de aprendizaje supervisado. De este modo, los valores de las variables del sistema de reconocimiento seleccionadas anteriormente servirán como datos de aprendizaje de la red. Cada una de las tuplas de variables obtenidas a partir de una determinada ventana temporal será etiquetada con la actividad que se encuentre realizando el usuario en ese momento. Dicha etiqueta nos permitirá posteriormente entrenar la red neuronal obteniendo una serie de parámetros que generen el mínimo error de clasificación.

Se debe tener en cuenta que el conjunto de aprendizaje debe ser lo suficientemente grande para poder realizar un aprendizaje eficaz. Con este fin, todas las actividades que puedan ser reconocidas por el sistema deberán haberse realizado durante el proceso de aprendizaje. Tras realizar un estudio del

número óptimo de ventanas temporales de cada actividad que deben ser generadas durante el proceso de aprendizaje, se ha llegado a la conclusión que deben existir al menos 40 ventanas temporales para cada una de las actividades reconocidas previo a la realización del aprendizaje. El número obtenido no es en absoluto aleatorio, sino que se basa en el equilibrio entre dos factores:

- Precisión del sistema tras el aprendizaje: el sistema de reconocimiento debe ser lo suficientemente preciso a la hora de realizar la clasificación de las actividades que son llevadas a cabo por el usuario. Generalmente se alcanza una mayor precisión cuanto mayor es el tiempo dedicado al aprendizaje, aunque debe ser tenido en cuenta el riesgo de sobreaprendizaje del sistema de reconocimiento.
- Comodidad y eficacia del usuario: en la mayoría de los casos el tiempo dedicado al aprendizaje de las actividades que desea reconocer el sistema es incómodo para el usuario. Esto se debe a que la actividad y los movimientos del usuario deberán estar restringidos a la actividad que se desea aprender. Por este motivo es conveniente que el aprendizaje se lo más corto posible.

En la gráfica mostrada en la Figura 88 se puede observar la curva de precisión del sistema de reconocimiento respecto al número de ventanas temporales sobre el que se llevó a cabo el aprendizaje. Si se presta atención a la curva de aprendizaje en torno a las 40 ventanas temporales, podemos observar que se produce un cambio en la dinámica ascendente de la precisión que, si bien sigue siendo creciente en las sucesivas ventanas temporales, la mejora no es lo suficientemente significativa para justificar un mayor tiempo de aprendizaje del sistema.

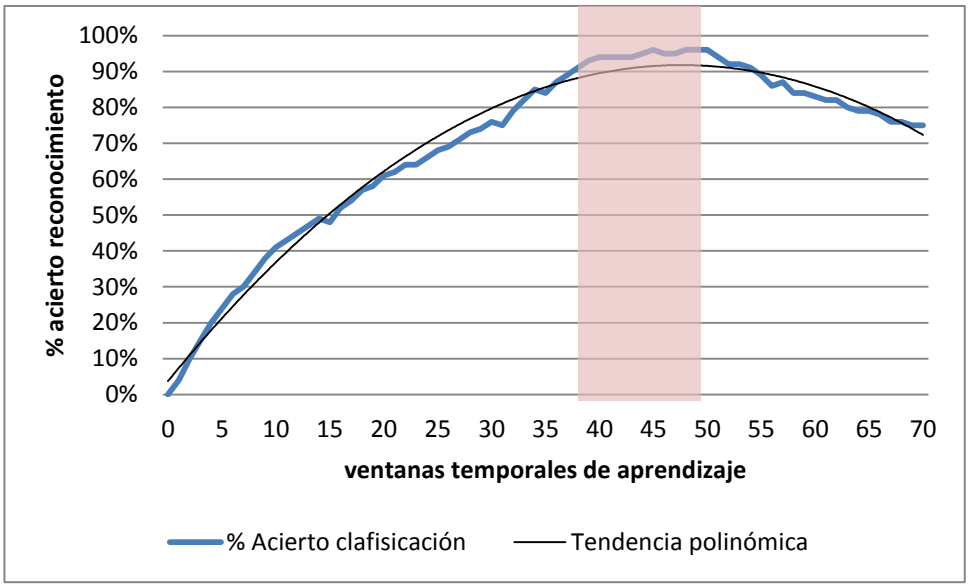


Figura 88: Tendencia-error reconocimiento en función del número de ventanas temporales de aprendizaje

De igual forma, si observamos la tendencia más allá de 50 ventanas temporales, se puede observar que se produce una involución en el porcentaje de acierto del sistema de reconocimiento. Este efecto de aumento del error de clasificación se debe al problema de sobreaprendizaje del sistema. Por este motivo se aconseja que se realice el aprendizaje con un número de al menos 40 ventanas temporales y no más de 50.

Cabe destacar que el estudio realizado anteriormente es muy generalista, de manera que el número de ventanas temporales óptimas para el proceso de aprendizaje variará en función de la actividad concreta. Del mismo modo, podría no ser posible realizar un aprendizaje del tamaño idóneo para determinadas actividades en función del usuario, como por ejemplo para la actividad *saltar* si se trata de un anciano. Obviamente esto influirá en el proceso de reconocimiento, aunque en el apartado de

Comparativa de los métodos de aprendizaje se propondrán soluciones para incrementar la precisión del sistema en estos casos.

Para estudiar el comportamiento de la red neuronal en el proceso de aprendizaje, mostraremos la evolución del error cuadrático medio a medida que se aumenta el número de iteraciones (epochs) del proceso. Como podemos ver en la Figura 89, existe un número de iteraciones para las cuales el error del proceso de validación adquiere un mínimo. Este punto se conoce como punto de mejor rendimiento y se produce para la iteración 17. El motivo de que al aumentar el número de iteraciones aumente también el error de validación es debido al sobreaprendizaje.

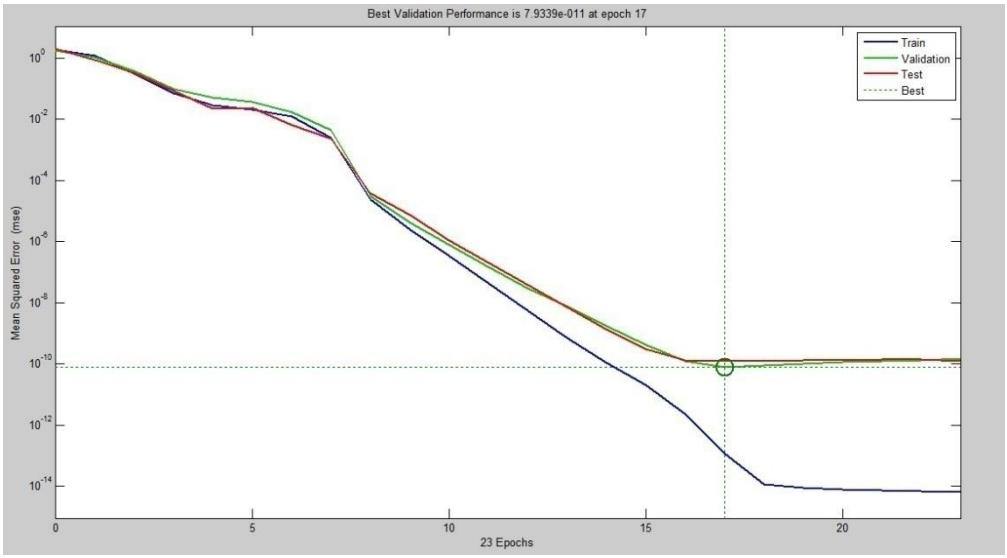


Figura 89: Validación del mejor rendimiento

En cuanto al algoritmo de aprendizaje, se empleara el algoritmo de Levenberg-Mardquardt. El algoritmo de Levenberg-Marquardt (LM) es un algoritmo iterativo de optimización en el cual se modifica el método tradicional de Newton. Las ecuaciones normales,

$$N\Delta=J^TJ\Delta=J^T\varepsilon$$

En la ecuación anterior, J representa el jacobiano de la función, Δ los incrementos de los parámetros y ε el vector de errores residuales resultantes del ajuste mediante el conjunto de datos de prueba. En la expresión anterior, los diferentes elementos son reemplazados por las ecuaciones normales aumentadas:

$$N'\Delta=J^T\varepsilon$$

donde $N'_{ii} = (1 + \lambda_i)N_{ii}$ y $N'_{ii} = N_{ii}$ para i≠j. El valor de λ es inicialmente puesto a algún valor, normalmente $\lambda = 10^{-3}$. Si el valor de Δ obtenido resolviendo las ecuaciones aumentadas conduce a una reducción del error, entonces el incremento es aceptado y λ es dividido por 10 para la siguiente iteración. Por otro lado si el valor de Δ conduce a un aumento del error, entonces λ es multiplicado por 10 y se resuelven de nuevo las ecuaciones normales aumentadas, este proceso continúa hasta que el valor de Δ encontrado da lugar a un decremento del error. Este proceso de resolver repetidamente las ecuaciones normales aumentadas para diferentes valores de λ hasta encontrar un valor aceptable de Δ es lo que constituye una iteración del algoritmo de Levemberg-Mardquardt.

Para observar cómo se comporta la red neuronal con los datos del conjunto de prueba, en la Figura 90 se representa un diagrama de regresión. En él se visualizan cada una de las actividades y la recta de regresión que interpola los resultados obtenidos por la capa de salida de la red. Como es obvio, un mayor coeficiente de regresión significará que los datos se ajustan mejor a la recta de regresión. En la

siguiente figura se puede observar que los valores de los coeficientes están en torno a 0.99, lo cual significa que alrededor del 99% de los datos se han clasificado correctamente. Aunque los resultados serán comentados más adelante.

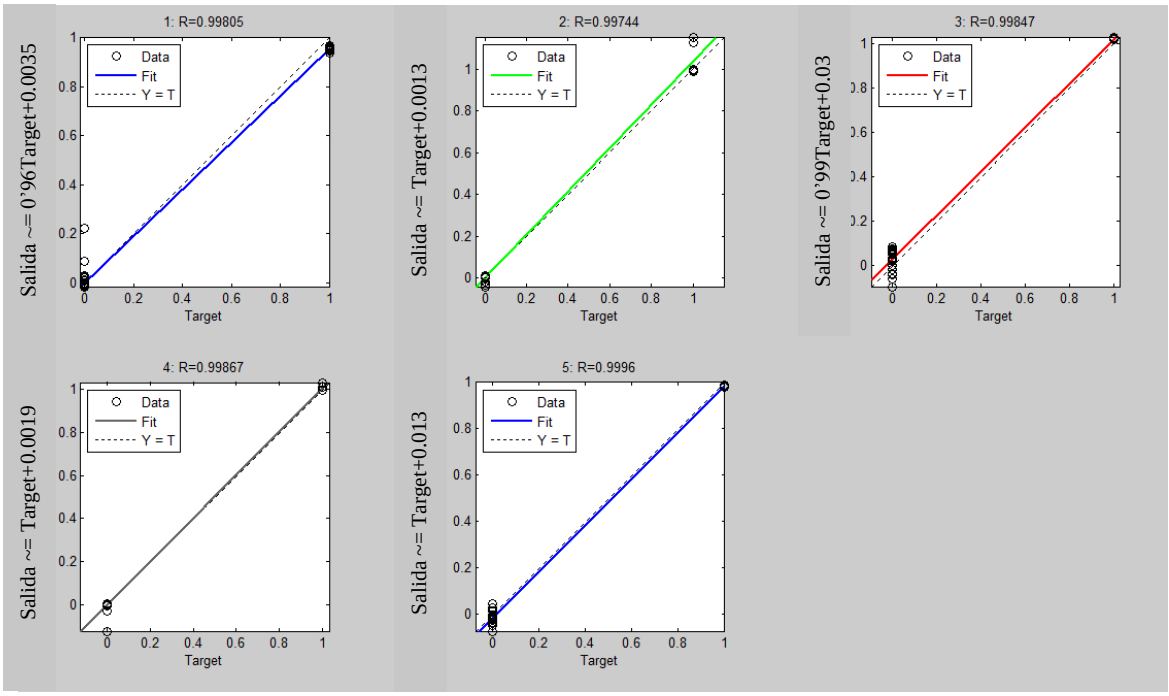


Figura 90: Diagrama de regresión con proceso de eliminación de variables

Los resultados anteriores se han obtenido tras aplicar el proceso de eliminación de variables con un elevado valor de correlación, el cual se explicó en el apartado anterior. Esta reducción de variables hace que el aprendizaje sea más eficaz ya que, por un lado, se reduce el número de variables del sistema y por tanto, la complejidad de los métodos de aprendizaje es más baja. Por otro lado, la eliminación de variables correlacionadas hace que el sistema de aprendizaje sea más robusto frente a valores sensibles de las diferentes variables, lo cual se traduce en que las variables aportarán mayor información al sistema de forma individual, y no mediante correlaciones elevadas que conllevaría la aparición de información duplicada.

Esto se puede observar en la Figura 91, que muestra el diagrama de regresión para el proceso de validación sin eliminar las variables con alto grado de correlación. En ella se ve claramente que el factor de regresión es mucho menor, llegando a un 97%, es decir, un 2% menos que en el caso anterior. Evidentemente, un factor de correlación cercano a 100% indica que la correlación existente entre la clasificación realizada por el sistema de aprendizaje a partir del conjunto de test y las propias etiquetas de los datos que indican la actividad real es máxima, es decir, el error de clasificación es nulo.

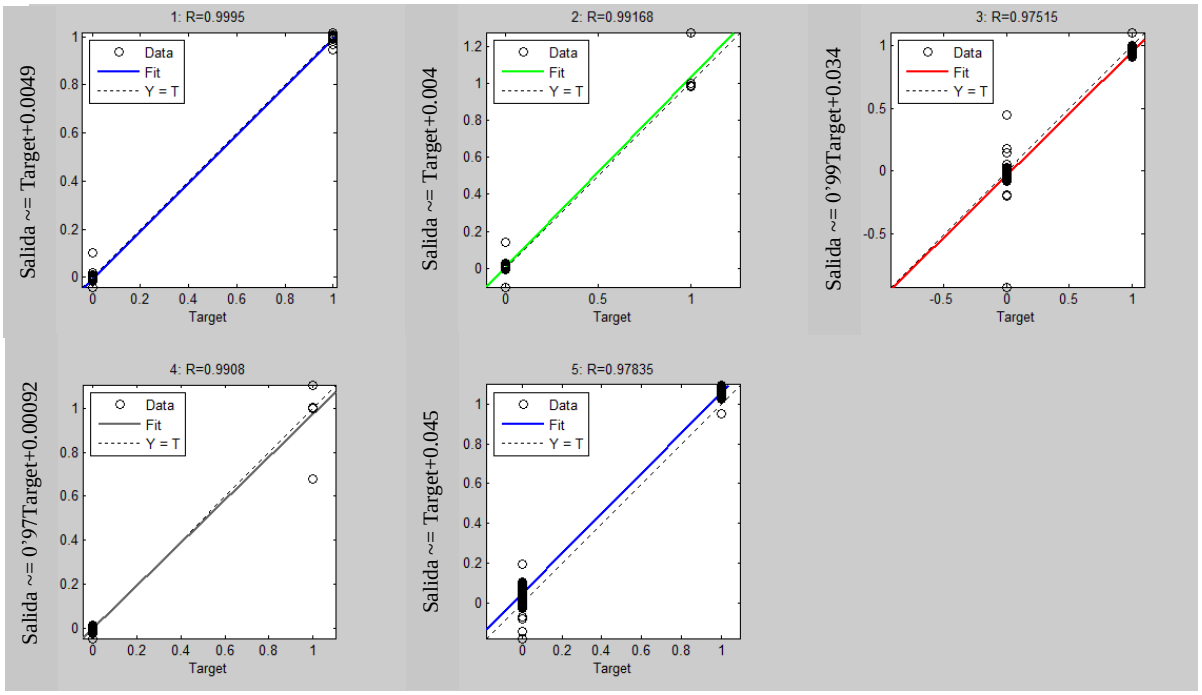


Figura 91: Diagrama de regresión sin procesamiento de eliminación de variables

COMPARATIVA DE LOS MÉTODOS DE APRENDIZAJE

Una vez expuestos los diferentes métodos de aprendizaje/reconocimiento que pueden ser aplicados al sistema de reconocimiento de actividades físicas, en la sección actual se realizará una comparativa atendiendo a una serie de indicadores que nos permitirán seleccionar el método más adecuado en función de las necesidades.

El hecho de dotar al sistema de un catálogo de alternativas de reconocimiento permitirá establecer el método más adecuado dependiendo de los requisitos y recursos que estén disponibles en el momento del aprendizaje. Para realizar esta comparativa comenzaremos dividiendo el conjunto de métodos en dos grupos, aquellos que necesitan un aprendizaje on-line, es decir, el aprendizaje se debe llevar a cabo en el servidor, y aquellos métodos en los cuales el aprendizaje puede realizarse en el propio dispositivo.

Los métodos on-line están compuestos por el sistema de aprendizaje continuo basado en redes neuronales y el de aprendizaje discreto basado en la generación de intervalos mediante el algoritmo Ameva. Si bien el algoritmo Ameva es mucho más eficiente y menos costoso computacionalmente que el basado en redes neuronales, diversas limitaciones técnicas halladas a la hora de realizar una implementación adecuada en el dispositivo móvil, han hecho que este algoritmo también deba ejecutarse en un servidor. Sin embargo, se recoge este punto como una ampliación al sistema de reconocimiento expuesto en esta tesis, por lo que actualmente se está trabajando en la integración del algoritmo Ameva en el propio dispositivo.

Por otro lado, los métodos off-line de aprendizaje estarán compuestos por aquellos basado en el aprendizaje Bayesiano mediante almacenamiento de probabilidades. En general este tipo de aprendizaje es tremendamente rápido y sencillo, por lo que puede ser llevado a cabo sin problemas en el dispositivo. Sin embargo, como analizaremos más adelante, es mucho menos preciso que los métodos on-line, lo que nos llevará a plantearnos bajo qué circunstancias deben ser usados unos u otros.

En cuanto a la arquitectura general que presenta el sistema para la realización del aprendizaje cliente-servidor, se explicará en profundidad en el apartado de Resultados. En dicha sección se hará una descripción pormenorizada de todas las tecnologías empleadas para realizar el sistema de reconocimiento, así como las diferentes capas en las que ha sido dividido el sistema. Por otro lado, no debemos perder de vista que dicho sistema formará parte del middleware para el desarrollo de aplicaciones ubicuas, por lo que se explicará además la estructura para adaptarlo al sistema de *Context Providers* propuesto en el Capítulo 3.

Siguiendo con los métodos de aprendizaje/reconocimiento, la primera cuestión que debemos plantearnos es cuándo emplear un sistema on-line de aprendizaje y cuándo emplear un método off-line. Como se ha comentado, la precisión de los métodos on-line hace que siempre que esté disponible una conexión de datos en el dispositivo, se opte por el método cliente-servidor, mientras que si dicha conexión no está disponible o el usuario no desea hacer uso de ella, se empleará los métodos locales.

Aunque se ha realizado una clasificación en dos grandes grupos, alguno de los métodos on-line podrían encuadrarse en un tercer conjunto, en el cual el aprendizaje debe ser llevado a cabo en el servidor, mientras que el reconocimiento se llevará a cabo en el propio dispositivo cliente. Este es el caso del aprendizaje neuronal y el intervalar basado en Ameva. La complejidad de ambos métodos radica en el aprendizaje, por lo que se ha decidido que debe ser llevado a cabo en el servidor. Sin embargo, el reconocimiento basado en ambos métodos es relativamente poco costoso, ya que en el caso del aprendizaje intervalar de Ameva, tan sólo es necesario comprobar el intervalo al que pertenece un determinado valor de las variables y obtener el máximo, que será aquel que represente a la actividad más probable. Por su parte, el reconocimiento basado en redes neuronales es algo más complejo, ya que se basará en la aplicación de una serie de pesos asociados a los valores de las variables y la aplicación de una función de activación sobre cada uno, de manera que de la complejidad de la función de activación y del tamaño de la red dependerá la complejidad general del método. En este sentido, el tráfico de datos a través de la red para enviar los datos al servidor encargado de realizar el aprendizaje, tan sólo deberá ser realizado una vez, tras la consecución de todos los procesos de aprendizaje para cada una de las actividades.

Dicho esto, en esta sección de comparativas nos centraremos en los siguientes aspectos para determinar la mejor técnica de aprendizaje o la más adecuada:

- *Complejidad del aprendizaje*. Analizaremos la complejidad temporal y a nivel de recursos de los diferentes métodos de aprendizaje expuestos anteriormente. También tendremos en cuenta la posibilidad de realizar el aprendizaje en el propio dispositivo, evitando de esta forma la necesidad de poseer una conexión de datos en el momento del aprendizaje.
- *Complejidad del reconocimiento*. Tanto o más importante que el propio aprendizaje resulta estudiar la complejidad del proceso de reconocimiento. Mientras que el aprendizaje tan sólo deberá ser realizado una vez al finalizar la supervisión de las diferentes actividades, el proceso de reconocimiento se realizará de manera continuada durante todo el tiempo de ejecución del módulo de reconocimiento de actividades. Por este motivo, deberemos identificar los principales problemas presentes en cada uno de los métodos y prestar atención a los recursos disponibles y necesarios para cada uno de ellos.
- *Tráfico de red requerido*. En ciertos métodos de aprendizaje, dicho proceso debe realizarse en el servidor, por lo que será necesario establecer un canal de comunicación entre éste y el dispositivo desde el que se obtienen dichos datos. Este canal de comunicación se basará en una conexión de datos a través de la cual realizar el intercambio de información. En general, las prestaciones de las conexiones de datos de los dispositivos móviles suelen ser inferiores a

los equipos de sobremesa o portátiles, por lo que será necesario realizar un estudio pormenorizado del volumen de datos que será necesario intercambiar para cada uno de los métodos. Estudiando esta cantidad de datos será posible determinar bajo qué circunstancias será posible emplear los distintos métodos.

- *Precisión del reconocimiento*. El objetivo de los métodos de reconocimiento de actividades desarrollados fue generar un método eficiente y preciso para obtener las actividades físicas llevadas a cabo por un usuario a lo largo de un periodo de tiempo. En esta sección de comparación expondremos, la diferente precisión que es posible alcanzar con cada uno de los métodos.
- *Influencia del número de actividades reconocidas*. El hecho de aumentar el número de actividades reconocidas puede suponer un problema para mantener la estructura de reconocimiento según el método empleado. En este caso serán vistas las repercusiones que tiene en cada uno de los métodos el añadir una nueva actividad al conjunto de actividades reconocibles.

Todas las comparativas se han llevado a cabo mediante un entrenamiento de 15 minutos por cada una de las actividades reconocidas y sobre un conjunto de 30 individuos. Para evitar problemas relacionados con el sesgo muestral sobre un determinado rango de edad, los 30 individuos han sido elegidos cuidadosamente imponiendo una igualdad tanto en edad como en sexo de los usuarios. De los 30 usuarios seleccionados, la muestra se distribuye según lo recogido en la Tabla 5.

Sexo/Edad	(15-25]	(25-35]	(35-45]	(45-55]	(55-65]
Femenino	3	4	3	2	2
Masculino	4	4	3	3	2

TABLA 5: DISTRIBUCIÓN DE USUARIOS PARA LAS PRUEBAS DEL SISTEMA

En base a la anterior distribución de usuarios, se han realizado los diversos tipos de prueba que han sido llevadas a cabo y cuyos resultados se exponen a continuación.

Complejidad de los métodos de aprendizaje

Para poder llevar a cabo un sistema eficiente y dinámico, debemos conseguir minimizar el tiempo necesario para llevar a cabo el aprendizaje. Aunque éste tan sólo deberá ser ejecutado cada vez que se desea añadir una nueva actividad al conjunto de actividades reconocidas, sea necesario completar el entrenamiento sobre ciertas actividades o el usuario decida ampliar el aprendizaje con nuevos datos, en ciertas ocasiones un tiempo de entrenamiento excesivo puede suponer un problema sensible que puede influir en la calidad de uso del sistema.

En este sentido, a lo largo de este punto, estudiaremos los diferentes métodos de detección de actividades desde el punto de vista del rendimiento del aprendizaje. La relación calidad-tiempo del aprendizaje no será necesariamente directa, como comprobaremos a lo largo del proceso de pruebas, sino que un mayor tiempo de aprendizaje podría suponer una menor precisión en el reconocimiento de las diversas actividades.

Con el fin de independizar la comparativa de los métodos de aprendizaje y las diversas herramientas para reducción de variables, en todos los resultados de esta sección se ha empleado el mismo método de reducción, basado en el ACP reduciendo de 42 variables descriptivas del sistema a

27. De esta forma, el conjunto de 27 variables pasará a ser los datos de entrada para el proceso de aprendizaje.

Comenzaremos observando los tiempos para el método basado en Ameva. Este método posee una característica singular, que consiste en que se aplicará de manera repetida sobre cada uno de las variables del sistema, con el fin de discretizar dichos valores. Por tanto, el rendimiento de este método estará claramente influenciado por el número de variables del sistema, de forma que cuanto menor sea este número, menor será el tiempo de aprendizaje.

En la Figura 92 se puede observar el tiempo de aprendizaje para cada una de las variables del sistema. En general, el tiempo está en torno a 1.2 segundos, excepto para la primera variable, debido al coste de inicialización de los diferentes atributos y métodos necesario para llevar a cabo el sistema de aprendizaje.

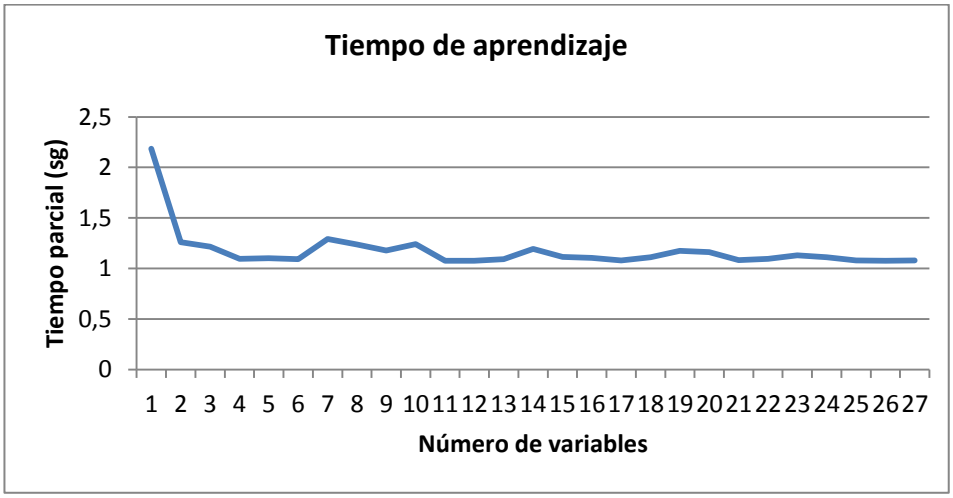


Figura 92: Tiempo de procesamiento en milisegundos del método de aprendizaje basado en ameva en función del número de variables

Para tener una idea más clara acerca del tiempo necesario para llevar a cabo el aprendizaje basado en Ameva, en la Figura 93 se muestra el tiempo acumulado resultante de obtener el tiempo total de aprendizaje en función del número de variables. La tendencia lineal creciente con pendiente constante viene dada por el tiempo constante de aplicación del algoritmo de Ameva sobre el conjunto de datos asociado a cada una de las variables. En este análisis concretamente se puede observar el impacto que tendría la utilización de un método de reducción de variables respecto al tiempo de aprendizaje del sistema.

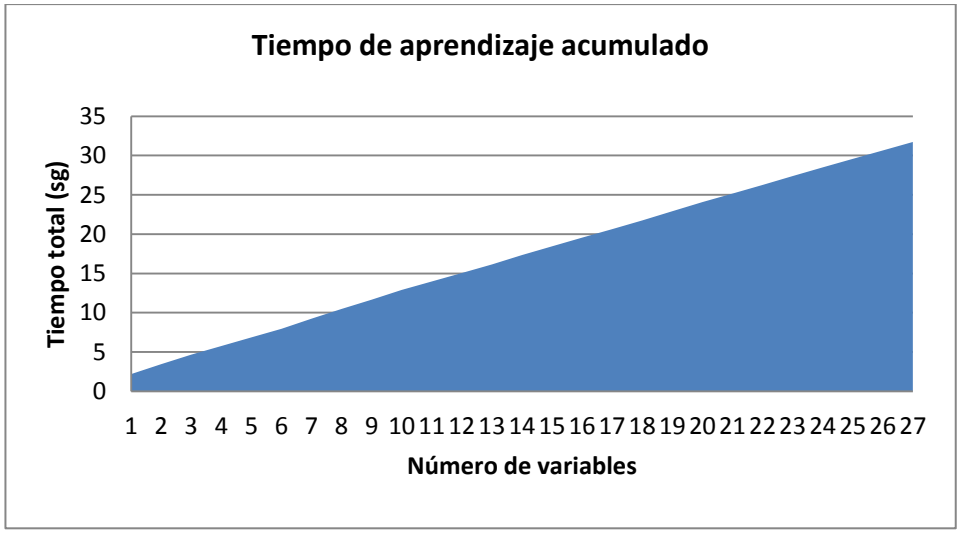


Figura 93: Tiempo de aprendizaje en función del número de variables

Continuando con el análisis temporal, estudiaremos en este caso el tiempo de ejecución del algoritmo de aprendizaje para el sistema basado en redes neuronales. En la Figura 94 se puede observar el comportamiento irregular que presenta el sistema de aprendizaje basado en redes neuronales en función del número de variables. Dichas irregularidades se deben a la inexistencia de mínimos en el error de clasificación. El hecho de que el conjunto de datos sea reducido o las variables estén correlacionadas, puede afectar en general de manera significativa al método basado en RNA. Esto hace que se deba prestar especial atención al estudio de las variables elegidas para representar al sistema.

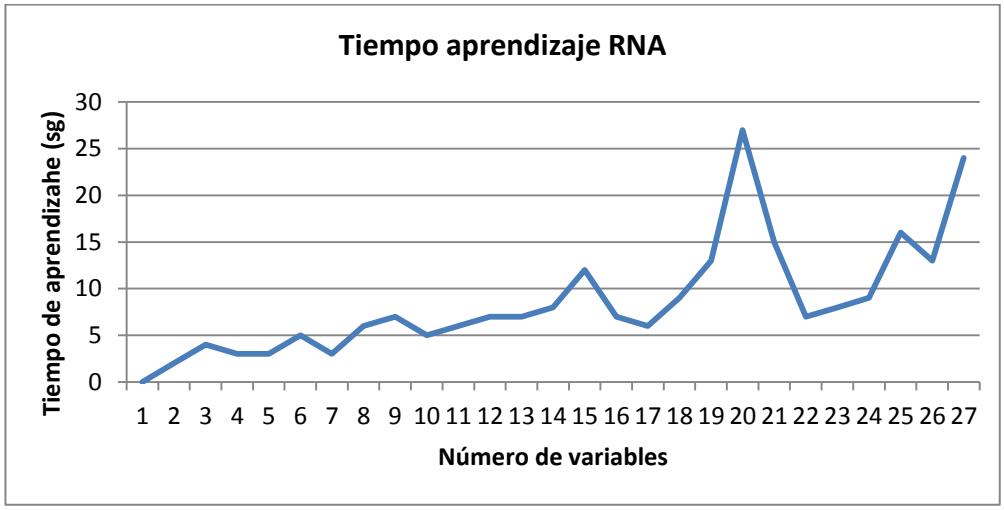


Figura 94: Tiempo de aprendizaje para el método basado en redes neuronales en función del número de variables

En la figura anterior se muestra que los datos no siguen una progresión lineal, como era el caso del método basado en Ameva, por lo que no es posible determinar una clara relación entre el número de variables y el tiempo de aprendizaje. Por este motivo, aunque pueda parecer que el tiempo de aprendizaje acumulado de Ameva es superior al tiempo de aprendizaje necesario por el método de RNA, no podemos asegurar que esto sea así en todo caso. De hecho, durante la realización de los distintos casos de aprendizaje para los datos almacenados durante esta comparativa, se han dado casos en los que el método de aprendizaje basado en RNA no ha encontrado una ponderación de las

diferentes conexiones entre neuronas capaz de obtener un error de clasificación inferior al umbral mínimo establecido. Esto hace que, aunque preciso y en cierta medida eficiente, sea un método poco previsible en cuanto al tiempo necesario para su ejecución, pudiendo ocasionar en ciertos casos un bloqueo excesivo del tiempo de proceso.

Por último, analizaremos el tiempo de ejecución asociado al método de generación de intervalos automáticos mediante la identificación de valores límite. La gran ventaja de este método respecto a los anteriores es la posibilidad de ser implementado en el propio dispositivo, evitando de esta forma la necesidad de conexión de datos para realizar el aprendizaje. Sin embargo, esta ventaja se convierte en un problema a la hora de llevar a cabo el aprendizaje, ya que debemos asegurar que éste se realice en el menor tiempo posible para que sea totalmente transparente para el usuario.

Los procesadores de los dispositivos móviles de última generación, aunque de menor potencia que los de los equipos de escritorio, poseen una elevada capacidad de cálculo. Este hecho unido a la simplicidad del método de generación de intervalos propuesto, hace que el aprendizaje se realiza de manera muy eficiente. En la Figura 95 se representa el tiempo de aprendizaje en función del número de variables para el método bajo estudio. Como se puede observar, el tiempo de aprendizaje es bastante eficiente, a pesar de que los tiempos se han obtenido a partir de la ejecución en el propio dispositivo. Al igual que para el caso del aprendizaje mediante Ameva, la tendencia temporal es ascendente en función del número de variables, por lo que sería interesante aplicar previamente al aprendizaje un método de reducción de variables, como es el caso expuesto en esta sección, o bien reducir el número de variables de las 48 iniciales.

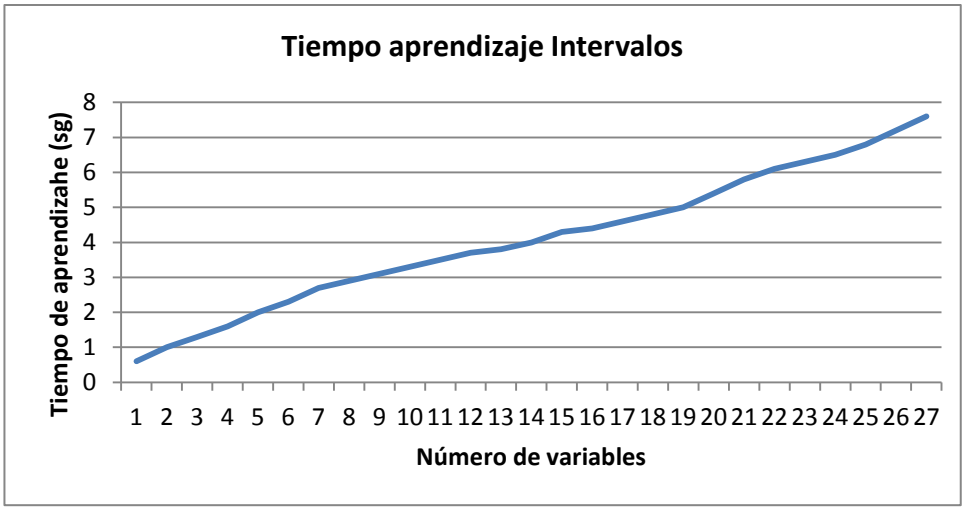


Figura 95: Tiempo de aprendizaje para el método de generación dinámica de intervalos en función del número de variables

Para un total de 27 variables, el tiempo de ejecución de este método está en torno a 8 segundos, lo cual asegura la eficiencia del sistema. Además, tras realizar una batería de 50 pruebas sobre el método, no se han detectado anomalías temporales, a diferencia de lo que ocurría en ciertas ocasiones cuando era aplicado el método de aprendizaje basado en RNA. Esto hace que el sistema sea muy eficiente en todos los casos.

Un aspecto debe ser tenido en cuenta a la hora de desarrollar cualquier método que sea ejecutado en el dispositivo, es el consumo energético resultante del procesamiento realizado. En este caso, el procesamiento es breve y poco intensivo, además de poco frecuente, por lo que podemos asegurar que

realizando un uso adecuado del método de aprendizaje, el tiempo útil de la batería no se verá reducido debido al coste energético del proceso.

A modo de resumen de esta sección, en la Figura 96 se puede observar una comparativa directa entre los distintos métodos de aprendizaje en función del tiempo necesario para llevarlos a cabo tanto en el dispositivo como en el servidor, en función del método seleccionado. Respecto al tiempo de aprendizaje mediante el método de intervalos dinámicos, decir que posee un reducido tiempo de aprendizaje, el menor de todos los métodos, y la ventaja de ser ejecutado en el propio dispositivo, aunque el principal inconveniente será analizado posteriormente cuando se muestre la precisión de los diferentes métodos.

Seguidamente se presenta el tiempo de aprendizaje basado en redes neuronales, el cual es mayor que el anterior, aunque se producen ciertas irregularidades que se describieron anteriormente. Concretamente para el caso de 20 variables, el tiempo de entrenamiento asciende a 27 segundos, por encima de todos los métodos estudiados. Esto hace que sea necesario prestar mucha atención a los datos de entrenamiento debido a la posibilidad de obtener tiempos de aprendizaje excesivos.

Por último se presenta el método basado en Ameva que, aunque con un tiempo de aprendizaje mayor, se comporta de una manera muy regular en función del número de variables. De esta forma, si es posible reducir de manera significativa el número de variables a través de los métodos de reducción expuestos en la sección anterior, se podría conseguir un entrenamiento eficaz en un tiempo reducido.

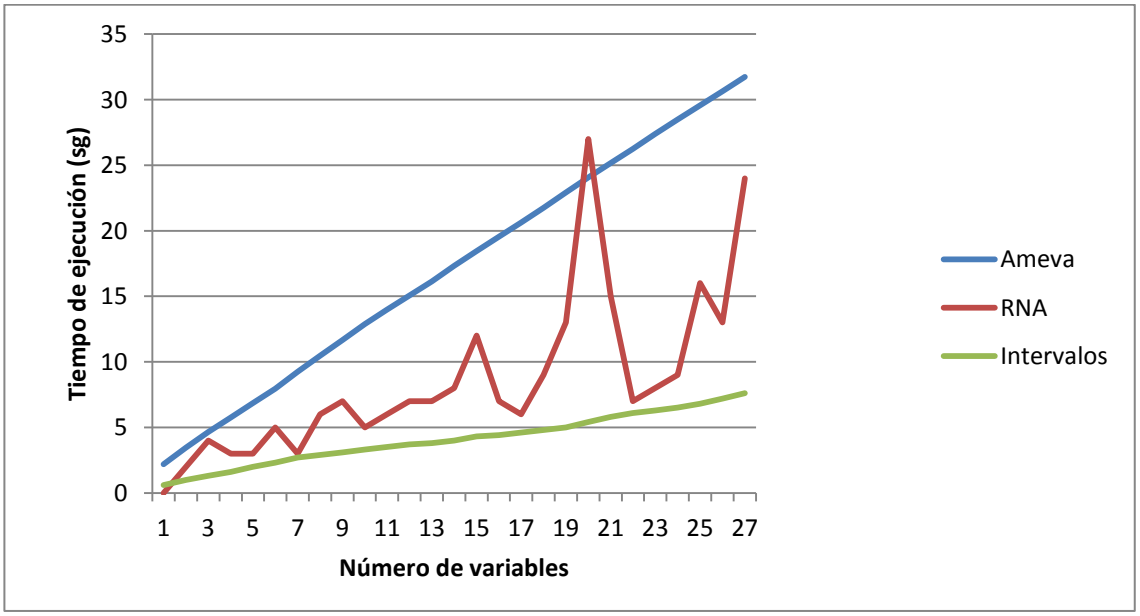


Figura 96: Comparativa de los tiempos de ejecución para los diferentes métodos de aprendizaje

Tráfico de red requerido en el aprendizaje

El tráfico de red producido por la realización de los diferentes tipos de aprendizaje es un parámetro que deberá ser tenido en cuenta a la hora de decantarse por un método u otro en dispositivos que no posean conexión de datos, o bien que dicha conexión sea costosa o demasiado lenta.

De todos los tiempos de aprendizaje que se han llevado a cabo en esta sección tan sólo el basado en generación dinámica de intervalos puede ser realizado en local. Esto provoca que el coste de tráfico

de red para este método sea nulo. Obviamente, esto posee unas limitaciones en cuanto a rendimiento del sistema de reconocimiento, dado que el aprendizaje es muy sencillo y como veremos más adelante, provoca un elevado número de fallos en el reconocimiento.

Además de este último método, el método de aprendizaje basado en Ameva, como se dijo anteriormente, está siendo implementado actualmente para poder ser ejecutado en el propio dispositivo móvil, para lo cual se está llevando a cabo una simplificación del mismo. Sin embargo, en esta sección será analizado desde el punto de vista de un aprendizaje on-line, al igual que para el método basado en redes neuronales.

Los dos métodos más complejos hacen que sea necesario enviar la información de las variables obtenidas en el momento de recopilar los datos del aprendizaje al servidor, siendo este el que llevará a cabo todo el proceso de aprendizaje. Una vez llevado a cabo el aprendizaje, el servidor devolverá la información generada con el fin de que el proceso de reconocimiento sea llevado a cabo en el propio dispositivo.

Para agilizar el aprendizaje y reducir el riesgo de bloqueo de la red o el fallo del envío de un determinar paquete, los datos serán enviados al servidor cuando se haya obtenido un número de ventanas temporales igual al tamaño del buffer de salida. Esto hará que cada cierto tiempo, la información sea enviada al servidor, almacenándose en éste hasta que se finalice el proceso de captura de datos y deba ser llevado a cabo el aprendizaje.

Los datos que se mostrarán a continuación son correspondientes a un proceso de aprendizaje mediante los métodos Ameva y RNA, sobre los que previamente se ha llevado a cabo una reducción de variables mediante el método de ACP. Esta reducción ha conseguido pasar el número de variables de las 42 iniciales a 28, por lo que tan sólo será necesario enviar al servidor 28 datos por cada una de las ventanas temporales. El aprendizaje se ha realizado sobre un total de 10 actividades.

En la Figura 94 se representa el número de KBytes que se envían al servidor según el método de aprendizaje empleado. Como se puede observar, ambos envían el mismo número de bytes. Esto se debe a que la información que deben enviar los métodos al servidor es el conjunto de valores de las variables para cada ventana temporal.

Se pueden apreciar ciertas irregularidades en la cantidad de información, dando como resultado una tendencia de crecimiento que no es estrictamente lineal. Estas irregularidades se deben a la cantidad de información adicional que debe enviar el dispositivo a la hora de encapsular los distintos conjuntos de ventanas temporales. En la práctica, el buffer de salida de ventanas temporales tiene un tamaño de 5 unidades, por lo que tan sólo se enviará un grupo de ventanas cuando se hayan recopilado 5 ventanas temporales. El resultado es que cada grupo de encapsulamiento de ventanas consumirá una cantidad de información adicional, la cual incluye la cabecera y los datos necesarios del paquete de datos que será enviado al servidor. Esta información tiene un tamaño de 980 Bytes, es decir, algo menos de 1KB. En cambio, la cantidad de información aproximada para almacenar los datos correspondientes a una ventana temporal es de alrededor de 192 Bytes.

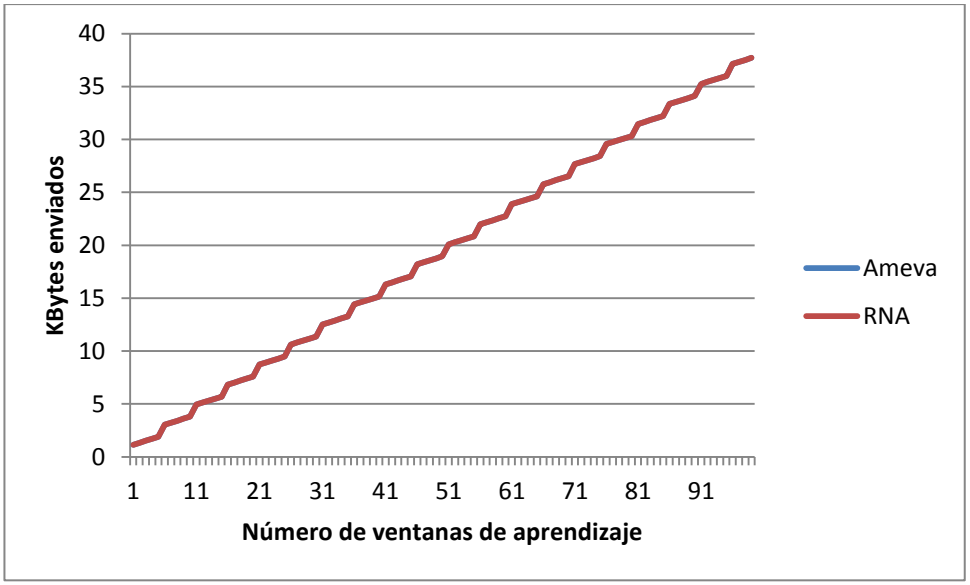


Figura 97: KBytes de información enviados al servidor para cada uno de los métodos de aprendizaje propuestos

En este sentido, no encontramos ninguna diferencia entre ambos métodos de aprendizaje respecto al tamaño de la información enviada al servidor para llevar a cabo el proceso necesario. Sin embargo, la información recibida por el dispositivo una vez realizado el aprendizaje en el servidor presenta una serie de diferencias en función del método empleado. Estas diferencias pueden ser observadas en la Figura 95. En este caso se aprecia que la diferencia entre el número de bytes recibidos por el dispositivo tras haberse realizado el aprendizaje es sustancialmente mayor en un caso que en otro. En el método basado en Ameva, el tamaño de la información se sitúa en 500 Bytes, en los cuales se incluyen los bytes correspondientes a los límites asociados a cada intervalo de las diferentes variables. En cambio, mediante el método basado en RNA, el tamaño de información asciende a 5 KBytes debido a que es necesario enviar la información de todos los pesos asociados a la red neuronal. Debido al elevado número de conexiones entre neuronas, la cantidad de información es más elevada que en el caso anterior.

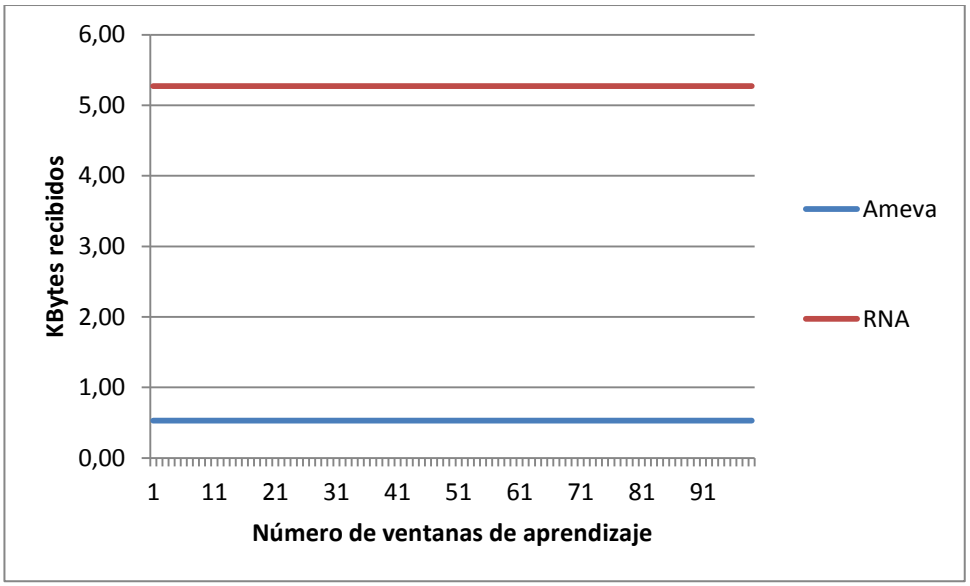


Figura 98: KBytes de información recibidos desde el servidor para cada uno de los métodos de aprendizaje

Tras analizar los tres métodos de aprendizaje, podemos concluir que el método más eficiente en cuanto al número de datos intercambiados con el servidor es el método basado en generación de intervalos, en el cual no es necesario el paso de ninguna información entre las arquitecturas, ya que todo el aprendizaje se lleva a cabo en el dispositivo. En cuanto a los restantes métodos, la diferencia es muy reducida, siendo el método basado en Ameva el que menor cantidad de información comparte debido a que el tamaño de la trama de aprendizaje devuelta por el servidor es más reducido. En cambio, el método basado en RNA requiere un mayor tamaño de la trama de información recibida por el dispositivo tras el aprendizaje, siendo la diferencia entre ambos métodos 5 KBytes de datos por cada aprendizaje realizado.

Complejidad de los métodos de reconocimiento

En este apartado estudiaremos la complejidad de los diferentes métodos de reconocimiento derivados de los métodos de aprendizaje expuestos anteriormente. Desde un punto de vista muy general, es posible detectar dos métodos de reconocimientos diferenciados en función de la técnica usada. En primer lugar aparecen los métodos de reconocimiento basados en clasificación intervalar, es decir, a partir de una serie de intervalos para las diferentes variables, el reconocimiento se basa en la actividad más probable a partir de la clasificación realizada por cada actividad. De esta forma, un intervalo posee un cierto grado de probabilidad para cada actividad, por lo que finalmente la actividad más probable será aquella que maximice la suma de dichas probabilidades para cada variable. Los métodos estudiados que emplean esta técnica de reconocimiento son los basados en Ameva y la generación dinámica de intervalos.

Por otro lado aparecen aquellas técnicas que emplean una red neuronal para la clasificación de las actividades. En este caso el proceso de reconocimiento es sensiblemente más complejo debido a que es necesario cargar una red neuronal en el dispositivo, introducir los valores de las diferentes variables en las entradas de la red neuronal y, por último, seleccionar aquella actividad asociada a la neurona de la capa de salida con mayor nivel de activación. En este caso, el método de reconocimiento que se basa en esta función es el basado en RNA.

A partir de esta clasificación, analizaremos el comportamiento de ambos métodos desde un punto de vista de la eficiencia, es decir, del tiempo de ejecución necesario para llevar a cabo el reconocimiento una vez realizado el aprendizaje. Los valores del tiempo de ejecución en este caso, serán obtenidos totalmente desde el propio dispositivo, ya que será el lugar donde se lleve a cabo el reconocimiento para todos los métodos.

A diferencia del aprendizaje, el reconocimiento será realizado cada pocos segundos con el fin de mantener el mayor nivel de actualización sobre la actividad realizado. Esto hace que sea necesario reducir al máximo el coste del reconocimiento, ya que como se expuso anteriormente, puede provocar un colapso del sistema de reconocimiento debido al tiempo excesivo de computación. El colapso del sistema sería un caso ciertamente extraño, ya que el tiempo de reconocimiento que se debería dar para que se produjera debería ser mayor o igual al tamaño de la ventana temporal. Sin embargo, aunque no se llegue el efecto descrito, un elevado tiempo de reconocimiento llevaría a un problema de consumo excesivo de procesamiento, que como se ha detallado, conlleva un consumo energético elevado que reducirá el tiempo de vida de la batería del dispositivo.

En primer lugar veamos el tiempo de ejecución del proceso de reconocimiento para el método basado en clasificación intervalar. En la Figura 99 se muestra el tiempo de ejecución medida sobre un total de 100 ventanas temporales obtenidas a partir del dispositivo móvil durante la realización de

varias actividades físicas por parte del usuario. El primer dato que se puede obtener del diagrama es que no existen tiempos de reconocimiento superiores al valor umbral de colapso que está ajustado a 5 segundos. Concretamente, el máximo tiempo obtenido para la obtención de una actividad a partir de las variables de entrada fue de 0,75 segundos. La sencillez del método de clasificación intervalar se traduce, como puede observarse, en un tiempo de reconocimiento muy bajo y sencillo, el cual se elegirá como método de reconocimiento prioritario en el sistema. Muestra de esta eficiencia es la media del tiempo de reconocimiento establecida a tan sólo 0,6 segundos.

Otra conclusión que puede ser obtenida a partir de la Figura 99 es la estabilidad del tiempo de reconocimiento, de manera que la diferencia entre el máximo y el mínimo tiempo registrado es de tan sólo 2 décimas de segundo.

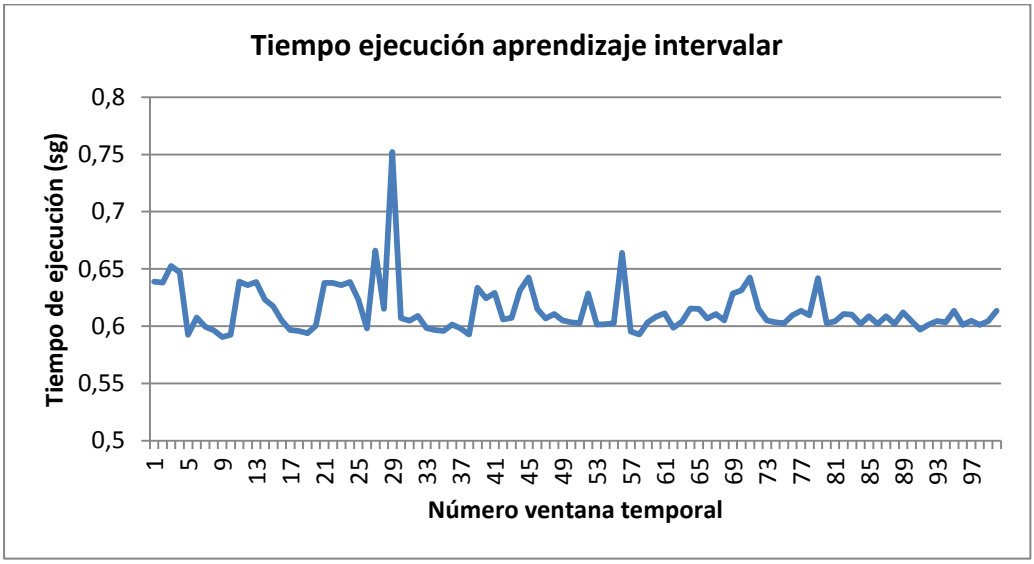


Figura 99: Tiempo de ejecución del proceso de reconocimiento para el método basado clasificación intervalar

Una vez presentados los resultados para el método de reconocimiento intervalar, se expondrá los tiempos obtenidos para el método de reconocimiento basado en redes neuronales. Como se ha comentado anteriormente, el hecho de incluir una red neuronal en el propio dispositivo con los valores generados a partir del proceso de aprendizaje, supondrá un coste añadido al método anterior. Sin embargo, este coste tan sólo deberá asumirse en el caso del aprendizaje, siendo la incumbencia del proceso de reconocimiento tan sólo el usar dicha red previamente entrenada.

Por otro lado, para que la clasificación mediante RNA sea correcta, deberá aplicarse a los valores de entrada de cada una de las variables una serie de operaciones lineales que determinen la ponderación de las conexiones sinápticas entre neuronas así como activación de la propia neuronal, regida generalmente por una función de tipo sigmoide. Estas operaciones se verán reflejadas en un coste temporal más elevado que el caso anterior para el proceso de reconocimiento.

En la Figura 100 se puede observar la evolución del tiempo de reconocimiento del método de RNA para el caso de 100 ventanas temporales. En primer lugar, el tiempo de ejecución para este método es claramente más elevado que para el método basado en clasificación intervalar. Esto se debe a lo expuesto anteriormente, aunque aun así está lejos del valor umbral de colapso. Sin embargo, el ajuste del tamaño de la ventana temporal a un valor más pequeño podría producir problemas. En todo

caso, el principal inconveniente de este tiempo de procesamiento adicional es el tiempo útil de la batería tras llevar a cabo el reconocimiento durante un tiempo elevado de manera ininterrumpida.

Otro efecto que puede ser observado en la Figura 100, es la fluctuación del tiempo de reconocimiento, siendo la diferencia entre el máximo y el mínimo de 23 décimas, un valor algo superior al caso anterior en el que se ajustaba en 16 décimas de segundo.

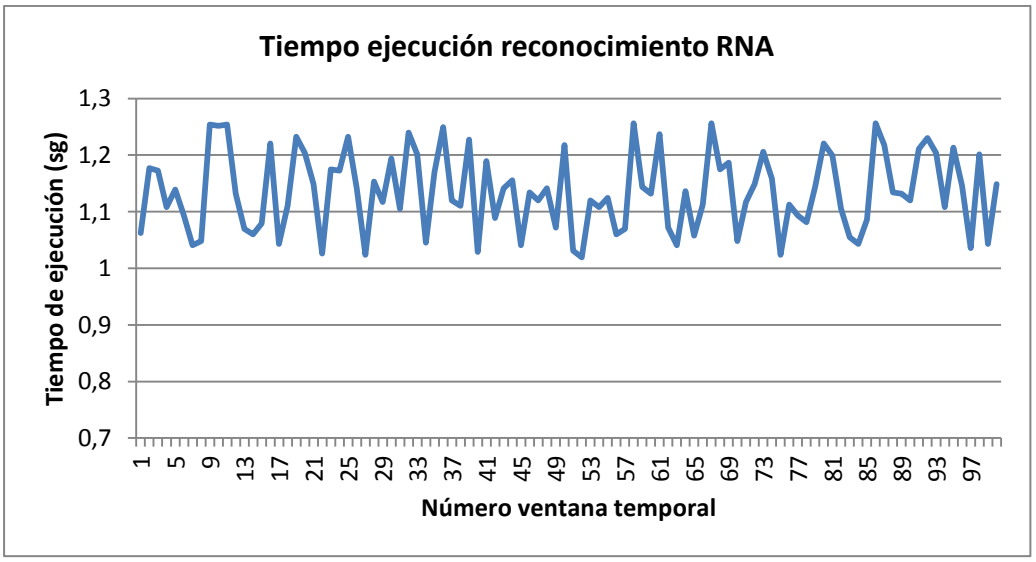


Figura 100: Tiempo de aprendizaje del proceso de reconocimiento para el proceso basado en el método RNA

Como conclusión de la comparativa realizada sobre los diferentes métodos podemos decir que el método de reconocimiento basado en clasificación intervalar es sensiblemente mejor que el basado en RNA en cuanto a tiempo de reconocimiento se refiere. Concretamente, el tiempo de reconocimiento se reduce a la mitad mediante el uso del método basado en clasificación intervalar. Aun así, deberemos evaluar la precisión para decantarnos definitivamente por uno de estos métodos, para lo cual se llevará una comparativa en la sección siguiente.

Precisión de los métodos de reconocimiento

Comenzaremos esta comparativa describiendo los diferentes casos de pruebas que han sido empleados para determinar la precisión de cada método. A lo largo de toda la sección se trabajará con dos paquetes de datos distintos; por un lado un paquete correspondiente a un conjunto de datos obtenidos a partir de acelerometría modular, con un total de 12 variables y 870 ventanas temporales de análisis. Por otro lado, el test se realizará sobre otro paquete de datos obtenidos a partir de acelerometría triaxial, con lo que trabajaremos con 48 variables durante las 960 ventanas temporales recopiladas para llevar a cabo el test. Sobre cada uno de estos test de prueba se han efectuado dos variaciones, por un lado una reducción de variables basada en el método de análisis de componentes principales y por otro una reducción basada en coeficientes de correlación.

Para comprobar la precisión del método se ha entrenado el sistema para ser capaz de reconocer 8 actividades cotidianas en la vida de cualquier persona: andar, correr, saltar, parado, ciclismo, tumbado, subir escaleras y bajar escaleras. El aprendizaje realizado por todos los usuario sometidos al experimento ha consistido en 15 minutos por actividad, excluyendo la actividad de salto, para la cual el entrenamiento tan sólo ha sido de 5 minutos.

Comenzaremos analizando la precisión del método de reconocimiento basado en generación de intervalos dinámicos. Hemos omitido de la comparativa el método basado en intervalos estáticos debido al problema presentado al llevar a cabo el reconocimiento de actividades de un usuario distinto al que realizó el aprendizaje. En la Tabla 6 se presentan la tabla de resultados de la clasificación para el método de reconocimiento intervalar basados en el número de ventanas temporales clasificadas correcta e incorrectamente durante el proceso de reconocimiento.

Actividad	Ventanas totales	Ventanas correctas	% de acierto
Andando	145	122	84.00%
Saltando	94	78	83.00%
Parado	137	122	89.00%
Corriendo	86	68	79.00%
Subiendo escaleras	68	46	68.00%
Bajando escaleras	67	44	66.00%
Ciclismo	157	107	68.00%
Tumbado	116	92	79.00%

TABLA 6: RESULTADOS DEL RECONOCIMIENTO INTERVALAR

Como podemos comprobar, los porcentajes de acierto son bastante bajos en determinados casos como en los de subir y bajar escaleras y ciclismo. Esto se debe a que existen actividades que poseen perfiles semejantes de acelerometría, por lo que son fácilmente confundibles. El hecho de crear un intervalo para cada actividad, resulta beneficioso cuando las actividades son claramente distinguibles, pero presenta serios problema al producirse solapamientos. Al forzar la pertenencia de una actividad a un conjunto determinado, influye en la presencia de numerosos falsos positivos sobre ciertas actividades, las cuales se ven ponderadas muy superiormente a la ponderación real que debería poseer.

Para solucionar el problema presente en el anterior método, se propuso una generación intervalar basada en el algoritmo Ameva, el cual proporciona como máximo un número de intervalos igual al número de clases del sistema. Sin embargo, si las clases no son disjuntas frente a una variable determinada, Ameva recogerá en el mismo intervalo ambas actividades, evitando así el problema de falsos positivos de la clasificación. De esta forma, siguiendo con el reconocimiento de actividades basado en al ámbito discreto, en la Tabla 7 se muestran los resultados para el método basado en Ameva.

Actividad	Ventanas totales	Ventanas correctas	% de acierto
Andando	145	142	98.00%
Saltando	94	92	97.00%
Parado	137	135	99.00%
Corriendo	86	84	98.00%
Subiendo escaleras	68	65	95.00%
Bajando escaleras	67	65	97.00%
Ciclismo	157	152	97.00%
Tumbado	116	115	99.00%

TABLA 7: RESULTADOS DEL RECONOCIMIENTO BASADO EN AMEVA

En este caso, el porcentaje de acierto es mucho más elevado que en el método basado en la generación de intervalo. Concretamente se ha notado una mejoría sustancial en las actividades que presentan ciertas semejanzas, como *subir* y *bajar escaleras* o *correr* y *saltar*. Las principales ventajas por tanto del método basado en la clasificación mediante la generación de intervalos Ameva, resultan ser tanto la precisión como la complejidad en el dispositivo, así que será el método por defecto de la plataforma de reconocimiento desarrollada. Otra ventaja del método actual es que la clasificación es visual, de forma que hasta cierto punto la actividad reconocida por el sistema es predecible y transparente, a diferencia del método basada en redes neuronales.

El reconocimiento basado en redes neuronales posee una elevada precisión pero, como se vio en el apartado anterior, la complejidad del aprendizaje iterativo es grande. Por otro lado el manejo de pesos de la red neuronal no es trivial, puesto que será necesario crear una red en el propio dispositivo para simular el reconocimiento en base a la ponderación de cada una de las conexiones sinápticas entre las neuronas. Sin embargo, en el apartado actual evaluaremos la precisión, la cual puede verse de manera resumida en la Tabla 8.

Actividad	Ventanas totales	Ventanas correctas	% de acierto
Andando	145	140	97.00%
Saltando	94	89	95.00%
Parado	137	134	98.00%
Corriendo	86	83	97.00%
Subiendo escaleras	68	63	94.00%
Bajando escaleras	67	62	93.00%
Ciclismo	157	151	96.00%
Tumbado	116	115	99.00%

TABLA 8: RESULTADOS DEL RECONOCIMIENTO BASADO EN REDES NEURONALES

Tal y como se ha comentado anteriormente, la precisión del método es muy elevado, rozando prácticamente la perfección en distintas actividades. De esta forma, si nos ceñimos únicamente a la precisión, en principio no podríamos decantarnos por el método basado en redes neuronales o el basado en Ameva. En la Figura 101 podemos comparar los diferentes porcentajes de reconocimiento para las actividades en función del método de reconocimiento empleado. En una primera impresión, el aspecto más significativo de los datos es la clara diferencia existente entre el método intervalar y el resto. Como se ha comentado anteriormente cuando estudiamos en profundidad los resultados para cada método, las dos técnicas que se emplearán en la mayoría de los casos para llevar a cabo el reconocimiento será la técnica basada en la generación de intervalos mediante Ameva y las redes neuronales. Ambas gozan de un porcentaje de acierto elevado y, en general, el coste de reconocimiento no es excesivamente elevado.

En cambio, afinando aún más el análisis, podemos observar que para la mayoría de las actividades, el porcentaje de reconocimiento de la técnica basada en Ameva es mayor que la de RNA. Esto pone de manifiesto la ventaja de trabajar en el dominio discreto en determinados problemas, como es el caso del reconocimiento de actividades. A menudo, la pérdida de información provocada a la hora de llevar a cabo la discretización de una determinada variable, se ve compensada con la generalización que se hace del problema, haciendo mucho más flexible la interpretación así como más sencillo el trabajo con los datos.

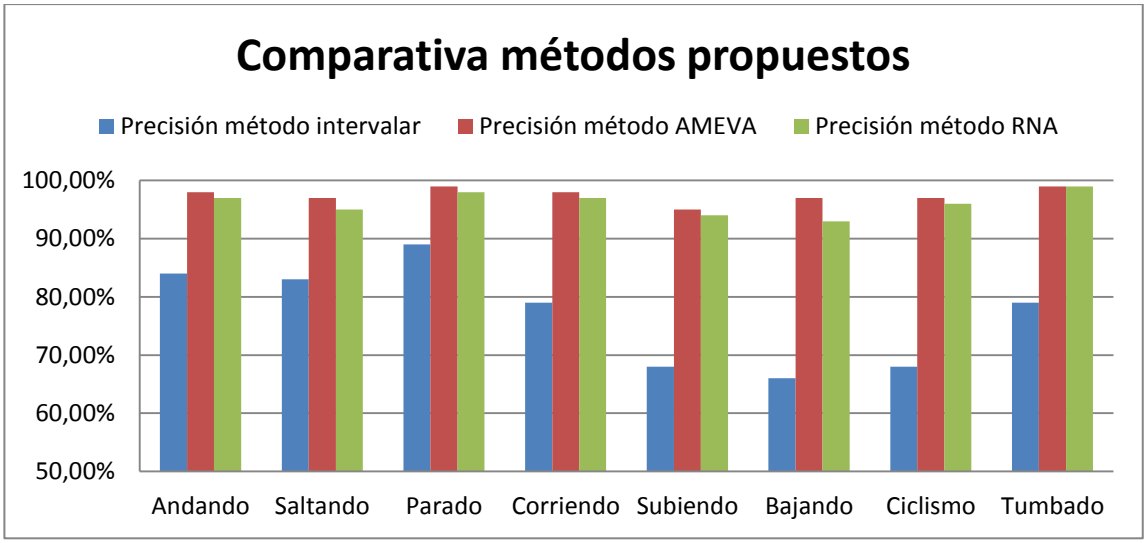


Figura 101: Comparativa métodos de reconocimiento

Como resumen de esta sección podríamos concluir que el método de reconocimiento discreto basado en Ameva es el más eficiente y preciso de todos los estudiados, además de proporcionar una escalabilidad adecuada a la hora de aumentar el número de actividades reconocidas. El problema del sobre-aprendizaje en este método no es excesivamente delicado, a diferencia del método de redes neuronales. De esta forma, el usuario tendrá más libertad para realizar un entrenamiento más laxo, sin ningún control del mismo por parte del sistema.

Tanto en este caso como en el resto de este capítulo, la realización de un entrenamiento adecuado es crucial para el buen funcionamiento del sistema. Por este motivo el usuario debe prestar especial interés a este aspecto del sistema, ya que en él se basará el resto de procesos que se llevan a cabo.

MODELADO DEL PROBLEMA

Llegados a este punto el sistema es capaz de determinar con más o menos precisión la actividad que está realizando el usuario. Sin embargo, si se realiza un análisis minucioso de los errores producidos durante el reconocimiento, podríamos llegar a la conclusión de que la mayoría de ellos se produce en un momento de *transición*, es decir, en el instante en el que el usuario deja de hacer una actividad para realizar otra distinta. A veces, este tipo de errores puede ser solventado incluyendo un conjunto de restricciones a la hora de detectar que una determinada actividad que se ha realizado durante un tiempo, ha dejado de llevarse a cabo y se está ejecutando otra actividad distinta. En esta sección trataremos de sentar los principios básicos para el modelado eficaz del problema del reconocimiento de actividades, identificando en él estado y transiciones. Sin embargo, este proceso aunque será planteado, no será evaluado, ya que formará parte de una investigación futura a cerca del aumento de la eficacia del sistema de reconocimiento apoyado por una cadena de Markov. Una vez obtenido el modelo, será posible dotar al sistema de una mayor precisión gracias al filtro que proporciona una cadena de Markov, la cual posibilita definir las probabilidades de las transiciones y hace que el método de reconocimiento sea más robusto frente a fallos producidos entre transiciones de actividad.

MODELO DE MARKOV

Una cadena de Markov, que recibe su nombre del matemático ruso Andrei Andreevitch Markov (1856-1922), es una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. En efecto, las cadenas de este tipo tienen memoria. "Recuerdan" el último evento y esto condiciona las posibilidades de los eventos futuros. Esta dependencia del evento anterior distingue a las cadenas de Markov de las series de eventos independientes, como tirar una moneda al aire o un dado.

Este tipo de proceso, introducido por Markov en un artículo publicado en 1907,1 presenta una forma de dependencia simple, pero muy útil en muchos modelos, entre las variables aleatorias que forman un proceso estocástico.

En matemáticas, se define como un proceso estocástico discreto que cumple con la propiedad de Markov, es decir, si se conoce la historia del sistema hasta su instante actual, su estado presente resume toda la información relevante para describir en probabilidad su estado futuro.

Una cadena de Markov es una secuencia X_1, X_2, X_3,... de variables aleatorias. El rango de estas variables, es llamado espacio estado, el valor de X_n es el estado del proceso en el tiempo n. Si la distribución de probabilidad condicional de X_{n+1} en estados pasados es una función de X_n por sí sola, es decir:

$$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_2 = x_2, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n).$$

MARKOV SOBRE ACTIVIDADES

En este punto, gracias a los métodos de reconocimiento estudiados anteriormente, el sistema ya tiene la capacidad de determinar la actividad más probable en función de las lecturas tomadas a partir del sensor de acelerometría. Sin embargo, en esta sección introduciremos un aspecto más de nuestro trabajo para controlar la detección de actividades, una cadena de Markov que controle a un nivel superior la decisión final de la actividad más probable. Dicho trabajo se completará posteriormente a la realización de este documento, exponiendo aquí tan sólo los principios básicos en los que se basará el método de filtrado de actividades mediante cadenas de Markov.

La elección de una cadena de Markov para modelar la actividad de un usuario se basa en la realidad evidente de que si un usuario está realizando una actividad, existen una serie de actividades afines que realizará con una probabilidad más elevada en el instante de tiempo siguiente. Pongamos un ejemplo. Si la actividad que el usuario ha realizado en el último instante temporal fue *sentado*, existe una probabilidad mucho mayor de que el usuario continúe *sentado* que de que su actividad siguiente sea *saltando*.

Como se ha presentado anteriormente, los estados de la cadena de Markov serán las diferentes actividades que el sistema puede reconocer y ciertas transiciones. Las probabilidades de transición de la cadena de Markov (p_{ij}) vendrán determinadas por la probabilidad de que dado el estado ∂_i, se produzca una transición a otro estado ∂_j. Dicho de otro modo, la probabilidad de que si el usuario está realizando una actividad, pase a realizar otra actividad distinta o igual a la anterior.

A lo largo de este capítulo se ha desarrollado un sistema de reconocimiento de actividades físicas llevadas a cabo por diferentes usuarios, a partir de los acelerómetros presentes en los dispositivos móviles de última generación. Para ello se han desarrollado diversas técnicas de aprendizaje y reconocimiento basadas en redes neuronales, aprendizaje bayesiano y aprendizaje discreto. Con el fin de aplicar dichos métodos han sido estudiados diversos métodos de discretización y clasificación de muestras. Todo este proceso se ha realizado partiendo en todo momento de los criterios de minimización de recursos y memoria que están asociados al desarrollo de proveedores de contextos y aplicaciones en general en dispositivos móviles. Por último, para la determinación de la precisión del sistema de reconocimiento se ha efectuado una batería de pruebas dando unos resultados muy satisfactorios. Concretamente el porcentaje de acierto de los diferentes métodos de clasificación está en torno al 97%, lo cual indica que el porcentaje de errores de clasificación es tan sólo del 3%, muy inferior al de los sistemas estudiados en la bibliografía.

CAPÍTULO 5: SISTEMA DE DETECCIÓN DE SALIDAS

By concentrating our efforts upon a few major goals, our efficiency soars, our projects are completed, we are going somewhere.
- Michael Korda -

A lo largo de este capítulo se desarrollará un sistema capaz de detectar transiciones entre un lugar techado y un lugar al aire libre. Este sistema permitirá incrementar de una manera significativa el tiempo total de uso de los sistemas de posicionamiento tradicionales en un dispositivo móvil, gracias al ahorro de energía que proporciona. Concretamente, el objetivo será desarrollar un sistema de posicionamiento ubicuo capaz de monitorizar la posición de un usuario durante al menos 24 horas, tiempo durante el cual no será necesario ningún tipo de recarga de baterías del dispositivo.

Para ello, en primer lugar se introducirá las diferentes técnicas de posicionamiento existentes actualmente, para continuar analizando las diferentes propuestas de localización desarrolladas en los últimos años para sistemas ubicuos. Seguidamente comenzaremos a desarrollar nuestra propuesta a partir de la justificación de la misma y la explicación en profundidad de las diferentes alternativas propuestas. Deberemos tener en cuenta que la elección del método concreto de detección de salidas deberá seleccionarse en base a las necesidades tanto energéticas como de precisión del caso que se aborde. Con este fin, llevará a cabo una comparativa que permitirá medir el ahorro energético de los distintos métodos.

INTRODUCCIÓN A LOS SISTEMAS DE POSICIONAMIENTO UBICUOS

El objetivo del sistema a desarrollar es conseguir que la localización personal sea un servicio de tiempo real totalmente ubicuo, es decir, que podamos disponer de datos precisos de posicionamiento en cualquier lugar y en cualquier momento. En cualquier lugar significa que el servicio debe ir con nosotros allá dónde vayamos, por lo que debe ir alojado en un dispositivo personal que tenga la capacidad suficiente. En cualquier momento implica que ese servicio podrá suministrar información tanto en interiores como en exteriores y que además, dispondremos de él el máximo tiempo posible. En este capítulo nos centramos en conseguir parte de este objetivo: obtener un servicio de localización personal de gran precisión en exteriores y de una duración superior a la de los servicios disponibles actualmente, permitiendo de este modo aumentar el tiempo de vida útil asociado al dispositivo para permitir la ejecución, entre otras aplicaciones, como sistema de seguimiento continuo de usuarios.

El sistema que será desarrollado se apoya en dos evidencias:

- El sistema de localización GPS no es útil en interiores.
- Las personas normalmente pasamos más tiempo en interiores que exteriores.

La primera evidencia viene definida por el propio diseño del sistema de posicionamiento global GPS, el cual necesita la visión directa de cuatro satélites para determinar con una precisión de varios metros la posición exacta del usuario. De esta forma, si el usuario se encuentra en el interior de un lugar techado, no es posible localizar los satélites necesarios por parte del sensor de posicionamiento.

La segunda evidencia la podemos observar en diferentes estudios, como por ejemplo en (Device Positioning Using Radio Beacons in the Wild, 2005) y en (Transportation Research Record, 2005), los cuales aseguran que la mayor parte del tiempo, las personas permanecen en lugares techados a lo largo de un día. Para corroborar dichos estudios, hemos estudiado 50 usuarios de diversas edades, y hemos concluido que tan sólo el 9% del tiempo total diario (129 minutos al día) el usuario se encuentra en exteriores. Se debe tener en cuenta que este porcentaje variará en función de la edad de los individuos bajo estudio.

Estas evidencias nos llevan a pensar que podemos utilizar el sistema de posicionamiento GPS (Global Positioning System) en exteriores y cuando entremos en interiores desactivarlo, manteniendo como localización válida la última obtenida. Sin embargo, mientras que la detección de entrada en lugares techados es relativamente sencilla, detectar las transiciones de un usuario desde el interior de un edificio al exterior para poder reactivar la localización por GPS, será una tarea compleja que se asentará como la cuestión fundamental de este capítulo.

En exteriores, el sistema más maduro y utilizado es el sistema GPS, cuyas "balizas", en este caso satélites, están situadas en el espacio y gracias a la visión directa de cuatro de ellos, es posible determinar con una precisión de pocos metros la posición exacta del dispositivo receptor (receptor GPS). Los satélites se posicionan siguiendo una órbita previamente establecida, de manera que gracias a un registro almacenado en el receptor (almanaque) es posible conocer con exactitud la posición de la baliza. Mediante triangulación de las señales recibidas, el receptor obtiene la posición exacta en la que se encuentra. Se trata del único sistema de posicionamiento global funcional que existe.

Sin embargo, no hay una solución definitiva para la localización en interiores. El coste energético del receptor en el sistema GPS es muy elevado, especialmente cuando el receptor pierde la visión de los satélites. Esto provoca que el dispositivo entre en una búsqueda continua de señal, lo que hace que la eficiencia disminuya de manera acusada. Además de este inconveniente, hay problemas de reflexión (efecto can) en grandes ciudades con rascacielos o calles estrechas, lo cual se puede observar en la Figura 102. Por este motivo, durante los últimos años se han utilizado diferentes técnicas que complementan (o incluso sustituyen) al GPS. Entre estas técnicas se incluye la tecnología GSM, cuyas balizas son las estaciones base de telefonía (cobertura de más del 80% mundial sin incluir la superficie marina) o la tecnología bluetooth, donde las balizas son los equipos fijos emisores de señales bluetooth.

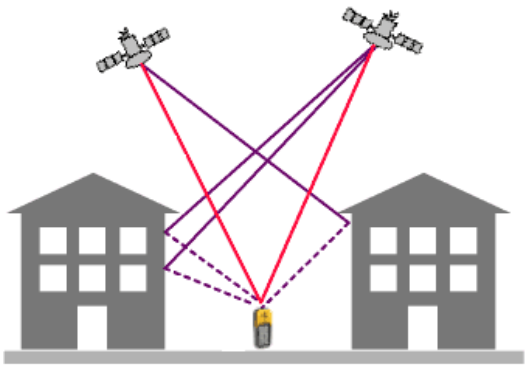


Figura 102: Efecto de la reflexión de señal en la tecnología de posicionamiento GPS

Mención aparte merece la tecnología WiFi, que día a día se está convirtiendo en una clara alternativa al geoposicionamiento GPS tradicional. La tecnología WiFi, en la que sus balizas son los puntos de acceso a internet inalámbricos, posee una precisión muy elevada en grandes ciudades y en general, en cascos urbanos muy poblados. Además, la cobertura de este sistema se amplía día a día gracias a la creación de nuevos puntos de accesos WiFi por parte de los usuarios y al geoposicionamiento de los mismos. Según un informe realizado en 2009 por el Grupo Gowex de servicios de telecomunicación, el número de redes WiFi instaladas en España a finales de 2008 era de 4891 y en Estados Unidos este número ascendía a los 69180.

Aunque el posicionamiento basado en redes WiFi es una buena alternativa, no es viable en lugares con baja densidad de redes por habitante. Además, incluso con una densidad media, el posicionamiento no es lo suficientemente exacto como para registrar un diario de las rutas realizadas por el usuario, ya que la movilidad de estos puntos de acceso hace que la posición de estos no permanezca constante.

En las tres técnicas citadas, el receptor no conoce la posición de las balizas, tal y como ocurría en el caso del sistema GPS, por lo que es necesaria una base de datos que almacene su identificador único y su posición asociada. Recopilar esta base de datos y actualizarla resulta complejo por lo que se utiliza la comunidad de usuarios de la respectiva técnica como gestora para que realicen esta tarea de manera eficiente. Incluso haciendo uso de la comunidad, la poca precisión del posicionamiento de las antenas y el abandono de algunas bases de datos hace que la información no se actualice de manera eficaz y, sobre todo, fiable.

A pesar de todas las ventajas de las técnicas alternativas de localización, existe un gran problema cuando lo comparamos con el GPS: no existe cobertura global. Esto es un gran problema cuando el uso se realiza en zonas con poca densidad de balizas como por ejemplo, en alta mar, zonas rurales o ciudades poco desarrolladas. En estos casos, el único sistema válido será el tradicional GPS.

En esta sección abordaremos el problema del consumo de energía desde un punto de vista novedoso, ya que se busca determinar el momento en el que un usuario ha entrado en un recinto cerrado y, lo más importante, detectar cuando sale de él. De esta forma, sería posible desconectar el dispositivo GPS durante el tiempo que el usuario permanece en un lugar cerrado sin cobertura GPS y volverlo a conectar a su salida. Esto es muy importante si consideramos que la mayor parte del consumo del sistema GPS se produce cuando intenta obtener la señal de los satélites.

Sabiendo el momento en el que el usuario sale al exterior, podremos ahorrar de manera drástica el consumo de energía del dispositivo. Además, debemos tener en cuenta que el tiempo que nos desplazamos en exteriores suele ser mucho menor que el que permanecemos en interiores, por lo que el porcentaje de localizaciones correctas en un día puede ser en torno al 9% del total del tiempo en el que el receptor está conectado. Esto nos lleva a diseñar un sistema que permita la localización durante el máximo tiempo posible combinando diferentes técnicas buscando un máximo ahorro energético.

Dado que lo que buscamos es un sistema de posicionamiento ubicuo, es indispensable que el dispositivo que permita la geolocalización del usuario esté próximo al él y que esté disponible el mayor tiempo posible. En general, los tres elementos que normalmente nos acompañan en cualquier desplazamiento fuera de nuestro hogar son: el móvil, las llaves y la cartera. La inclusión de otro elemento adicional supondría aumentar la probabilidad de olvido y por consiguiente los fallos de localización, además de la incomodidad añadida al usuario al deber llevar un dispositivo adicional. La mayoría de las tecnologías que emplearemos (GPS, WiFi, GSM-GPRS-UMTS y Bluetooth) han sido

integradas en los móviles de nueva generación como un elemento más del hardware, por lo que esta será la pieza fundamental de nuestro desarrollo.

Por todo ello, se busca desarrollar un método eficaz que permita disminuir al máximo el coste energético de la geolocalización. De esta forma será posible integrar en un dispositivo móvil la capacidad de localizar al usuario durante largos periodos de tiempo sin necesidad de recargar la batería y sin pérdida de precisión en la localización. Este sistema, permitirá además ampliar la capacidad de localización del usuario, ya que al estar integrado en su propio dispositivo móvil, sin ningún otro artefacto adicional, podrá ser portado continuamente por éste, favoreciendo así las características de los sistemas ubicuos.

ESTADO DEL ARTE

El primer requisito para conseguir un sistema de geoposicionamiento ubicuo es reducir al máximo el coste energético del sistema dado que, de esta manera, conseguiremos geoposicionar al usuario durante más tiempo. Existen algunas alternativas hardware que realizan esta labor (Battery-Aware Embedded GPS Receiver Node, 2007) pero el inconveniente de ellas es el coste y el empleo de dispositivos específicos que hacen más difícil la integración con el usuario. Un ejemplo de un dispositivo específico de localización se muestra en la Figura 103, donde la circuitería se corresponde con un sistema basado en GSM como método de posicionamiento.

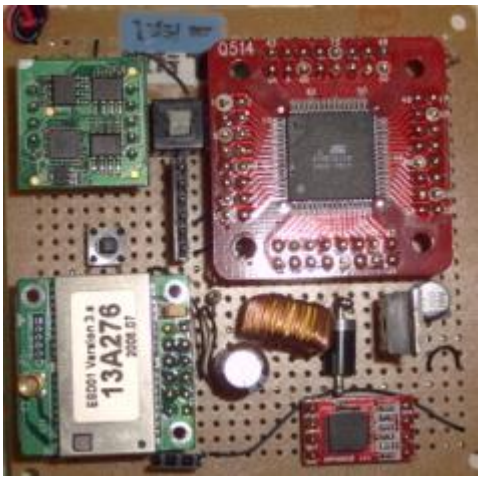


Figura 103: Dispositivo de localización de hardware específico desarrollado en el año 2008

Uno de los dispositivos específicos es la herramienta Placelab (Large-Scale Localization fromWireless Signal Strength, 2005), (Device Positioning Using Radio Beacons in the Wild, 2005). Se trata de un sistema de localización basado en redes Wifi's. La ventaja de esta aplicación es que no necesita de un hardware propietario, sino que es capaz de correr bajo distintos modelos de dispositivos existentes en el mercado. Sin embargo, el principal problema que presenta reside en la precisión. Puesto que Placelab emplea la detección basada en redes inalámbricas, la precisión se vería comprometida si el usuario reside en zonas poco urbanizadas o en pueblos alejados de las grandes ciudades. Además de esto, incluso si el usuario se sitúa en zonas con una gran cantidad de densidad de redes, la localización no tiene porqué ser precisa, ya que es necesario la geolocalización de las redes para que sea posible el posicionamiento aproximado del usuario. Esto es un grave problema en lugares donde no existe esta geolocalización y, dado que es la propia comunidad la que se encarga de esto, en

la mayoría de los casos, sólo en grandes ciudades será posible crear una base de datos lo suficientemente grande como para obtener un sistema fiable de localización basado en redes Wifi. Como veremos más adelante, nuestro sistema no necesita que las redes estén geoposicionadas, sino que simplemente haremos uso de las intensidades y una serie de identificadores únicos para cada una de las redes que detectadas y susceptibles de ser usadas mediante el sistema.

Otro trabajo muy relacionado con el sistema anterior es el realizado por Jun Rekimoto et. Al (LifeTag: WiFi-Based Continuous Location Logging for Life Pattern Analysis, 2007), donde trata el tema de la localización mediante redes inalámbricas. En este caso, el sistema se desarrolla en Tokyo donde la densidad de redes Wifi es espectacular. Mediante esta técnica se consigue un posicionamiento muy eficaz incluso en interiores aunque, como dijimos antes, el precio que hay que pagar en este tipo de técnicas es la poca eficiencia en zonas donde la densidad de redes es menor.

Otro sistema de teleasistencia dedicado al posicionamiento de usuarios es "Keruve". Se trata de un sistema de localización para personas con alzhéimer. El sistema es muy eficaz y, a diferencia del anterior, se basa en la localización mediante GPS. Por tanto, queda resuelto el problema de la precisión de la localización. En lo referente a la autonomía, es muy elevada, ya que es capaz de permanecer activo durante 3,5 días. Sin embargo, este dato puede llevar a confusión, ya el uso del terminal no se realiza de manera continuada, sino a petición. Puesto que el sistema está compuesto por un emisor (reloj de muñeca) y un terminal, ambos dispositivos específicos, el sistema de posicionamiento del reloj sólo se activa cuando el terminal lo solicita, así que sólo es posible realizar un seguimiento continuado si sacrificamos la autonomía y realizamos peticiones constantes. Otro problema del sistema es que la comunicación entre emisor y receptor se realiza mediante mensajes de texto, por lo que el coste de una localización continua es muy elevado.

Por último, debemos decir que el sistema Keruve está basado en hardware específico, por lo que el coste de la localización debido a la compra de los terminales es bastante elevado. En cambio, nuestro sistema está pensado para cualquier dispositivo móvil, por lo que el inconveniente del hardware propietario está solventado. Además, no necesitaríamos llevar un dispositivo localizador específico (que en el caso de Keruve era un reloj), sino que es el propio móvil del usuario el que realiza todas las funciones.

Además de Keruve, existen otra serie de dispositivos capaces de realizar tareas de localización parecidos a éste. Estos son, por ejemplo, Columba o Simap. Sin embargo, estos sistemas poseen una menor precisión en la localización y el dispositivo que debe llevar el usuario es de mayores dimensiones. En caso de Simap, se trata de un gran proyecto español para la monitorización del posicionamiento y la actividad de los usuarios. La principal ventaja de este sistema es su autonomía, la cual supera las 50 horas y su alto nivel de configuración, puesto que permite definir una serie de zonas *seguras* las cuales no podrán ser rebasadas por el usuario bajo seguimiento. Sin embargo, el problema que puede ser identificado en este sistema es la necesidad de portar un hardware específico que podría hacer su uso más incómodo y poco natural.

Figura 104: Conjunto de emisor (a la derecha) y receptor (a la izquierda) del sistema de teleasistencia Keruve

El punto destacable de los sistemas anteriores, que como vimos era la autonomía, es lo que pretendemos alcanzar con nuestro desarrollo sin sacrificar la continuidad del posicionamiento ni la precisión. Es decir, queremos lograr un sistema GPS de localización continua que, aun corriendo sobre un dispositivo móvil, sea capaz de obtener una autonomía semejante al de un sistema específico sin costes adicionales. En la sección de conclusiones veremos si hemos conseguido el objetivo y compararemos el tiempo de vida de la batería con el de los sistemas vistos anteriormente.

Por otro lado, existe una serie de sistemas de posicionamiento que se basan en la acelerometría para activar o desactivar ciertos sensores como el GPS (Less is More: Energy-Efficient Mobile Sensing with SenseLess, 2009). Esto hace que usando un sensor cuyo consumo energético es muy reducido, como el caso del acelerómetro, sea posible detectar movimiento del usuario, condición que será tenida en cuenta para activar el sensor GPS. Sin embargo, como se detallará en las secciones siguientes, este tipo de aproximación tiene un grave problema si la actividad que el usuario está realizando en interiores conlleva algún movimiento. En este caso, el sensor GPS estaría activado la totalidad del tiempo, a pesar de que el usuario se encuentra en interior. A pesar de esto, las pruebas realizadas con este sistema indican que se produce un aumento del tiempo de baterías de las 9 horas iniciales a 22 horas. Sin embargo, debemos tener en cuenta que el dispositivo móvil empleado posee una duración en espera de 6 días, frente a los 3 días que ofrecen los dispositivos sobre los que se ha llevado a cabo la prueba del sistema actual.

No debemos finalizar este apartado sin citar una característica común a todos ellos: es el usuario el encargado de activar y desactivar el sistema. En nuestro caso, dado que pretendemos diseñar un sistema de posicionamiento ubicuo, es necesario que el usuario sea totalmente independiente de la aplicación y que ésta, se rija por sí misma. De esta manera, el propio sistema podrá determinar cuándo es necesaria la activación o desactivación de la propia aplicación, o en qué momento se debe apagar o encender el dispositivo GPS sin que la persona sobre la que se va a realizar el seguimiento tenga ningún tipo de intervención.

Nuestro sistema por tanto, une las ventajas de la geolocalización por GPS, es decir, la precisión y la globalidad, con las de las redes Wifi, el bajo coste energético del sistema. Además, gracias a esta fusión no será necesario mantener ninguna base de datos con la posiciones de las distintas redes Wifi, sino que el propio sistema de balizas satelitales nos proporcionará la posición. Mediante la integración de ambas tecnologías en combinación con la acelerometría o la red GSM, podremos obtener un sistema que pueda funcionar durante largos periodos de tiempo (superiores a 24 horas), mientras es capaz de monitorizar la posición exacta de los usuarios cuando se mueven en el exterior.

Pretendemos diseñar un sistema que permita geoposicionar durante más de 24 horas de manera precisa a una persona utilizando un móvil de última generación, el cual permite integrar diversas tecnologías de localización y acelerometría que se expondrán a continuación. Basado en este sistema, han sido creadas dos aplicaciones: una que genera de manera automática un diario de lugares por los que el usuario ha pasado y otra que utilice la base de datos generada por la anterior para preguntar al usuario en qué lugar se encontraba en un determinado instante del pasado para ejercitar su mente.

Como hemos comentado, la tecnología GPS es la más precisa en exteriores y sobre todo, es global. Sin embargo su coste en baterías y su ineficacia en interiores hace que muestrear la posición de una persona a través de un móvil que integre esta tecnología se torne imposible dado que las baterías expirarían en pocas horas. De ese modo, desearíamos activar el GPS cuando salgamos a exteriores y lo desactivásemos en interiores. Por ello necesitamos detectar los siguientes dos eventos:

- *Entrada en interiores, fundamental para apagar el GPS y evitar el consumo innecesario.*

- *Salida a exteriores, para activar de nuevo el GPS y continuar posicionando de manera precisa al usuario.*

El primer evento es fácil de obtener, dado que la pérdida prolongada de la señal GPS (pérdida de la visión directa de los satélites) supone que hay un obstáculo sobre el receptor que evita la localización.

Sin embargo, el segundo evento es más complejo de detectar. Sería necesario un sistema de apoyo que no consuma demasiada energía. Las técnicas que proponemos son las siguientes:

- *Detección de puntos de acceso inalámbricos*. Detectando las intensidades de los diferentes puntos de acceso WiFi al entrar en un edificio techado, podremos buscar intensidades parecidas para saber cuándo volvemos a la entrada, es decir, cuando salimos.

- *Acelerometría*. Buscamos un desplazamiento, por lo que podemos estudiar los patrones de movimiento del dispositivo que suponemos lleva puesto el usuario.

- *Localización por técnicas de telefonía*. Esta técnica de localización la emplearemos integrándola con alguna de las técnicas anteriores, ya que la precisión por sí misma es muy baja pero, combinada con otra, se mostrará que hace posible que el sistema sea más eficaz. También veremos que pese a su baja precisión, la detección es muy fiable.

Dado el elevado número de tecnologías citadas, en primer lugar haremos un estudio del coste en baterías de las distintas alternativas, con el fin de determinar en base a estos resultados y a la eficacia de cada método, la tecnología más eficiente para la detección de salidas al exterior. Posteriormente, tras implementar el sistema, realizaremos pruebas de eficacia, que junto con las pruebas de eficiencia, darán el mejor sistema de detección.

En esta sección realizaremos una comparativa entre el consumo energético de las distintas tecnologías que pretendemos emplear. Esto será fundamental a la hora de elegir el mejor método de detección de salidas a exteriores dado que junto al rendimiento, la eficiencia energética será uno de los aspectos sobre los que basaremos nuestra decisión final.

Debemos destacar en primer lugar que las distintas gráficas sólo recogen el consumo del hardware asociado a las distintas técnicas sin que el software asociado a la detección entre en juego. Esta decisión se debe a que el software de fondo es común a todas las tecnologías, salvo determinadas modificaciones realizadas para la adaptación a cada una de ellas de manera concreta. De esta forma, al resultar constante el coste para todos los métodos, podemos abstraerlo de nuestra comparativa.

También tendremos en cuenta que todas las comparativas se han realizado sobre el mismo dispositivo, así que el consumo de elementos genéricos como por ejemplo el procesador, la memoria o la pantalla son comunes a todos los métodos. Para evitar deviaciones de la distribución de consumo frente a determinados parámetros ajenos a la comparativa, la prueba se ha llevado a cabo en un entorno de 10 paquetes de test. Cada uno de estos paquetes de test corresponde con un periodo de carga/descarga completo y el análisis del tiempo empleado en descargarse hasta el apagado del dispositivo.

La comparativa será hecha entre una serie de tecnologías que nos permitirán dotar al sistema de detección de salidas de la funcionalidad esperada. Gracias a estos sensores el sistema podrá obtener información del contexto de ejecución de la aplicación, entre cuyos parámetros se encuentra la localización del usuario. Concretamente, las tecnologías entre las que realizaremos la comparativa serán las siguientes:

- Redes inalámbricas (Wifi networks)
- Redes GSM de telefonía
- Acelerometría
- GPS

Todas las tecnologías empleadas están presentes en el dispositivo de manera interna. En ningún momento se ha empleado hardware adicional y, como se ha dicho anteriormente, todas las pruebas se han realizado partiendo de una carga inicial de 100% hasta llegar al punto en el que el dispositivo se apaga debido a la falta de batería.

Para evitar accesos innecesarios a la memoria durante el proceso de recopilación de datos, tan sólo se han recogido aquellos momentos en los que la batería disminuía su valor en un 10%. Por este motivo, la gráfica de consumo energético mostrada en la Figura 105 presenta una forma escalonada. Aun así, el consumo energético tiene un aspecto casi lineal, por lo que se ha procedido a una interpolación de puntos usando un polinomio de tercer grado. Esto nos permite observar sobreimpreso de una manera más clara, la tendencia energética del dispositivo en función de la tecnología empleada.

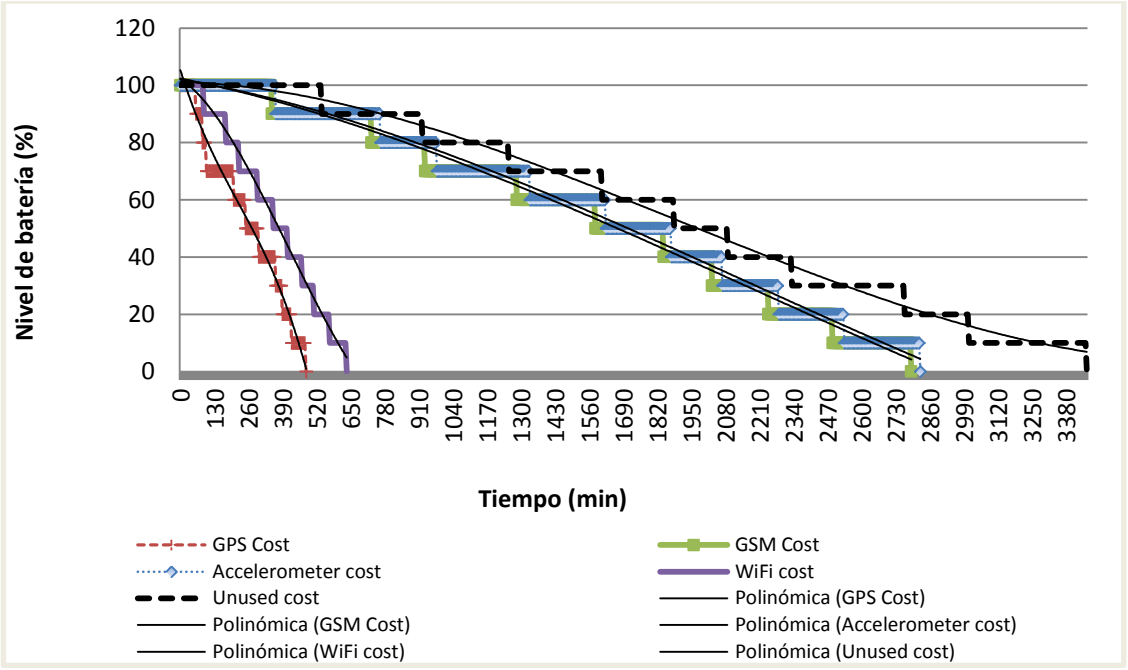


Figura 105: Gráfica de consumo energético de los distintos sensores presentes en un dispositivo móvil

Como podemos ver, la tecnología que más batería consume es el sensor GPS, como suponíamos a priori. El tiempo medio de vida de la batería empleando este sistema es de unas 7 horas, dependiendo de la cobertura de satélites. En el lado opuesto se encuentra el sensor de acelerometría, que debido a que funciona de manera continuada en el dispositivo, no posee ningún coste energético adicional, salvo el de la ejecución del propio software de detección. Por lo tanto, podríamos decir que el tiempo de vida aproximado de la batería debería coincidir en este caso con el tiempo que el dispositivo permanece encendido sin uso alguno. Si observamos la gráfica, este tiempo es de aproximadamente 60 horas. Sin embargo, el tiempo de vida especificado por el dispositivo es de 130 horas. Esta diferencia se debe al uso de la pantalla y diferentes recursos del propio sistema operativo empleados durante la ejecución del software de recopilación de datos.

En un punto intermedio se sitúan las tecnologías basadas en Wifi y GSM. En cuanto a la segunda, debemos decir que el tipo de cobertura telefónica empleada por el dispositivo (GSM, 3G o UMTS) influye en el coste energético del software. Este coste no se debe a la mera adquisición de datos, dado que los datos empleados para el posicionamiento se obtienen directamente de la señal recibida, sino por el coste adicional que provoca el envío de información por parte del dispositivo a la red de telefonía y la búsqueda de señal GSM válida continuamente.

Por último, fijémonos en el tiempo de batería empleando redes WiFi's. En este caso, la batería dura escasamente 11 horas, algunas horas más que empleando el dispositivo GPS, aunque su duración es bastante baja. Aun así, si podemos apoyarnos en la detección de redes WiFi para la localización de salidas al exterior, ganaríamos unas horas de batería que sería de gran importancia para el propósito que nos ocupa: el posicionamiento ubicuo. Sin embargo, esta técnica, aunque se consiga implementar con éxito, como veremos más adelante, no será una solución definitiva ya que el coste en baterías sigue siendo muy elevado para nuestro fin. Por lo tanto, veremos que tendremos que recurrir a técnicas combinadas para conseguir nuestro propósito.

Centrémonos ahora en el coste energético del receptor GPS y veamos cómo influye en él la cobertura de satélites. El objetivo de nuestro trabajo es, como bien sabemos, evitar gasto añadido de batería debido al uso del GPS en interiores. El punto de partida de nuestra investigación fue el alto consumo energético del receptor GPS en general y especialmente cuando se encuentra en interiores. Por ello, si desconectamos el receptor cuando esto ocurra, ahorraremos energía doblemente: por un lado el consumo normal del receptor de posicionamiento y por otro, el gasto añadido de no encontrar señal.

Es obvio que no podemos dar por hecho la suposición anterior tratándose de un artículo de investigación, por lo que a continuación nos centraremos en obtener el consumo de energía efectuado por el dispositivo GPS en función del nivel de cobertura. Si conseguimos demostrar que el consumo del GPS en interiores es más elevado que en el exterior con vista directa de los satélites, habremos demostrado la suposición que dio origen a la motivación del trabajo.

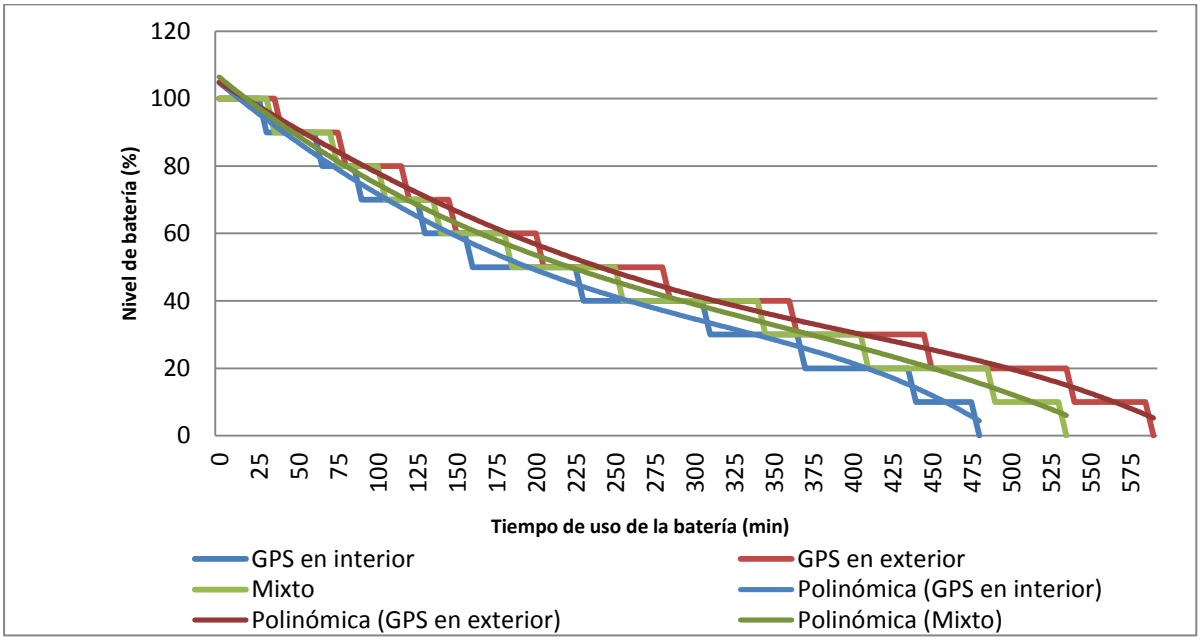


Figura 106: Gasto de energía producido por el sensor GPS en función de la posición del usuario

Con el fin de determinar de manera más clara la tendencia del consumo energético en los diferentes casos de la localización del dispositivo GPS, se refleja en la gráfica anterior las aproximaciones polinomiales de cada uno de los casos. De esta forma, en la gráfica se visualiza que el consumo energético del dispositivo GPS cuando éste se sitúa en exteriores es mucho menos elevado que cuando se encuentra en un lugar donde no existe visión directa con los satélites. Tanto es así, que la diferencia temporal en el tiempo de vida de la batería entre situar el dispositivo en el interior y en el exterior es de 115 minutos, es decir, casi 2 horas.

Debemos señalar que todas las medidas llevadas a cabo en este documento se han hecho sobre el software real empleado, de esta forma los resultados son totalmente empíricos. Esto es de especial interés en las conclusiones del tiempo de vida de la batería según las distintas técnicas, ya que gracias a ellas podemos ver claramente el impacto que tienen en un entorno real de la aplicación.

De este modo, una vez expuestos los diversos sensores que serán empleados a lo largo de este capítulo, se llevará a cabo un procedimiento a través del cual el sistema podrá determinar la salida al

exterior de un usuario. Dichos procedimientos estarán basados en los datos obtenidos a partir de los sensores anteriores, por lo que evidentemente, el tipo de sensor escogido en el método concreto de detección de salidas influirá en la eficiencia de dicha técnica. El uso de un sensor como el GPS conllevará un coste en baterías muy superior a si se emplea tan sólo el sensor de acelerometría del dispositivo.

JUSTIFICACIÓN DE LA DETECCIÓN DE SALIDAS

En esta sección haremos un estudio sobre el tiempo medio que una persona permanece en localizaciones interiores y exteriores a lo largo de un día. Gracias a este estudio podremos sacar conclusiones sobre la eficiencia de nuestro sistema. Si el tiempo que una persona permanece al aire libre es muy elevado, nuestro sistema de detección de salidas al exterior no tendrá un resultado muy acusado en el rendimiento, puesto que la localización se llevará a cabo continuamente mediante el sensor GPS. Sin embargo, si llegamos a la conclusión de que el tiempo que una persona permanece en interiores es mucho mayor que el que permanece en exteriores, tendremos la certeza de que el sistema de localización de salidas al exterior tendrá un impacto positivo en el consumo energético del dispositivo.

Para la realización del estudio se ha usado una plantilla repartida a 50 usuarios distintos. Esta plantilla consiste en una serie de notas que deben rellenar indicando la hora de salida o entrada a un recinto cerrado, con el fin de poder determinar mediante un sistema semiautomático el tiempo que permanece el usuario en recintos techados. Además, en el registro se debía incluir una descripción de la actividad realizada cuyo objetivo será evaluar la tipología de la actividad y determinar si podría haberse realizado también al aire libre. Para ello se ha llevado a cabo una aplicación que puede ser ejecutada en el dispositivo del usuario para facilitar la obtención de los datos. La interfaz de la aplicación se muestra en la Figura 107.

Figura 107: Interfaz de usuario de la aplicación para la recolección de datos de estancia en interiores

El tiempo en base al cual se han calculado los resultados es de siete días, de esta forma se recoge no sólo la actividad laboral, sino también actividades de ocio que los usuarios lleven a cabo algún día de la semana. De esta forma, los usuarios instalaron en sus dispositivos la aplicación mostrada anteriormente y anotaron todas las actividades realizadas durante este periodo de tiempo. A partir de las conclusiones obtenidas, se ha realizado una previsión del tiempo que el sistema de localización podría estar en funcionamiento si se aplicaran las técnicas desarrolladas en este capítulo. Sin embargo, esto se verá en la siguiente sección cuando se presenten los diversos métodos de detección de salidas.

En base a los resultados de las estadísticas, se ha obtenido la distribución temporal mostrada en la Figura 108, en la cual se presenta el porcentaje del tiempo total a lo largo de la semana que los usuarios pasan en exteriores. Todo ello se agrupa en tramos de edad, pudiendo ver así la clara dependencia entre la edad del usuario y el tiempo que pasa en exteriores. Sin embargo, estos resultados pueden verse influenciados por muchos motivos como por ejemplo, clase social, cultura, tipo de actividad laboral, perfiles de las aficiones, etc. Por lo que, pese a distar mucho de un estudio sociológico completo, cubre las necesidades pretendidas en este caso: dotar de una aproximación al tiempo que permanecen en interiores de los individuos de una población.

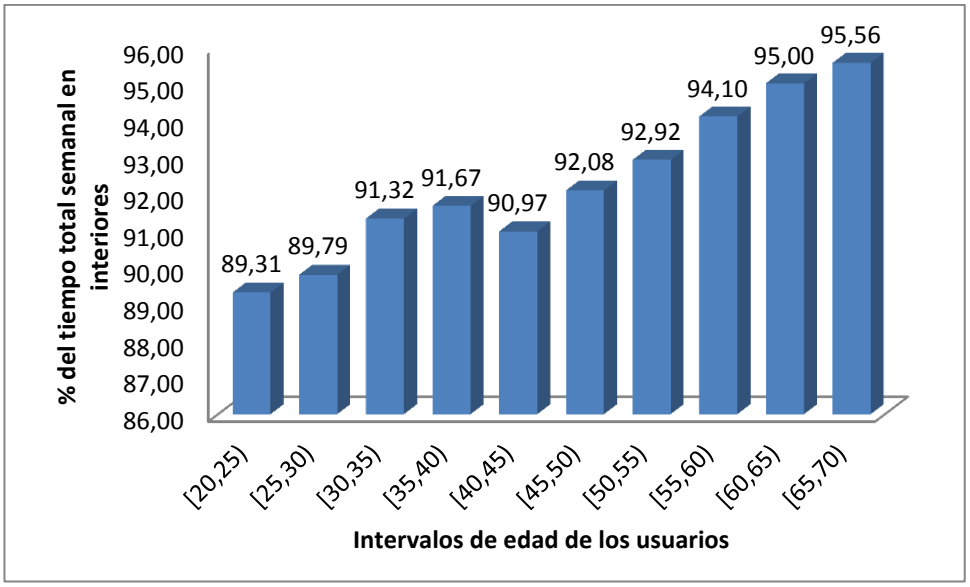


Figura 108: Tiempo de estancia de los usuarios en interiores agrupados por rangos de edad

Los resultados del análisis realizado aportan una conclusión inmediata: el tiempo que un usuario permanece en el interior de un recinto (casa, trabajo, gimnasio, etc.) es muy superior al que permanece en el exterior (al aire libre). El estudio da como resultado que una media del 92,27% del día, los usuarios permanecen en un entorno cerrado, frente al 7,73% que permanecen al aire libre. Es decir, los usuarios permanecen en el interior de un lugar alrededor de 22,14 horas diarias, y sólo 1,85 horas se encuentran en el exterior. Debemos tener en cuenta que el estudio se ha realizado a una población con edades comprendidas entre 20 y 70 años, por lo que, obviamente, los individuos más jóvenes permanecerán más tiempo en el exterior mientras los más ancianos, lo harán más tiempo en interiores. Todo esto puede verse claramente en la tendencia de la Figura 108.

Si realizamos un estudio más profundo sobre las tendencias según la edad, nos encontramos con que los individuos menores de 30 años pasan de media un 10,21% del día en exteriores (2,45 h/día), frente al 6,60% (1,2 h/día) que lo pasan los mayores de 60 años. No es el objetivo de este artículo

analizar los hábitos de los usuarios, aunque con los resultados obtenidos podemos afirmar rotundamente que el tiempo que los usuarios pasan en interiores es mucho mayor que el que pasan en exteriores. De esta forma, si logramos realizar un sistema que sea capaz de detectar las salidas y conectar el dispositivo GPS automáticamente, estaríamos hablando que el tiempo de funcionamiento del dispositivo pasaría a ser de 24 horas (todo el día funcionando y registrando datos de localización) a sólo unas 2 horas y media aproximadamente. De este modo, el consumo energético se vería reducido de una manera considerable.

Si reunimos los resultados obtenidos en este estudio con los obtenidos en el apartado de comparación de eficiencia energética de las tecnologías, podemos presuponer que reduciendo el uso del GPS en un 90% (tiempo medio que un usuario permanece en exteriores) y usando tecnologías auxiliares en su lugar, conseguiríamos ahorrar aproximadamente un 70% del consumo del dispositivo con esta tecnología. De esta forma, si con el GPS el tiempo estimado de vida de la batería era de 420 minutos, empleando tecnologías auxiliares tan sólo deberíamos hacer uso del sistema de localización durante 144 minutos, lo que indicaría que el tiempo de vida pasaría a ser de unos 2500 minutos, es decir, casi 2 días completos.

Sin embargo, la afirmación anterior no es del todo adecuada, ya que los distintos métodos de detección de salidas a exteriores poseen un coste computacional asociado, el cual repercutirá sobre el gasto energético del dispositivo. Por tanto, es esencial que el coste energético del sistema de detección no supere al del sistema GPS, ya que de lo contrario el estudio y las técnicas llevadas a cabo carecerían de sentido.

Teniendo en cuenta la consideración anterior, si combinamos las tecnologías que nos permitirán conocer cuándo el usuario ha salido al exterior con el sistema de posicionamiento GPS, podríamos estar hablando de que el tiempo de vida útil de la batería del dispositivo podría pasar de 7 horas (GPS) a más de 24 horas. Por tanto, si conseguimos realizar un sistema eficaz de detección de salidas al exterior, habremos conseguido lo que nos propusimos al comienzo del desarrollo, obtener un sistema de localización permanente cuyo consumo energético fuera lo suficientemente reducido como para realizar un registro de la actividad del usuario durante al menos un día completo, a lo cual denominaremos periodo de recarga.

CONSTRUCCIÓN DEL SISTEMA DE DETECCIÓN DE SALIDAS

Una vez introducido el trabajo que vamos a llevar a cabo, en esta sección procederemos a dar las ideas fundamentales sobre las que se basa el sistema que vamos a desarrollar. Para ayudar a clarificar las diferentes agrupaciones de métodos, se hará una clara diferencia entre las distintas tecnologías que serán empleadas para alcanzar los objetivos previstos. A grandes rasgos, se describirán un conjunto de sistemas basados en redes WiFi, acelerometría, GSM y por último un método resultante de la combinación de los anteriores.

SISTEMA BASADO EN REDES WIFI

El sistema de detección de exteriores basado en puntos de acceso WiFi consiste en una idea muy básica, marcar las redes inalámbricas detectadas en el momento en el que se detecta una pérdida de la señal GPS. Una vez marcadas las redes, comenzaría el periodo de búsqueda de salida. Este periodo consiste en la detección constante de las redes inalámbricas al alcance del dispositivo y la comparación de éstas con las redes marcadas en el momento en el que se perdió la cobertura. Cuando los dos

conjuntos de redes coincidan, llegaremos a la conclusión de que hemos vuelto al punto de partida en el que se perdió la señal y, por tanto, habremos detectado la salida al exterior.

En realidad, este proceso no es tan fácil como parece, ya que la fluctuación de la potencia de las señales inalámbricas hace que las señales con menor potencia aparezcan y desaparezcan de la cobertura en un lugar concreto, incluso podrían verse alterado el número de redes disponibles en un mismo punto en distintos instantes en caso de que se produjeran desconexiones de los puntos de acceso. Debido a esto, debemos desarrollar un sistema más avanzado y que posea una capacidad de reconocimiento del punto de salida más elevado.

Veamos ahora en profundidad la idea sobre la que se basa la implementación real mediante este sistema de detección de salidas. En primer lugar, como hemos dicho antes, debemos marcar las redes inalámbricas al alcance del dispositivo. Es decir, mediante una búsqueda conseguiremos obtener un listado de redes al alcance con sus respectivas potencias de señal. Esta búsqueda no se realizará una sola vez, sino que se harán varias búsquedas y se guardarán los conjuntos de redes detectadas. De esta forma, se obtendrá un conjunto de búsquedas que a su vez contendrán un conjunto de las redes encontradas con sus potencias e identificadores. Las búsquedas se realizarán de manera consecutiva, de forma que no transcurra excesivo tiempo entre ellas, con el fin de evitar una varianza excesiva de los datos.

De esta forma, los requisitos de información que necesitaríamos a la hora de elaborar el sistema basado en redes Wifi serían las redes inalámbricas, identificadas por la dirección física del adaptador (dirección MAC) y la información del nivel de intensidad de la señal medido en decibelios. La potencia de la señal puede obtenerse de manera sencilla a partir del sistema de búsqueda de redes WiFi que la mayoría de sistemas operativos para dispositivos móviles ofrecen. Sin embargo, la dirección física del adaptador no es posible obtenerla a menos que se establezca una conexión con éste, por lo que será necesario determinar otro identificador del punto de acceso. Para ello será seleccionado el SSID como identificador, de manera que la información necesaria por cada elemento resultante de una búsqueda vendrá denotada por la siguiente expresión:

$$wifiNet = \{SSID, I_{SSID}\}$$

Por otro lado el conjunto de redes encontradas en cada una de las búsquedas, consiste en una lista formada por toda la información de las redes halladas en el lugar y en un instante *t* concreto. Este último conjunto se notará de aquí en adelante como:

$$fingerprint_t = \{wifiNet_1, wifiNet_2, \ldots, wifiNet_m\},$$

donde *m* es el número de redes WiFi detectadas en cada ventana temporal. De este modo, *m* no es constante, sino que podría variar entre ventanas consecutivas debido precisamente a aquellas redes con menor potencia o cuyos puntos de accesos hayan sido desconectados. Un claro ejemplo de la variabilidad de *m* sería que se realizara una búsqueda en un momento dado, obteniendo por tanto un conjunto de redes. Al siguiente instante se vuelve a realizar otra búsqueda, pero justo antes el usuario cruzó una puerta. Esta puerta puede causar la amplificación de ciertas señales y la atenuación de otras, las cuales podría desaparecer.

Finalmente definiremos una ventana Wifi entre los instantes t_0 y t_n como las sucesivas huellas WiFi detectadas durante esos instantes y denotaremos como:

$$wifiWindow = \{fingerprint_{t0}, fingerprint_{t1}, \ldots, fingerprint_{tn}\}$$

En este caso *n* representa el número de ventanas que deben ser obtenidas para llevar a cabo posteriormente el proceso de búsqueda del sistema de detección de salidas.

Para aclarar este apartado pongamos un ejemplo. Imaginemos que hemos configurado la aplicación de reconocimiento de salidas a exteriores para que se realicen 3 búsquedas consecutivas. Pues en el momento en el que se pierda la señal GPS, el dispositivo procederá a realizar la primera búsqueda. El resultado de ésta será un conjunto de las redes al alcance en ese instante junto con las potencias de éstas. Una vez concluida la búsqueda (cuyo tiempo está en torno a los 5 segundos), se realizará otra, con su correspondiente resultado de redes encontradas, que no tiene por qué coincidir con el resultado de la primera búsqueda, es más, lo habitual es que no coincidan las intensidades y, en ocasiones, ni siquiera las redes encontradas. Tras otros 3 segundos aproximadamente (el tiempo de búsqueda de las redes) se procederá a la última búsqueda y al almacenamiento del conjunto de redes encontradas.

En la Figura 109 se recogen los valores correspondientes a un conjunto de redes detectadas durante un proceso de búsqueda de 2 minutos durante el cual el usuario ha efectuado un desplazamiento por el recinto. Como se puede observar, la variabilidad de los datos de las ventanas de búsqueda es significativa, hasta el punto que cuando la intensidad es baja, como la del punto de acceso WLAN_EA, la señal puede aparecer y desaparecer en un breve periodo temporal. Evidentemente, analizar los datos en bruto sería una tarea compleja y probablemente, llena de errores debido a la varianza de la muestra. Por este motivo será necesario trabajar con un conjunto de estadísticos que representen la marca de redes en un instante determinado.

La recopilación de estadísticas permite evitar la presencia de falsos positivos y falsos negativos en la detección, aumentando así la tasa de aciertos del sistema. Como indicábamos anteriormente, las intensidades de las redes (sobre todo de las redes inalámbricas domésticas) cambian constantemente y de manera elevada. Por este motivo, no podemos comparar directamente las redes almacenadas en el proceso de búsqueda con las redes actuales para obtener el punto de salida al exterior, sino que necesitamos ir un paso más allá. Con la obtención de la media de las lecturas de la potencia para cada una de las redes encontradas en los procesos de búsqueda, conseguiremos recopilar información sobre cada una de las redes y el rango de fluctuación de sus valores de intensidad, con el fin de generar un margen de error automático en función de las características de cada red encontrada.

Expliquemos esto de una manera más formal. Supongamos dos conjuntos (n conjuntos en general) formados por una serie de elementos (identificadores de la redes junto a la intensidad de la señal) todos ellos distintos en cada uno de los conjuntos. Una vez generados los conjuntos, buscaremos aquellos elementos comunes entre ellos, tomando como elemento común aquel par de elementos (n elementos como máximo para el caso de n conjuntos) que posean la misma identificación (dirección física MAC). Para cada grupo de elementos comunes generaremos la siguiente información:

- Media aritmética de I_{SSID}: $\overline{I_{SSID}}$
- Desviación típica de *fingerprint*: $\overline{S_{SSID}}$

Una vez calculados estos valores para cada una de las redes de los conjuntos de búsqueda, los resultados de ambas medidas serán almacenados para cada una de ellas, dado que en el proceso de búsqueda del punto de salida recurriremos a estos valores para identificar dicho punto.

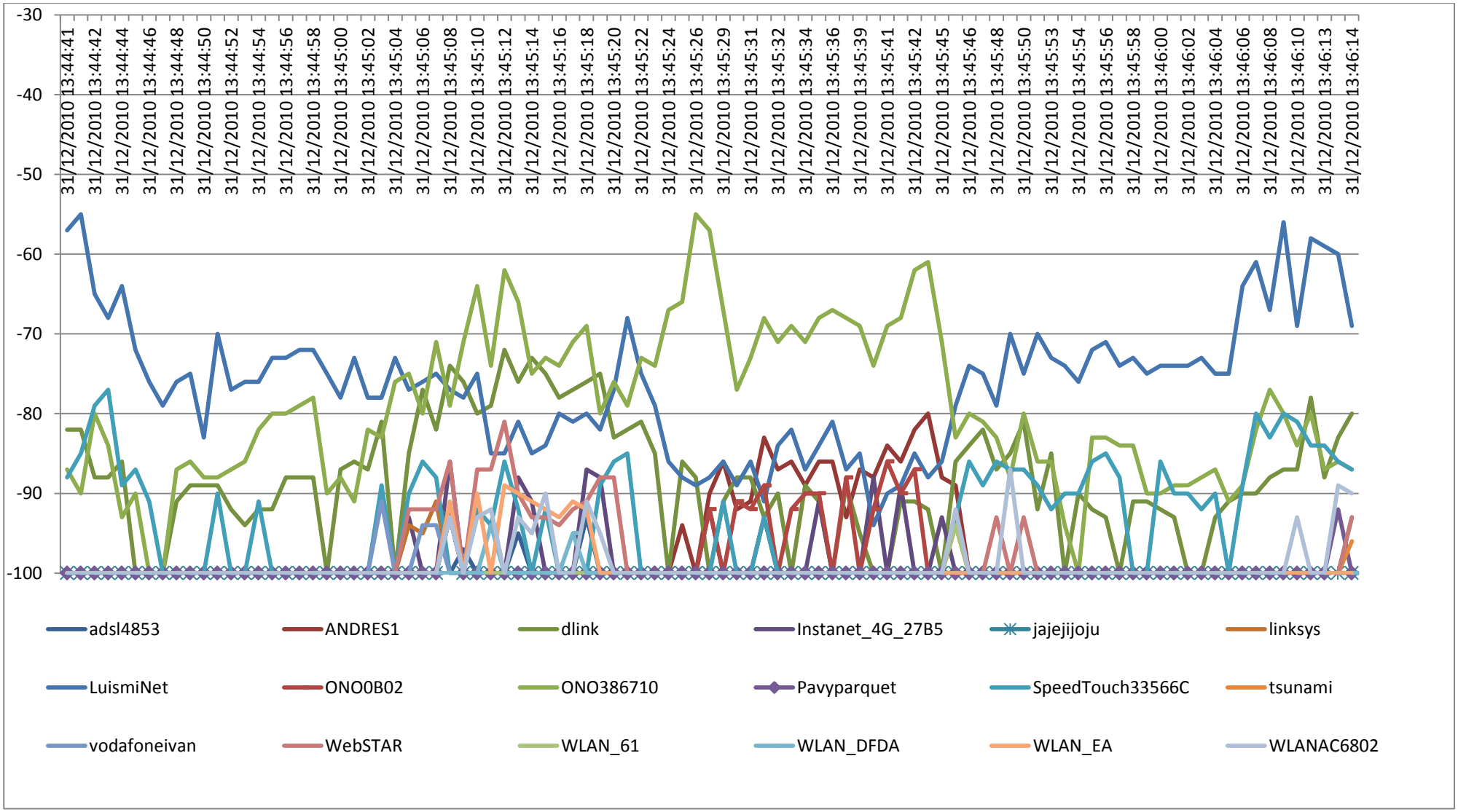


Figura 109: Valores de intensidad para una búsqueda de 2 minutos ininterrumpida

El hecho de emplear el SSID como identificador del punto de acceso supone un problema debido a que este valor puede no ser único para cada red del conjunto de búsqueda. En este sentido, se podría dar la circunstancia de hallar dos redes con el mismo identificador y distinta intensidad. Este tipo de duplicidad es muy habitual en redes institucionales o empresariales, en las que sus puntos de acceso están replicados con el mismo nombre. Este problema puede evitarse mediante el uso de la selección de la distancia media mínima basada en lecturas anteriores de dichas redes. De este modo, si se obtiene un punto de acceso con una determinada intensidad $Intensity_a$ y existen dos huellas con el mismo SSID, las cuales tienen unas intensidades $\overline{I_{SSID1}}$ y $\overline{I_{SSID2}}$ respectivamente, la lectura coincidirá con $fingerprint_1$ si y sólo si:

$$|\overline{I_{SSID1}} - Intensity_a| < |\overline{I_{SSID2}} - Intensity_a|$$

Este método se emplea tanto en el proceso de marcado como en el proceso de reconocimiento de salidas. A través de este sistema, el error producido en la plataforma de detección de salidas a causa de SSID duplicados se ve reducido.

Volviendo de nuevo al proceso de marcado y a la obtención de estadísticos asociados, supongamos que hemos configurado la aplicación para que se realicen 3 búsquedas. El resultado de los conjuntos de búsqueda se puede ver en la Tabla 9.

Momento	SSID	I_{SSID} (dB)
1	A2341f3F	-40 dB
	B112231F	-80 dB
	11132323	-20 dB
2	A2341f3F	-30 dB
	B112231F	-90 dB
	11132323	-10 dB
	23423424	-30 dB
3	A2341f3F	-40 dB
	B112231F	-80 dB
	11132323	-40 dB

Tabla 9: Resultado del proceso de búsqueda para un conjunto de redes WiFi

Cada momento hace referencia a un proceso de búsqueda independiente. A partir de estos conjuntos con las redes y sus correspondientes intensidades se obtienen los estadísticos descritos anteriormente, los cuales se muestran en la Tabla 10.

SSID	$\overline{I_{SSID}}$ (dB)	S_{SSID} (dB)
A2341f3F	-37 dB	4,72 dB
B112231F	-83 dB	4,72 dB
11132323	-23 dB	12,47 dB
23423424	-30 dB	0 dB

Tabla 10: Estadísticos asociados al resultado de la búsqueda de redes

Una vez obtenidos los valores de la media y la desviación típica de la señal para cada una de las redes detectadas, se aplicarán una serie de filtros con el objetivo de minimizar el error producido por la detección. El primero de esto filtros consiste en eliminar un porcentaje del total de redes detectadas con mayor desviación típica. Dicha desviación será calculada en base a la tabla anterior generada tras

el proceso de búsqueda de redes WiFi. Para la aplicación del filtro expuesto es necesario ordenar el conjunto de redes almacenadas en orden creciente en base a su desviación, y eliminar la últimas *P* instancias, donde *P* viene dado por el porcentaje de eliminación de redes impuesto

El valor óptimo del porcentaje de eliminación ha sido calculado a partir de un conjunto de pruebas empíricas, dando como resultado que la máxima precisión del sistema se obtiene para un coeficiente de eliminación del 80%. De este modo, de cada 10 redes, las dos cuya desviación sea más elevada serán eliminadas del conjunto de redes marcadas.

La detección del punto de salida se basará en las redes y los valores estadísticos almacenados y la comparación de éstos con los de las redes detectadas en cada instante. Para determinar si una red encontrada coincide con una red almacenada se debe cumplir el siguiente criterio de similitud: la diferencia entre la potencia media de la red almacenada y la potencia de la red encontrada debe ser menor que la desviación típica de dicha red en el momento de la búsqueda.

Si finalizamos aquí nuestro algoritmo de búsqueda de salidas al exterior, llegaríamos a la conclusión de que el número de falsos negativos es muy elevado. Esto se debe a que la función de pertenencia de las redes encontradas al grupo de las almacenadas es demasiado restrictiva. Debemos encontrar algún otro parámetro que suavice las restricciones y haga que los falsos negativos tiendan a desaparecer, ya que esto implicaría una salida al exterior no detectada por el dispositivo, por lo que no se reactivaría la recepción de señal GPS. Posteriormente se analizará el impacto negativo que supone la aparición de un falso negativo en el sistema, lo cual es mucho más grave que la aparición de un falso positivo. Esta adecuación del criterio de pertenencia se llevará a cabo mediante la inclusión de un margen de error predefinido y un porcentaje de acierto determinado.

El margen de error se empleará a la hora de determinar la pertenencia de una red al conjunto de redes almacenadas. El valor vendrá dado en decibelios y hará menos restrictiva la condición de la que hablábamos anteriormente. De esta manera, una red pertenecerá al conjunto de redes almacenadas si la media de las intensidades obtenidas en el proceso de búsqueda para la red menos la potencia de la red en el momento de la detección es menor que la media de la deviación de la red respecto a la media en el momento de la detección multiplicado por el margen de error establecido.

Definición: Se dice que una red detectada está representada en el conjunto marcado siempre y cuando se cumpla la siguiente condición:

$$R_j \in C_i \leftrightarrow \frac{|\, I\Delta_{SSID} - \overline{I_{SSID}}|}{S_{SSID}} < \varepsilon,$$

donde R_j es una red detectada en el momento actual y C_i es el conjunto formado por todas las apariciones de la red actual (es decir, redes con la misma SSID que R_j) en la ventanaWifi generada en el proceso de marcado del punto de salida. Por último $I\Delta_{SSID}$ representa la intensidad de la lectura de la red R_j.

El hecho de dividir la diferencia entre $I\Delta_{SSID}$ y $\overline{I_{SSID}}$ por la deviación típica es necesario para proporcionar un comportamiento dinámico al sistema de detección de salidas. De esta forma, si la varianza de una red es muy elevada, el margen de error (ε) será más elevado que para el caso de una red cuya varianza sea mínima.

Con esto habremos evitado el problema de las fluctuaciones de intensidad de las redes inalámbricas. Tanto el margen de error adaptativo (el basado en la media) y el constante dan muy buen

resultado, como será demostrado más adelante en la sección de resultados del sistema. De nuevo en la sección de pruebas y resultados veremos una comparativa del nivel de precisión del sistema de detección de salidas en función del parámetro margen de error (ε). Sin embargo, podemos vaticinar que tras realizar una batería de pruebas, se ha llegado a la conclusión de que el mejor ε para el sistema es 1,7.

Hemos solventado el problema de la intensidad variable de las redes pero, sin embargo, aún tenemos un problema, las redes más débiles tienden a desaparecer en los procesos de detección, incluso podría producirse una caída de una red inalámbrica tras el proceso de búsqueda, por lo que en el proceso de escaneo constante, esta red nunca estaría disponible. El problema de redes fantasma o redes que desaparecen lo resolveremos mediante el porcentaje de acierto.

Este porcentaje consiste en un valor que indica la relación mínima entre el número de redes almacenadas durante el proceso de búsqueda en relación con el número total de redes almacenadas durante el proceso de marcado y el conjunto de éstas que pertenecen al conjunto de redes marcadas siguiendo el criterio de pertenencia anterior. Si esta relación es mayor que el umbral de relación dispuesto, indicará que ambos conjuntos son coincidentes, por lo que el sistema informará del encuentro del punto de entrada y activará el dispositivo GPS. En cambio, si el factor de relación no alcanza dicho mínimo, el sistema desechará el conjunto escaneado y procederá con la búsqueda.

Definición: Un punto será interpretado como punto de salida cuando el número de redes detectadas en el momento actual que cumplen la función de pertenencia expuesta medida en tantos por ciento, sea mayor que el porcentaje de acierto definido estáticamente, es decir:

$$isExitPoint(C) = 1 \leftrightarrow \frac{(\sum_{i=0}^{n} belongSet(C_i))}{n} > \delta$$

donde δ es el porcentaje de acierto y

$$belongSet(C_i) = \begin{cases} 1, & if\ \exists R_j \in R\ /R_j \in C_i \\ 0, & otherwise \end{cases}$$

También hemos deducido en base a los resultados y pruebas realizadas que un valor adecuado para el porcentaje de acierto δ es 70, es decir, al menos el 70% de las redes detectadas en el proceso de marcado del punto de salida deben coincidir con las detectadas en el momento actual.

La elección de los parámetros δ y ε como hemos dicho, no es arbitraria sino que se basa en un conjunto de test que se ha realizado a partir de unas pruebas llevadas a cabo a partir de la recopilación de datos desde los dispositivos de los usuarios sobre los que se han realizado las pruebas de la aplicación. Dichos resultados pueden verse expresados en las ilustraciones que se explicarán a continuación.

En primer lugar, debemos comentar que no tiene sentido estudiar los factores δ y ε de manera independiente, ya que ambos influyen en la precisión del reconocimiento. En su lugar, realizaremos este estudio agrupando los resultados obtenidos para cada uno de ellos para la media aritmética del otro indicador. Es decir, si queremos estudiar cómo se comporta el porcentaje de errores totales en función del parámetro ε, el porcentaje de error para un determinado ε vendrá determinado por la media aritmética de los errores para todos los valores obtenidos, cuyo margen de error sea ε y para todo .

Siguiendo la idea anterior, comenzaremos estudiando el porcentaje de aciertos del sistema en función del valor de δ. Para comprender los resultados presentados en la Figura 110 debemos indicar que el porcentaje de acierto tan sólo refleja aquellos fallos relacionados con falsos negativos, es decir, circunstancias en las que un usuario ha realizado una salida y el sistema no la ha reconocido correctamente. De esta forma, en ningún momento se tiene en cuenta la influencia de falsos positivos (detección de salidas que no son tales) y por tanto, no indica la energía ahorrada por el uso del sistema. Por tanto, si analizamos la influencia de δ en el porcentaje de aciertos del método, se puede observar que a mayor valor de δ la influencia sobre el porcentaje de aciertos es negativa. Esto es evidente debido a que el parámetro estudiado indicará el porcentaje de redes del total de las detectadas que deberán coincidir como mínimo con el conjunto almacenado para concluir que se ha producido una salida a exteriores. Si este porcentaje está próximo a 0, indica que no es necesario que coincida ninguna red, por lo que obviamente el porcentaje de acierto será el 100%, todo lo contrario de si el valor de δ está próximo a 1, lo que indica que será necesario que el 100% de las redes coincidan. En este último caso, el criterio a satisfacer se endurece, por lo que el porcentaje de aciertos bajará.

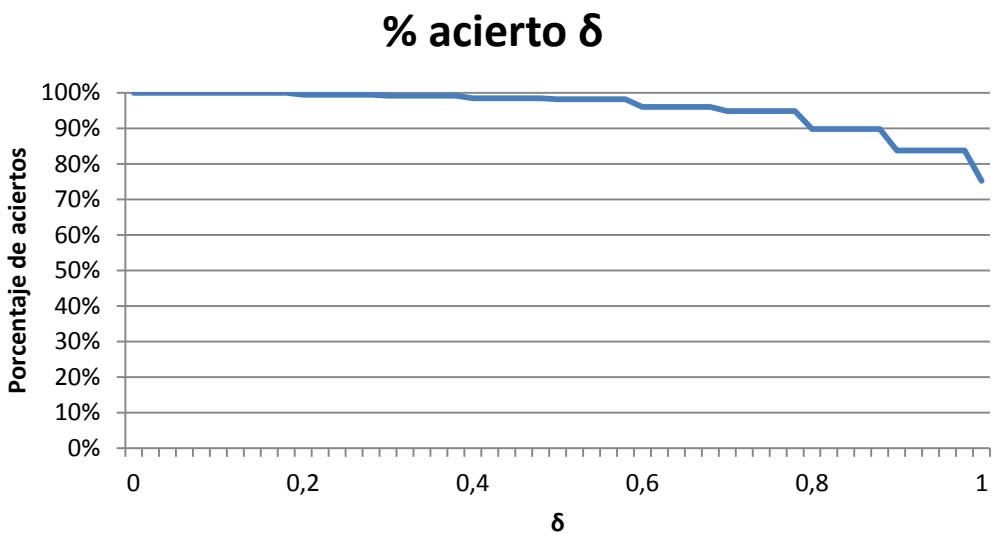


Figura 110: Porcentaje de acierto del método de detección de salidas en función del parámetro δ

Una pregunta que podemos hacernos en base al resultado anterior sería ¿por qué no seleccionar δ igual a 0 si el porcentaje de aciertos es el 100%? La respuesta a esta pregunta puede resumirse en la Figura 111. El porcentaje de falsos positivos indicará la relación de salidas incorrectas detectadas, es decir, salidas que en realidad no se han producido pero que el sistema ha determinado que sí lo eran. De este modo, el indicar que δ sea igual a 0, influirá positivamente sobre el porcentaje de aciertos, pero como se puede observar en la figura, dispara el número de falsos positivos, llegando a obtener valores superiores al 90%. Este valor indica que 9 de cada 10 salidas detectadas fueron erróneas, ya que tan sólo una de ellas fue efectivamente una salida real.

El problema anterior no sería tal si no intentásemos minimizar el coste en baterías del sistema de posicionamiento. El hecho de activar el GPS cada vez que el sistema detecta una posible salida a exteriores, debe manejarse con gran cuidado, ya que numerosas activaciones consecutivas del GPS sin necesidad de hacerlo debido a que no se ha producido una salida real, puede no sólo eliminar toda la eficiencia del sistema propuesto, sino aumentar el consumo de baterías respecto a la alternativa de dejar el GPS encendido constantemente.

Dicho lo anterior, debemos llegar a un equilibrio entre el porcentaje de aciertos total del sistema y el número de falsos positivos obtenidos. El valor óptimo será descrito más adelante, cuando sean expuestos todos los datos necesarios para elegir dicho valor.

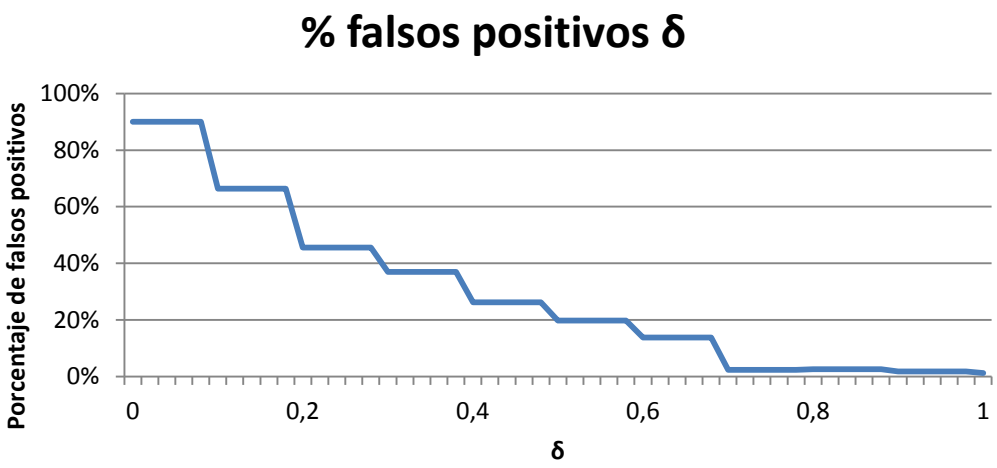


Figura 111: Porcentaje de falsos positivos del método de detección de salidas en función del parámetro δ

Continuando con el análisis del parámetro δ, podemos visualizar a través de la Figura 112 el porcentaje de falsos negativos producidos a medida que se incrementa el valor de éste. A diferencia del caso anterior, la imposición de que se cumpla la pertenencia de un mayor número de redes al conjunto de redes almacenadas, implica que el sistema será más restrictivo a la hora de determinar que se ha producido una salida. Dicha restricción puede llegar hasta tal punto que en numerosas ocasiones se produce una salida real que el sistema no ha detectado, debido a que la fluctuación de las intensidades o la desaparición de alguna red haya influido en demasía sobre la decisión. Mientras que los falsos positivos influían en el ahorro energético del sistema, los falsos negativos influyen directamente en la viabilidad, puesto que un sistema excesivamente restrictivo dejaría pasar una salida sin activar el sensor GPS, perdiendo así quizás la oportunidad de volver a reactivarlo.

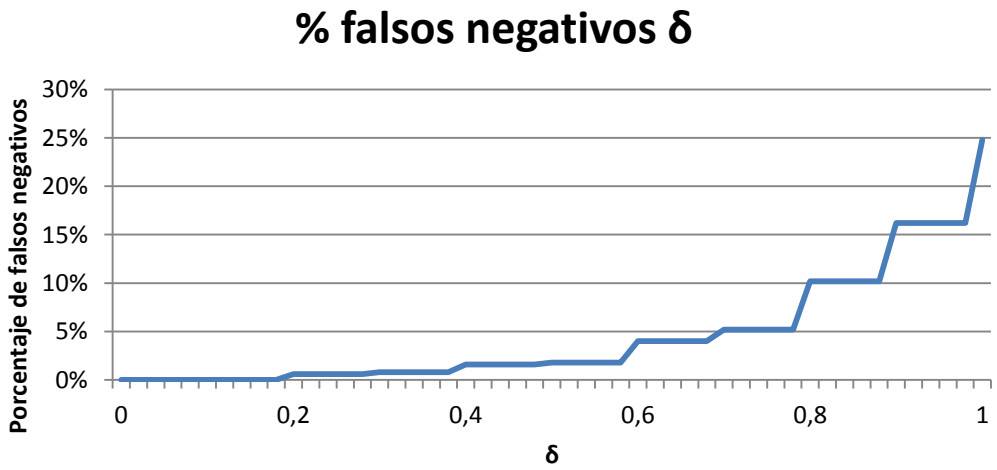


Figura 112: Porcentaje de falsos negativos del método de detección de salidas en función del parámetro δ

Una vez analizados los falsos positivos y los falsos negativos, haremos un estudio sobre el porcentaje total de errores. Este porcentaje vendrá determinado por la sumo de los errores estudiados anteriormente. En la Figura 113 podemos ver manera clara la evolución del error total frente a cambios del parámetro δ. Si analizamos detenidamente la forma de la curva resultante en la figura, podemos observar que posee forma de "U", es decir, posee un mínimo en un valor en torno al cual las tendencias sobre crecientes. Esto se debe a la influencia por un lado de los falsos positivos, los cuales aparecen a la izquierda del mínimo, y la influencia de los falsos negativos, a la derecha del mínimo. En este sentido, sería interesante ajusta el valor de δ a aquel en el que la relación entre falsos positivos y falsos negativos sea óptima. Precisamente el valor que estamos buscando es el mínimo de la función representada en la Figura 113, el cual se produce para δ igual a 70%, donde el porcentaje de falsos positivos es 2% y el de falsos negativos 3%. Este valore será precisamente el que emplearemos a lo largo de todo el proceso para la optimización de la precisión del sistema de detección de salidas.

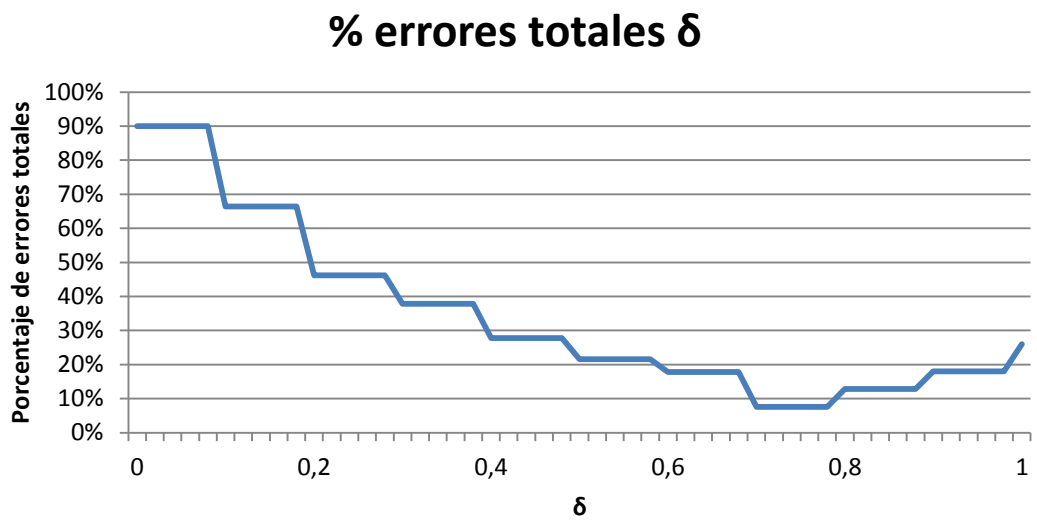


Figura 113: Porcentaje de errores totales del método de detección de salidas en función del parámetro δ

Paralelamente al estudio del parámetro δ, debemos estudiar el comportamiento del sistema para la modificación del parámetro de error variables (ε). Al igual que en el caso anterior, comenzaremos estudiando el porcentaje de aciertos media para cada valor de ε. Este análisis puede verse ilustrado en la Figura 114, en la cual se observa que la tendencia de la función de porcentaje de acierto es creciente, lo que indica que a mayor valor de ε, el porcentaje de acierto será más elevado. Esto, al igual que en el caso anterior cuando estiábamos el parámetro δ, es intuitivo, dado que si reflexionamos acerca del significado de ε en el sistema nos daremos cuenta que cuanto mayor sea, mayor será la distancia determinada como asumible entre la intensidad de una red marcada durante el proceso de entrada a interiores y la intensidad de la misma red en el momento de la búsqueda. El aumento de este parámetro, por tanto, hará menos restrictivo el criterio de pertenencia y de esta forma la cantidad de redes marcadas como detectadas será mayor.

Aunque el estudio anterior puede verse de manera clara en la Figura 114, la varianza de ε con respecto a δ hace que la media aritmética, que es el valor tomado como referencia, sea semejante. Esto se debe a la influencia de los valores extremos en el valor de la media para cada valor de .

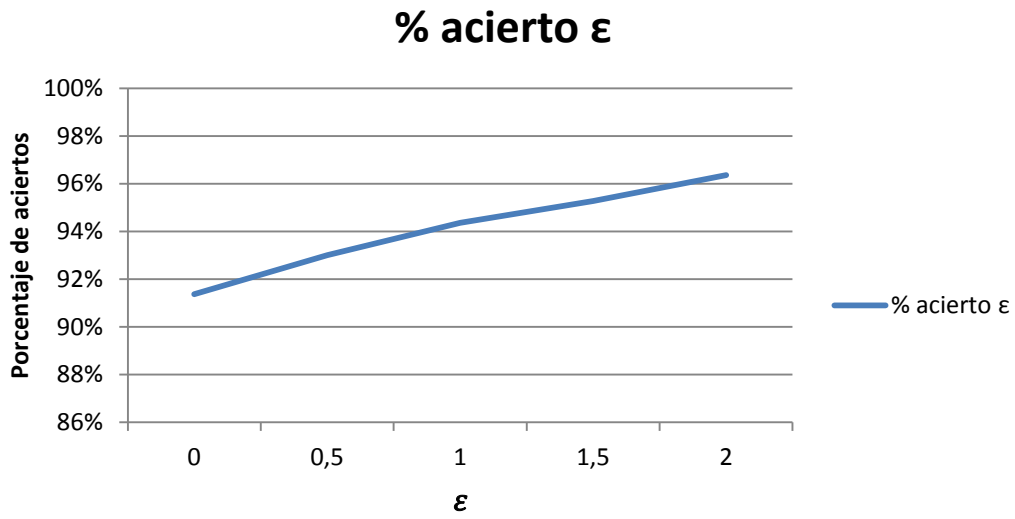


Figura 114: Porcentaje de acierto del método de detección de salidas en función del parámetro ε

A continuación estudiaremos el número de falsos positivos, los cuales aumentan al incrementar el valor de *ε* como se puede observar en la Figura 115. Este aumento se debe a que al aumentar la variabilidad admitida entre el conjunto marcado y la red escaneada, el criterio de pertenencia se hace más laxo, por lo que el número de redes reconocidas como parte del conjunto marcado aumenta. Este aumento hace que el número de salidas detectadas aún sin haberse producido aumente. Como se ha comentado en el caso anterior, este tipo de error no es crítico, aunque afecta negativamente sobre el rendimiento y la eficiencia energética del dispositivo.

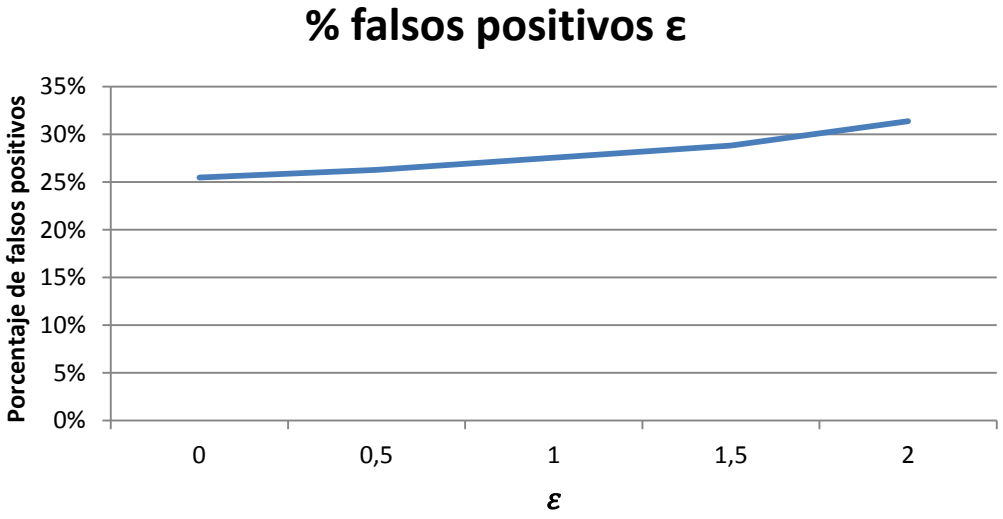


Figura 115: Porcentaje de falsos positivos del método de detección de salidas en función del parámetro ε

Para continuar con el análisis del parámetro ε, se determinará el porcentaje de falsos negativos en función de dicho parámetro. Esto, que se puede ver ilustrado en la Figura 116, se traduce en un decremento del número de falsos positivos a medida que aumenta el valor de ε, justo a la inversa que en el caso del análisis de falsos positivos. Sin embargo, como puede verse en la ilustración, el

porcentaje de falsos negativos independientemente del valor de ε es relativamente bajo, pero al tratarse de un fallo crítico deberemos prestar especial atención a este resultado a la hora de escoger un valor idóneo de ε.

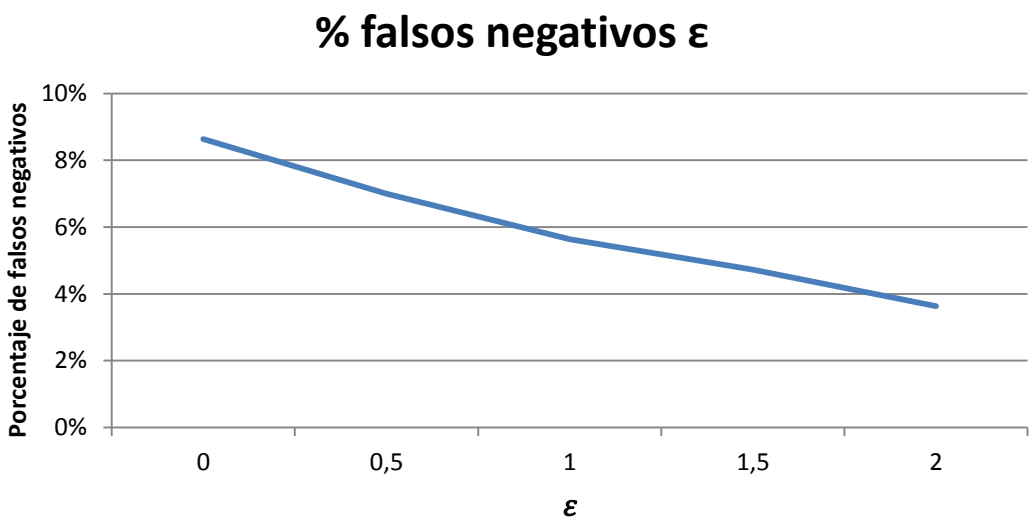


Figura 116: Porcentaje de falsos negativos del método de detección de salidas en función del parámetro ε

Por último, veremos la relación existente entre el número de falsos positivos y el de falsos negativos, con el fin de determinar un valor óptimo para ε en función de estos porcentajes. La relación entre ambos valores se muestra en la Figura 117, en la que se observa la existencia de un mínimo entre los valore 1 y 1.5, lo cual indica que en este intervalo se produce el valor mínimo para la suma de falsos positivos y falsos negativos del parámetro ε.

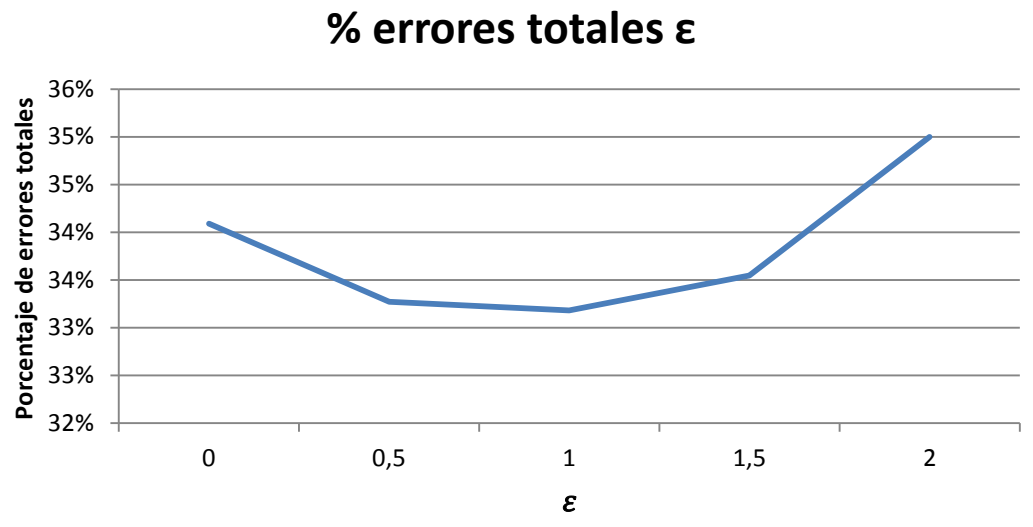


Figura 117: Porcentaje de errores totales del método de detección de salidas en función del parámetro ε

Resultante de este estudio, el valor óptimo de ε se establece a 1.5, a pesar de producirse el mínimo para el valor 1. Este cambio del valor de ε respecto a la ilustración se debe a que, estudiando los datos

conjuntos para los valores de ε y δ se produce un error global del 5% para un ajusta de $\delta = 0.7$ y $\varepsilon = 1.5$. Este mínimo puede observarse en la Figura 118, donde en torno a $\delta = 0.7$ y $\varepsilon = 1.5$ la curva representada toma un valor más oscuro, el cual indica que se produce un máximo en este punto. Debemos tener en cuenta que los valores óptimos se calculan en base al porcentaje de error total, es decir, se ponderan de igual manera tanto los falsos positivos como los falsos negativos, puesto que el error total no es más que la suma de ambos.

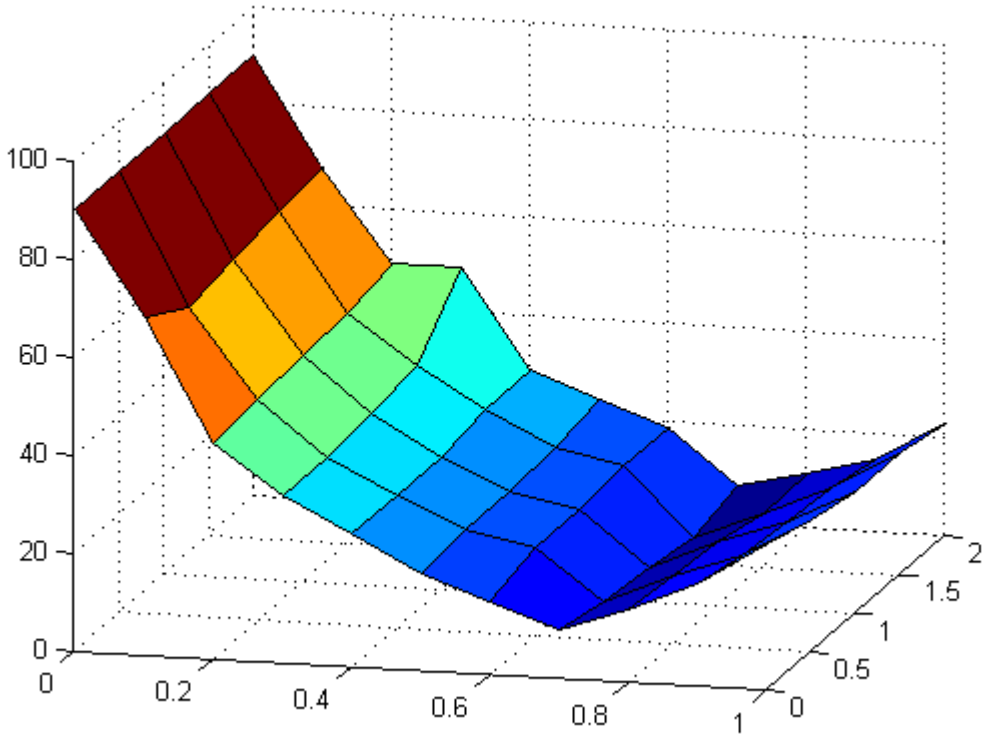


Figura 118: Representación tridimensional del porcentaje de errores totales respecto al valor de δ y de ε

A continuación expondremos dos ejemplos de detección de redes WiFi y clasificación de salida a exteriores. En el primer ejemplo, representado en la Tabla 11, vemos como 3 de las 4 redes detectadas cumplen con los criterios que determinan si una red pertenece o no al conjunto de redes almacenadas, por lo que el porcentaje de acierto de las redes es de un 75%. Este criterio, como se ha visto anteriormente, dependerá de la desviación de la muestra para la red en concreto y el valor de intensidad de la propia red. Puesto que el porcentaje de acierto mínimo está definido en un 70% (δ=0,7), podemos decir que el punto en el que se detectó esa lectura de redes coincide con el punto que se firmó a la pérdida de la señal GPS.

Sin embargo, en el segundo ejemplo, el cual se ilustra en la Tabla 12, no ocurre lo mismo. Dos de las cuatro redes no cumplen el criterio de pertenencia dado que la desviación respecto a la media es superior a la admitida, por lo que el porcentaje de acierto es de un 50%. Dicho porcentaje es menor que el mínimo para que el punto se considerara como punto de salida.

SSID	$I\Delta_{SSID}$	Desviación respecto a la media	Desviación admitida	Pertenece al conjunto
A2341f3F	-40 dB	3 dB	8 dB	Sí
B112231F	-81 dB	2 dB	8 dB	Sí
11132323	-23 dB	0 dB	21,2 dB	Sí
23423424	-32 dB	2 dB	0 dB	No
		$\delta = 0{,}7$	$\varepsilon = 1{,}7$	Salida detectada (75%)

Tabla 11: Ejemplo de un proceso de búsqueda exitosa de un punto de entrada marcado previamente

SSID	$I\Delta_{SSID}$	Desviación respecto a la media	Desviación admitida	Pertenece al conjunto
A2341f3F	-47 dB	10 dB	8 dB	No
B112231F	-81 dB	2 dB	8 dB	Sí
11132323	-23 dB	0 dB	21,2 dB	Sí
23423424	-32 dB	2 dB	0 dB	No
		δ=0,7	ε=1,7	Salida no detectada (50%)

Tabla 12: Ejemplo de un proceso de búsqueda exitosa de un punto de entrada marcado previamente

Una vez explicado detenidamente en qué consiste el método de detección de salidas a exteriores a través del marcado de señales WiFi, en la Figura 119 se puede ver de forma esquematizada el proceso de detección de salidas con el fin de resumir lo explicado anteriormente y tener una idea más general de la técnica empleada. En primer lugar, el sistema activará el dispositivo GPS y desconectará el subsistema de búsqueda de señales WiFi. Una vez el dispositivo GPS haya proporcionado una señal válida, se seguirá procesando la información GPS como dato de localización, hasta que se pierda dicha señal. A la pérdida de la señal, el sistema realizará el proceso de marcado de señales inalámbricas, para lo cual activará el subsistema WiFi y almacenará las redes almacenadas para todas las búsquedas llevadas a cabo, calculando para cada una de las redes los estadísticos descritos anteriormente.

Podría darse el caso de que no existiera ninguna señal WiFi en la zona, por lo cual no podría marcarse el punto de entrada y la única opción válida será reactivar de nuevo el subsistema GPS. En este caso el sistema propuesto no mejoraría la eficiencia energética del dispositivo, puesto que se optará por una política de ahorro de siempre encendido.

En caso de que existan señales WiFi en la zona, se marcará el punto y se desconectará el sensor GPS, momento en el cual comenzará el proceso de detección de salida. Durante este proceso, el subsistema WiFi realizará lecturas continuas de las redes WiFi disponibles, de manera que en cada una de estas búsquedas se procesará el conjunto de redes obtenidas y se comparará con los datos procesados en el momento del marcado del punto de salida. Si ambos conjuntos son coincidentes según el criterio expuesto anteriormente, el sistema habrá determinado que se ha producido una salida, por lo que se reactivará de nuevo el sistema GPS.

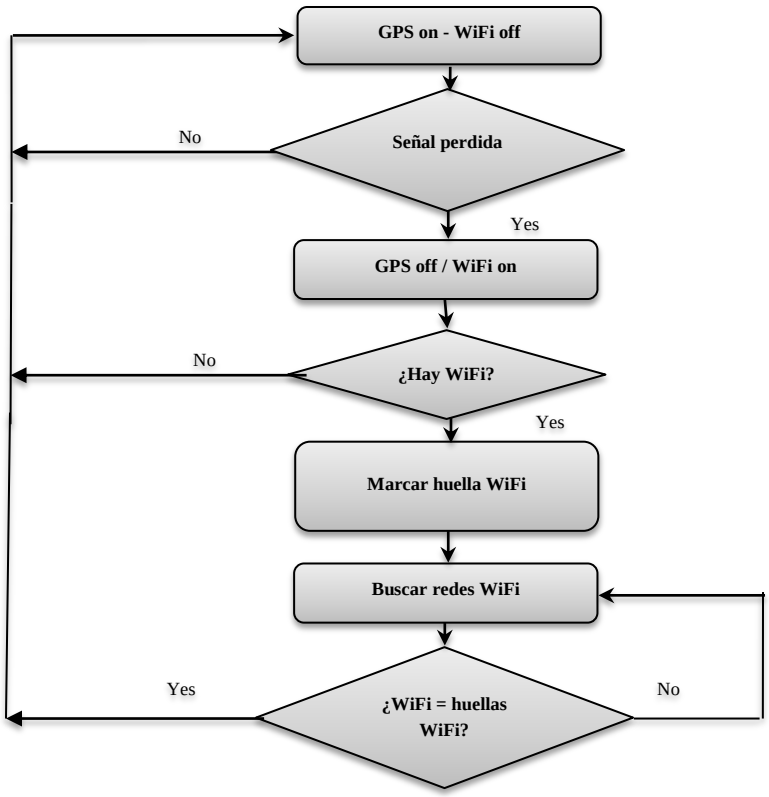


Figura 119: Diagrama de control del sistema de detección de salidas mediante redes WiFi

Mejora de la eficiencia del método basado en redes WiFi

Se comentaba anteriormente que el principal problema que presentaba el método de detección de salidas a exteriores basado en redes WiFi es la escasa mejora en la eficiencia energética aportada al dispositivo. Sin embargo, aunque esto será resuelto mediante el empleo de métodos adicionales al actual, introduciremos a continuación una sencilla acción que puede ahorrar entre un 30 y un 60% de batería a la hora de detectar salidas mediante redes WiFi.

El método anterior basado en puntos de acceso realizaba una búsqueda intensiva y constante de redes inalámbricas con el objetivo de detectar patrones de coincidencias entre el punto de entrada marcado y la posición en la que se encuentra el usuario. Con variante presentada a continuación, pretendemos minimizar el número de búsquedas de redes, ya que es precisamente el motivo del gasto energético del método anterior. Esto se hará definiendo una *distancia estimada* entre el punto de entrada marcado y el punto en el que se encuentra el usuario. Dicha distancia se calculará en base a las redes obtenidas y a la diferencia entre la intensidad en el punto de entrada marcado y en el punto actual.

Puesto que anteriormente se definió el criterio de pertenencia de una determinada red a un conjunto de redes marcadas de la siguiente forma:

$$R_j \in C_i \leftrightarrow \frac{|I\Delta_{SSID} - \overline{I_{SSID}}|}{S_{SSID}} <,$$

Reutilizaremos el cociente anterior para determinar un criterio de vecindad entre el punto marcado con la intensidad $\overline{I_{SSID}}$ y el punto actual determinado por la intensidad $I\Delta_{SSID}$. En este caso, en lugar de comparar el cociente anterior con el margen de error estático ε, acumularemos los cocientes anteriores para todas las lecturas obtenidas, y definiremos una tabla de relación entre el índice obtenido y el tiempo entre búsquedas de redes WiFi.

De esta forma, el índice de coincidencia para todas las redes marcadas vendrá determinado por la siguiente expresión:

$$IC = \frac{1}{n}\sum_{i=0}^{n}\frac{\left| I\Delta_{SSID_i} - \overline{I_{SSID}}_i \right|}{S_{SSID\,i}}$$

donde n representa el número de redes detectadas en la búsqueda actual, $S_{SSID\,i}$y $\overline{I_{SSID}}_i$ la varianza y la intensidad de la red i-ésima en el conjunto marcado a la entrada respectivamente e $I\Delta_{SSID_i}$ la intensidad detectada para dicha red en la búsqueda actual.

El índice de coincidencia (*IC)* indica una estimación de la coincidencia entre el conjunto de redes marcadas a la entrada y el conjunto de redes detectadas en el momento actual. En base a dicho índice podemos establecer una relación entre su valor y el tiempo necesario entre búsquedas del sistema. Esto se basa en que cuanto menor sea el índice de coincidencia, menor será la coincidencia entre el punto de entrada y el actual, por lo que la distancia entre ambos será mayor. Puesto que la distancia es mayor, el tiempo necesario para recorrerla por parte del usuario es mayor igualmente, así que en la siguiente expresión se establece la relación aproximada entre el valor del *IC* y el tiempo estimado en recorrer dicha distancia por un usuario a una velocidad relativamente alta, con el fin de evitar el hecho de no detectar la salida por un reducido tiempo de búsqueda.

$$tiempoEstimado = \frac{IC}{1.7}$$

De esta forma, si el *IC* entre un conjunto de redes marcadas y un conjunto de redes leídas es de 7.9, el tiempo al cual se realizará la siguiente búsqueda será aproximadamente 5 segundos, evitando de esta forma realizar la búsqueda cada 3 segundos sea cual sea la *distancia estimada*. Si el *IC* aumenta, significará que la distancia es mayor, por lo que el tiempo estimado aumentará y las búsquedas se realizarán de manera más esporádica.

La eficiencia de este método radica en que si el usuario permanece estático en un lugar lejano del punto de entrada, la frecuencia de búsqueda será significativamente menor, por lo que el gasto energético producido por la búsqueda continua disminuirá significativamente.

El algoritmo anterior da muy buenos resultados para la detección de salidas, pero tiene un grave inconveniente, si la salida se realiza por un lugar distinto al de entrada, las redes inalámbricas detectadas no coincidirán en el momento de la salida con las que se almacenaron a la entrada, por lo que no se detectaría dicha salida aun habiéndose producido. Esto no es un problema cuando el individuo se encuentra en su casa (dado que sólo suele haber una puerta de salida que es la misma que la de entrada), pero cuando se encuentra en un lugar público como, por ejemplo, un centro comercial o en un edificio institucional, donde suelen presentarse varias puertas de entrada y salida, sí que pasaría a ser un problema. Se intentará resolver esta deficiencia más adelante, cuando hablemos de técnicas combinadas de detección de salida a exteriores.

En este caso, cambiaremos la detección de salida por redes WiFi por una nueva técnica cuyo consumo es mucho menor: la acelerometría. Entendemos por técnicas acelerometría las técnicas cuyas funciones se basan en el uso de un acelerómetro que, como se describió en el capítulo 4 no es más que un instrumento destinado a medir las aceleraciones que se producen sobre un cuerpo, en este caso sobre el dispositivo móvil. Actualmente, la mayoría de los dispositivos móviles de gama media-alta que se comercializan ofrecen dicha funcionalidad, por lo que la aprovecharemos para el objetivo que nos ocupa.

Para ello nos basaremos en detectar patrones de movimiento en el usuario, de manera que gracias a esa identificación podamos suponer estados en los que la probabilidad de producirse una salida al exterior sea elevada. Pongamos un ejemplo sencillo para ilustrar esta situación y posteriormente, nos centraremos en la descripción formal de la técnica.

Supongamos que el usuario entra en la oficina en la que trabaja, donde no existe cobertura satelital y por tanto la señal GPS del dispositivo que porta se interrumpe. Una vez el usuario entre en el recinto, probablemente acuda a su puesto de trabajo, deposite el dispositivo sobre la mesa y de comienzo su actividad laboral, la cual desempeña durante 5 horas. Si no aplicamos ninguna técnica de detección de salidas, durante esas cinco horas, el dispositivo GPS estaría encendido e intentando buscar señal GPS sin resultado, con el consiguiente gasto de batería. Sin embargo, si usamos el acelerómetro que se incluye en el dispositivo para detectar que el usuario está sentado o que ha dejado el dispositivo sobre la mesa, apagaríamos el receptor GPS durante dichas horas, dado que no se ha detectado ningún movimiento que pueda dar lugar a una salida a exteriores. Tras la jornada laboral, el usuario cogería de nuevo el dispositivo y éste detectaría que ha comenzado a andar, por lo que se volvería a reactivar la señal GPS.

Por tanto, llegados a este punto, lo que pretendemos obtener es un sistema preciso y eficaz que sea capaz de detectar los posibles estados de actividad del usuario y actuar en consecuencia sobre el receptor GPS activándolo o desactivándolo según sea necesario.

De esta forma, tras capturar los datos de acelerometría proporcionados por el sensor, debemos tratarlos con el fin de obtener una serie de parámetros que puedan describir cada una de las actividades que pueden reconocerse en el usuario. Este proceso de tratamiento de datos fue ampliamente descrito en el Capítulo 4, por lo que no nos detendremos a explicarlo. Sin embargo, debemos recordar que dicho proceso es complejo y no trivial, por lo que nos debemos hacer una pregunta ¿es realmente necesario determinar las actividades que el usuario está realizando?, o ¿basta con identificar un patrón que, en general, indique movimiento?

Aunque existen varios métodos de reconocimiento de posturas y actividad del usuario basándose en acelerometría, hemos decidido emplear una red Bayesiana debido al poco coste computacional que precisa. Sobre todo, esto último es muy importante en nuestro caso, ya que el objetivo principal del sistema no es el reconocimiento de actividades, sino la detección de salidas a exteriores. De esta manera, es preferible sacrificar precisión con el fin de facilitar al usuario la labor de la detección, ya que mediante este método no sería necesario ni siquiera un periodo de aprendizaje. La única variable del sistema será el módulo de acelerometría, a partir del cual se identificará un valor límite o umbral que indica el valor de acelerometría límite entre estado de reposo del dispositivo y estado de excitación o movimiento.

Todo lo anterior indica que necesitaremos 3 niveles a la hora de obtener la actividad que el usuario está realizando: aprendizaje del sistema, obtención de datos y clasificación de la actividad. A continuación describiremos brevemente cada uno de estos niveles, aunque debemos recordar que la detección de la actividad del usuario no es el objetivo de este capítulo.

- *Aprendizaje del sistema*: esta fase es primordial a la hora de llevar a cabo la clasificación de la actividad. Durante este tiempo, se pide al usuario que realice determinadas acciones cuyo objetivo es recolectar información estadística y de tendencias para poder llevar a cabo el reconocimiento en las siguientes fases de la detección. Sin embargo, esta fase se puede internalizar de manera que el sistema se base en una serie de datos preestablecidos, a partir de los cuales llevar a cabo la tipificación de la actividad.

- *Obtención de datos*: tras el aprendizaje llevado a cabo por el usuario para habilitar la red bayesiana para el reconocimiento de actividades, el sistema comienza a leer la acelerometría registrada por el dispositivo. Durante todo este tiempo, el sistema creará ventanas temporales de datos acelerométricos que serán las que se estudien para determinar el tipo de actividad que está realizando el usuario. Tanto el tamaño de la ventana temporal como el número de datos que se incluyan en ella serán constantes a lo largo de toda la ejecución.

- *Clasificación de la actividad*: tras obtener la ventana temporal de datos procedentes del acelerómetro en la fase anterior, en esta fase se identificarán una serie de medidas estadísticas que permitirán al sistema reconocer el patrón de actividad o inactividad del dispositivo. Como se ha comentado, en este caso no necesitamos conocer exactamente la actividad concreta llevada a cabo, sino simplemente si se trata de un patrón de movilidad o si, de lo contrario, se trata de un patrón de detención.

Para concluir con esta sección, explicaremos cómo se llevará a cabo la detección de salidas a exteriores mediante este método. En primer lugar debemos recordar que el acelerómetro del dispositivo está siempre activo, por lo que no es necesario desconectarlo. En cuanto a la detección de entrada, seguiremos el mismo proceso empleados en la técnica mediante redes WiFi's, es decir, consideraremos que hemos entrado en un lugar cerrado cuando el dispositivo GPS pierda la señal durante un tiempo prolongado que, en nuestro caso, hemos ajustado a 90 segundos. Pero en este caso, dado que no podemos marcar la posición actual (como hicimos en el método anterior) debemos esperar a tener un evento que nos permita saber que la señal GPS del dispositivo no volverá. Este evento es la parada del usuario en algún lugar durante un tiempo superior a 30 segundos. Si se produce esto, indica que el usuario no se está movimiento y, probablemente, vaya a estar detenido durante un tiempo prolongado, por lo que la señal GPS no regresará al dispositivo hasta que el usuario se vuelva a poner en movimiento. Estas transiciones entre movimiento y parada serán detectadas mediante la acelerometría.

Como hemos comentado anteriormente y queda ratificado a través del análisis de la funcionalidad, tan sólo es necesario conocer la tipificación de movimiento/parada de la actividad realizada, en ningún momento es preciso determinar la actividad llevada a cabo. Esto hace que el reconocimiento sea mucho más sencillo, rápido y, sobre todo, computacional y energéticamente mucho más eficiente.

En la Figura 120 se puede observar de manera resumida los estados por los que pasará el sistema durante la detección de salidas y de entradas en lugares sin cobertura GPS.

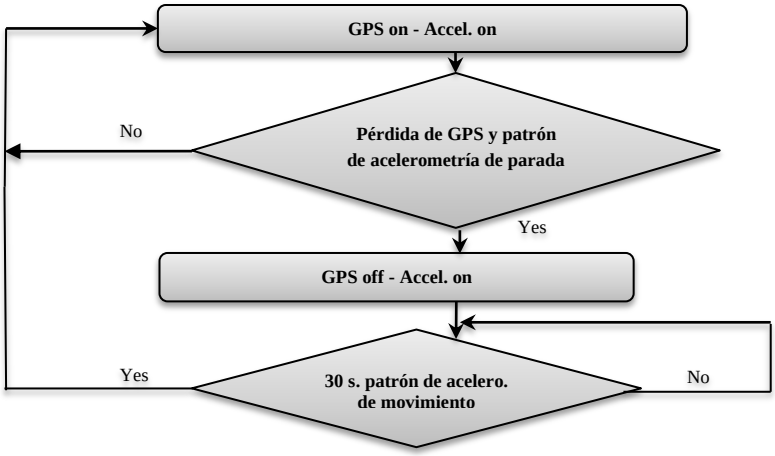


Figura 120: Diagrama de control del método de detección de salidas mediante sensores de acelerometría

Con esto queda finalizada la detección de salidas a exteriores empleando la técnica de acelerometría. Aunque es una técnica que no proporciona excesiva información cuando el usuario no se encuentra detenido, veremos que es muy eficaz en la realidad ya que, en la mayoría de los casos, pasamos la mayor parte del tiempo en nuestro trabajo o en nuestra casa y, en estos lugares donde obviamente no hay cobertura GPS, tendemos a dejar el móvil sobre el escritorio o sobre la mesa de nuestra casa. Por tanto, este método es muy eficaz en estos casos y hace que la duración de la batería se dispare ya que durante este tiempo el dispositivo GPS se encontrará desactivado.

Sin embargo, el porcentaje de falsos positivos en este método es muy elevado, ya que cualquier movimiento implicará la reactivación del dispositivo GPS, con el consecuente gasto energético innecesario producido por su activación en un lugar cerrado. Concretamente, según los estudios llevados a cabo, el porcentaje de acierto del sistema basado en acelerometría es del 100%, ya que no se produce ningún falso negativo. Recordemos que el porcentaje de acierto viene determinado por la relación entre el número de salidas no detectadas y el total de salidas, por lo que no influye en ningún caso los falsos positivos que se puedan dar. Es en este indicador donde el sistema basado en acelerometría presenta un elevado número de errores. En concreto el porcentaje de salidas detectadas respecto al total de salidas es del 87%, es decir, 87 de cada 100 salidas son detectadas en vano, tan sólo 13 salidas de cada 100 detectadas son reales. Esto, como se comentó anteriormente, influye de manera directa en el consumo, disminuyendo la eficiencia energética del dispositivo debido a las continuas reactivaciones del sistema GPS.

En general, cuando el sistema se emplea con usuarios cuyos hábitos son un trabajo sedentario o que pasan un elevado tiempo sin el dispositivo, el rendimiento energético es elevado, ya que el número de falsos positivos se reduce. En cambio, cuando un determinado usuario realiza algún tipo de actividad en interiores que implica movimiento, por ejemplo un profesor durante una clase explicando al alumnado o un pintor de automóviles en el interior de una nave industrial, el sensor GPS estará continuamente encendido, puesto que se registran constantemente patrones de acelerometría de movimiento.

En este caso introduciremos otro método de detección de salidas a exteriores basado en la red GSM de telefonía. El sistema global para las comunicaciones móviles (GSM) es un estándar definido para la comunicación entre dispositivos móviles que empleen tecnología digital para la transmisión de datos. No profundizaremos en la definición del sistema GSM ni su implementación física, aunque sí deberemos conocer algunas pinceladas sobre el sistema, de forma que sea posible comprender en qué consiste la solución basada en la tecnología GSM.

El sistema GSM debe dar soporte a una gran cantidad de usuarios accediendo a él al mismo tiempo, por lo que no es factible el uso de una sola antena para dar cobertura a todos ellos. En su lugar, la red GSM está basada en celdas, es decir, se colocan varias antenas repartidas por una zona concreta de manera que los usuario puedan acceder a la red telefónica a través de una de ellas y, en caso de saturación de la antena, sería posible acceder a través de otra. A nivel físico, se reserva a cada antena un rango de frecuencias distinto a la de las antenas cercanas, para evitar colisión.

Es precisamente esta división en celdas de la red la que nos permitirá llevar a cabo nuestro sistema de detección. Cuando hacemos un viaje, es obvio que no siempre estaremos en la cobertura de una sola antena GSM (a no ser que se trate de un viaje muy corto). En su lugar, al iniciar el viaje estaremos bajo la cobertura de una antena determinada y, tras algunas decenas de metros, estaremos en la cobertura de una nueva. Gracias a las características de los dispositivos móviles actuales, podemos observar el identificador de la antena que nos brinda la cobertura, de modo que cuando cambiemos de identificador significará que hemos cambiado nuestra posición.

Por desgracia, la precisión de este método depende del lugar dónde nos encontremos. En caso de localizarnos en una zona urbana como, por ejemplo, una gran ciudad o una población de gran tamaño, la densidad de antenas GSM es tan elevada, que podemos encontrar una cada 100 metros aproximadamente. Sin embargo, en caso de zonas poco pobladas el panorama cambia, llegando a ser la distancia entre antenas de unos 5 kilómetros. Sabiendo esto, podemos adelantar que este sistema de detección de salidas a exteriores no es válido en principio de manera aislada, sino que su potencial lo encontraremos al combinarlo con otras técnicas de localización.

En la actualidad existen algunos sistemas de posicionamiento basados en la señal GSM que brindan una mayor precisión en zonas rurales, llegando incluso a tener un margen de error de 300 metros en lugares donde las antenas están separadas varios kilómetros. Sin embargo, el principal problema de todos estos métodos es que necesitan de una modificación en la infraestructura de redes para soportarlos, la cual conllevaría un coste excesivo para la compañía de telefonía.

Por un lado se encuentra el sistema denominado “Time of arrival” (TOA). Este método se basa en la diferencia temporal existente entre la trama que se envía desde la estación base y el instante de llegada al dispositivo. Mediante el cálculo de dicha diferencia temporal, es posible aproximar de una manera más eficaz la distancia entre el dispositivo y la estación base. Esta técnica es especialmente útil en ambientes rurales, ya que la diferencia temporal es apreciable a partir de 500 metros, por lo que si las antenas de telefonía se separan por una distancia menor, el sistema no es aplicable. La distancia es obtenida sabiendo la velocidad de la señal y el tiempo de llegada de la señal, gracias a estos dos datos es posible aplicar la siguiente relación:

$$d = v \cdot \Delta,$$

donde d es la distancia que separa la estación base del dispositivo, v la velocidad de transmisión de la señal telefónica y Δt la diferencia temporal entre la hora de emisión de la señal y la de recepción en el dispositivo móvil. Como hemos dicho, este método no supone un incremento de la precisión mucho mayor que el basado en identificación de celdas (Cell-ID), aunque en zonas rurales puede llegar a ser bastante útil.

Por otro lado aparece el método denominado "Uplink Time Diference of Arrival" (U-TDOA). En este caso la localización, al igual que en el método TOA, se basa en la diferencia de tiempo de llegada de la señal, solo que cambia drásticamente la visión. En lugar de procesarse la diferencia temporal en el móvil, la operación se realiza en cada una de las estaciones base a las que llega la señal del dispositivo. Es decir, cada estación base registrará la diferencia temporal entre ella y el dispositivo del que ha recibido la señal y, gracias una triangulación múltiple en la que participan todas las estaciones bases que ha recibido la señal del dispositivo, es posible posicionarlo en un lugar concreto del espacio. Este método posee varias ventajas respecto al anterior. En primer lugar el proceso se realiza en las estaciones bases, por lo que no es necesario adaptar cada uno de los dispositivos que forman parte de la red móvil del lugar, sino que basta con adaptar las estaciones de comunicación, por lo que el impacto es mucho menor. Por otro lado se aumenta sobremanera la precisión de la localización, llegando a ser el margen de error de unos 50 metros en la mayoría de los casos. Este método es usando habitualmente para la localización de llamadas al servicio de emergencia y es un estándar de obligado cumplimiento para las llamadas al 112 en EEUU.

Sin embargo, para el sistema que vamos a realizar tenemos un grave problema para cada uno de los métodos. En el primer caso (TOA), la precisión, como vimos, es muy baja, por lo que no tiene sentido predecir salidas a exteriores partiendo de un margen de error de unos 500 metros. En el caso de la segunda técnica de localización (T-DOA), el margen de error es elevado, aunque admisible, sin embargo la localización se realiza por parte de las estaciones bases y de la compañía de telecomunicaciones, por lo que no es posible acceder a los datos de posicionamiento. Por este motivo, el método de salidas a exteriores basado en señal GSM se empleará como método de apoyo y usaremos la técnica de identificación de celdas (Cell-ID) como técnica de detección que, aunque es el menos preciso, no precisa de cálculos adicionales y las ventajas que posee el método TOA respecto al Cell-ID no son necesarias en nuestro caso.

Veamos detalladamente en qué consistirá la detección de salidas basada en GSM en nuestro sistema. Cuando perdamos la cobertura GPS, al igual que en los demás métodos, significará que hemos entrado en un lugar que ha impedido la visión directa de satélites. En ese instante, almacenaremos el identificador de la antena GSM que nos está proporcionando la señal así como la cobertura que tenemos en ese instante. Mientras nos movemos por el interior del edificio, el identificador de la celda permanecerá constante y sólo cambiará cuando nos hayamos desplazado una larga distancia (en torno a 100 metros en zonas urbanas) del punto en el que se perdió la señal. Cuando detectemos este cambio de antena, iniciaremos de nuevo el dispositivo GPS ya que se ha detectado un movimiento del usuario, por lo que posiblemente haya salido del edificio y la cobertura GPS sea de nuevo adecuada.

Usando este sistema nos encontramos con dos problemas. En primer lugar la cuestión de la precisión de la que antes hablábamos. Sólo detectaremos un cambio de posición cuando el usuario haya avanzado lo suficiente para cambiar de antena de cobertura y esto, en zonas urbanas es admisible pero en zonas rurales, donde las distancias entre antenas son muy elevadas, conllevaría una falta de eficiencia terrible. Por otro lado tenemos el problema de los cambios de cobertura en interiores. Estos se producen cuando en el interior de un mismo lugar, brindan cobertura dos antenas distintas, de modo

que una de ellas provea a parte del edificio y la otra al resto. En este caso, cuando el usuario avanza en el interior del lugar donde se encuentra, se producirían cambios de antenas, por lo que el sistema lo interpretaría como un movimiento y una posible salida a exteriores. De esta manera, activaría el receptor GPS y esperaría inútilmente una señal GPS válida pero esto, dado que el usuario aún se encuentra en el interior del edificio, nunca llegará. El coste en batería del arranque del GPS sería innecesario y, por tanto, nuestro sistema no estaría desempeñando la función para la cual se está realizando: detectar salidas a exteriores con el fin de ahorrar consumo en el dispositivo GPS.

En cuanto al análisis de la relación de falsos positivos y falsos negativos, la realidad es que los resultados llevan a la confusión. Si tan sólo dijéramos que el porcentaje de acierto es del 100%, el de falsos positivos es del 10% y el de falsos negativos es del 0%, pensaríamos que nos encontramos ante el método definitivo de detección de salidas, pero nada más lejos de la realidad. Es cierto que el porcentaje de aciertos es del 100% dado que no se produce ningún falso negativo, aunque si analizamos la distancia transcurrida entre el momento en el que se detecta la salida y el momento en el que se realiza en realidad, nos daríamos cuenta que estaríamos hablando de cientos de metros incluso kilómetros. Por tanto, aunque el porcentaje de falsos negativos es 0%, la precisión es realmente baja. Respecto al número de falsos positivos, estos se darán en caso de que existan varias redes que brinden cobertura al mismo lugar, en tal caso se producirán cambios de antena GSM que implicarían una falsa detección.

Por los motivos descritos anteriormente, se comentaba anteriormente que este método no puede ser empleado de forma aislada, dado que los beneficios que aportaría serían cuestionables, por lo que se usará como apoyo a otros métodos como, por ejemplo, el basado en redes Wifi o en acelerometría. Este sistema lo expondremos a continuación.

SISTEMA BASADO EN TÉCNICAS COMBINADAS

Como ya ha sido mencionado anteriormente, el uso de técnicas basadas en acelerometría, WiFi o GSM plantean una serie de problemas cuando se utilizan estas técnicas de forma individual, los cuales podrían ser solucionados mediante el uso de una combinación de dichas técnicas. Para mejorar la eficiencia del sistema es necesario definir una estructura que aúne todos los métodos descritos anteriormente y de esa forma crear un método único que combine los beneficios de cada una de las técnicas individuales. Este método se presenta como "Sistema basado en técnicas combinadas". La estructura del sistema expuesto emplea los siguientes módulos:

- *Acelerometría*: se utiliza para detectar el movimiento del usuario. Esto permite que el sistema desconecte otros sensores empleados durante la detección de salidas cuando el dispositivo esté inmóvil.
- *WiFi*: se utiliza para marcar el punto de entrada cuando el usuario entra en el interior. También se utiliza para detectar las salidas cuando el usuario pasa por el punto de entrada marcado. Debido a que esta técnica consume más batería que el acelerómetro, el sensor de WiFi debe permanecer encendido durante el menor tiempo posible para evitar un impacto elevado sobre la eficiencia energética.
- *GSM*: dado el porcentaje nulo de falsos negativos de esta técnica, se empleará cuando ninguna de las anteriores de resultado, haciendo así que el sistema detecte la salida aunque con una precisión espacial reducida, la cual vendrá en función de las características de la red GSM en la que se encuentre el usuario.

La Figura 121 muestra claramente los estados a través de los cuales pasará el sistema para la detección de las salidas y entradas en lugares sin cobertura de GPS mediante un método basado en la combinación de las técnicas anteriores. Los algoritmos utilizados en esta técnica combinada son los mismos que los utilizados para el desarrollo de las técnicas aisladas. La única diferencia entre este método de detección de la salida y los anteriores es que las diversas técnicas se han vinculado con el fin de aumentar la eficiencia energética y la precisión de la detección. Obviamente, como se ve en la Figura 121, el esquema para este caso es más complejo, debido a la transición entre los tres métodos.

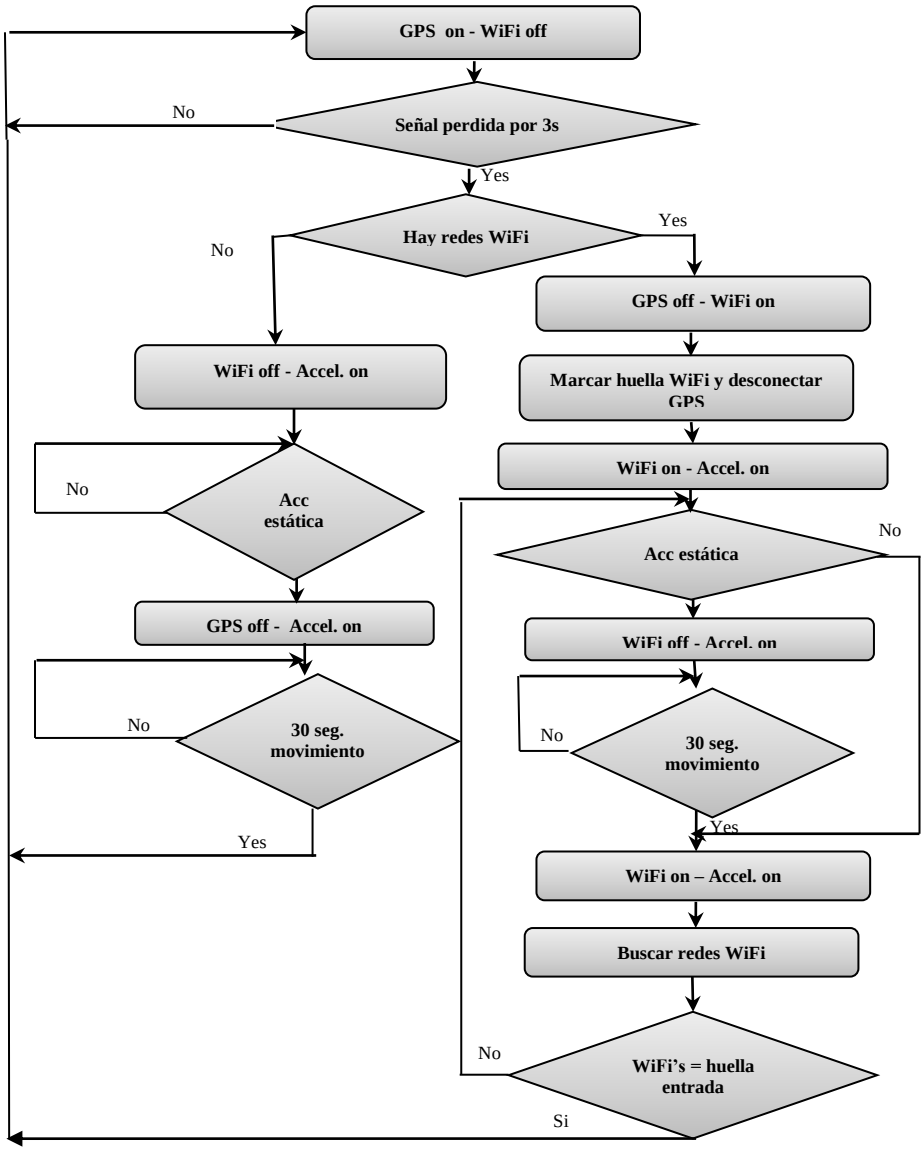


Figura 121: Diagrama de control del método de detección de salidas mediante el uso de técnicas combinadas

En primer lugar, se debe detectar la entrada de un usuario a un lugar interior, lo cual se determina cuando se reconoce un pérdida de la señal del GPS. Al perder la señal, existen dos alternativas en función de si hay cobertura WiFi o si, por el contrario, no existe ninguna red al alcance o las existentes no son suficientes para llevar a cabo el reconocimiento de salidas. Si existen redes WiFi en el momento de la pérdida de la señal GPS, entonces el punto de entrada será marcado y el receptor GPS se desconecta a continuación. El marcado del punto se hará de acuerdo a lo expuesto en el apartado

anterior. Por el contrario, si no existen redes en el momento de la pérdida de la señal GPS, es imposible marcar este punto mediante esta técnica y por lo que el sistema de detección basado en acelerometría será activado. En este caso el dispositivo GPS no será desactivado hasta que no se produzca un patrón de movimiento relacionado con inactividad.

Si se marcó la entrada usando la técnica WiFi, cuando se detecta que el usuario ha dejado de moverse por un período de tiempo específico, el sensor de WiFi será desactivado hasta que se vuelva a producir un movimiento. Por el contrario, si no se pudo marcar el punto de entrada mediante redes WiFi, será el GPS el que se desconecte cuando se detecte inactividad continuada.

Seguidamente, el sistema esperará cualquier patrón de movimiento del usuario mediante los sensores de acelerometría. Al detectar que el usuario está en movimiento, el sensor de WiFi se reactivará siempre y cuando el punto de entrada estuviera marcado con anterioridad, para proceder con la detección del punto de salida. De lo contrario, se activará directamente el sensor GPS y permanecerá a la espera de una señal válida.

RESULTADOS

Una vez explicadas en profundidad las técnicas que emplearemos para la detección de salidas a exteriores, realizaremos una comparativa entre cada una de ellas para obtener los resultados y la fiabilidad, así como la energía ahorrada en cada método. Gracias a este análisis podremos comparar todos los métodos y así elegir aquel que suponga un menor coste energético para el dispositivo y que posea una mayor precisión.

Realizaremos el análisis para cada uno de los métodos expuestos anteriormente. Para ello se ha escogido a un grupo de 50 usuarios con dispositivos compatibles con la aplicación desarrollada. Concretamente los dispositivos poseían un sistema operativo Android en diferentes versiones, sobre los cuales se instaló la aplicación desarrollada para comprobar la eficacia del sistema, la cual se muestra en la Figura 122. Una vez instalada la utilidad en los dispositivos de dichos usuarios se les indicó que realizasen la actividad diaria de manera normal, sin ningún condicionamiento. Además, se pidió que anotaran en un cuestionario proporcionado la hora de entrada y salida de los lugares techados a los que accedieran a lo largo del tiempo durante el que se llevó a cabo el experimento, en concreto 7 días. Mediante la comparación del log de la aplicación en el que se recogía la hora de detección de entrada y salida en lugares sin cobertura GPS, así como la posición GPS donde se produjo la entrada y la salida, y el formulario relleno por cada uno de los usuarios, se pudo determinar el porcentaje de aciertos, falsos positivos y falsos negativos de la aplicación para la detección de entradas y salidas. Paralelamente al proceso de detección de salidas, se realizó una aplicación que nos permitía monitorizar los cambios en el nivel de batería del dispositivo, por lo que ha sido posible visualizar el tiempo de uso resultante para los distintos métodos de detección de salidas en comparación con el sistema de monitorización continua sin la desconexión del GPS.

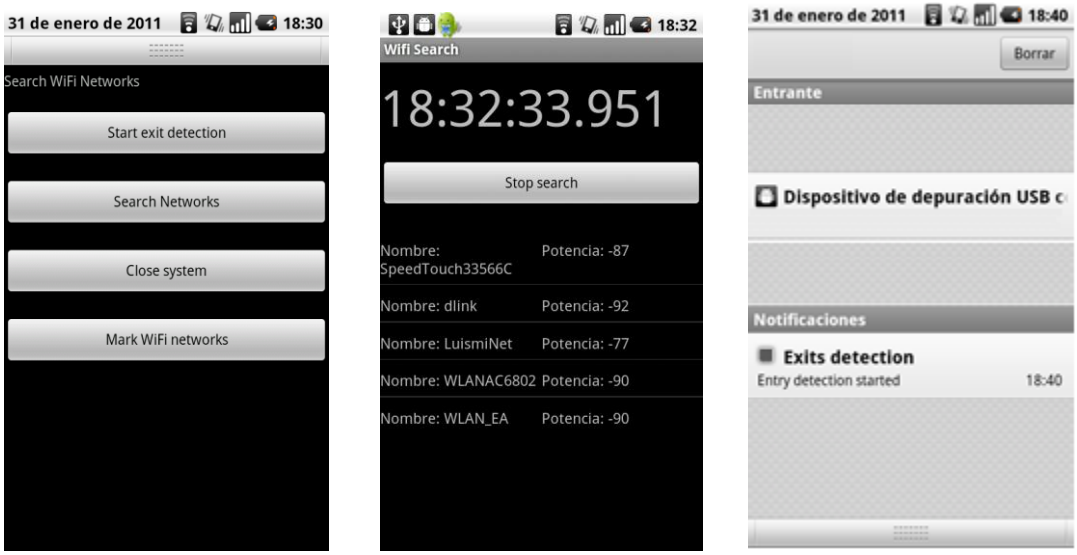


Figura 122: (a) Menú principal de la aplicación desarrollada para poner a prueba el método propuesto, (b) Proceso manual de búsqueda de WiFi para obtener los puntos WiFi cercanos, (c) Proceso de detección de salidas a exteriores

En cuanto a la categorización de errores, diremos que una lectura ha sido errónea cuando, a pesar de haber salido el usuario al exterior, esto fuera detectado 5 minutos más allá del momento en el que se produjera esta salida. Es decir, si un usuario realiza una salida al exterior a las 14:00, si el sistema detecta que ha salido al cabo de un minuto (14:01) la detección se clasificaría como positiva. En cambio, si la salida al exterior se detecta a las 14:08 (8 minutos después de haberse producido realmente) la detección se consideraría errónea.

Una vez explicadas las pruebas que se realizarán sobre el sistema, vayamos con los resultados de éstas y con la posterior elección del sistema más preciso. En la Tabla 13 se recogen los datos resultantes del proceso de recopilación de muestras a partir del dispositivo móvil y el formulario relleno por los usuarios. En dicha tabla se recogen las técnicas empleadas junto con sus correspondientes ratios de errores y aciertos. Por último, también podemos observar el valor en minutos de la duración de la batería del dispositivo, el cual se encuentra ejecutando de manera continuada el sistema de detección de salidas implementado.

Con los resultados anteriores podemos realizar varias comparativas distintas eligiendo así la técnica que mejor resultado ha obtenido desde ese punto de vista. Por un lado se encuentra, como es obvio, la fiabilidad de la técnica, que se calculará en base al porcentaje de errores. Por otro lado podemos comparar las técnicas en función al tiempo medio de detección, de manera que aquella que menor tiempo medio necesite será la que, en términos de media, mejor respuesta tenga a la detección de la salida. Por último, no podíamos olvidar comparar la duración de las baterías para cada una de las técnicas, dado que es el objetivo de nuestro estudio, reducir el consumo de baterías con el fin de poder realizar un seguimiento constante y eficaz sin necesidad de intervención por parte del usuario en el proceso y con la mayor autonomía posible.

En la Tabla 13, el primer elemento que podemos estudiar es el porcentaje de aciertos. Este porcentaje indica la relación entre el número de salidas reales y detectadas por el sistema, mientras que no tiene en cuenta en ningún momento el hecho de que se detectara una salida que en realidad no se produjo (falso positivo). Por ello, podemos observar como para el caso de la técnica basada en acelerometría, el porcentaje de acierto es muy elevado, ya que identifica siempre la salida, aunque la

mayoría de las veces que lo hace en realidad no se produjera. En el otro extremo, aparece la técnica basada en GSM. En este caso el índice de aciertos es tan reducido dado que imponemos un tiempo para que la salida sea detectada. Puesto que como se comentó en el apartado correspondiente a GSM, las antenas de telefonía pueden abarcar varios kilómetros, esta salida podría tardar demasiado tiempo en ser detectada, de ahí el bajo rendimiento en este sentido.

El resto de parámetros fueron estudiados independientemente en las respectivas técnicas, sin embargo, merece especial atención el porcentaje de falsos negativos del método de técnicas combinadas. En este caso, el valor es 0, ya que los falsos negativos típicos de la técnica WiFi quedan cubiertos por la de GSM, donde tarde o temprano se detectará la salida, de ahí que el este valor sea 0%. El número de falsos positivos permanece igual que en el caso del método WiFi individual.

Por último, si analizamos el tiempo de vida de la batería, podemos observar que la técnica más eficiente es la basada en GSM, aunque por desgracia su porcentaje de acierto es muy bajo. Seguidamente, en el extremo opuesto aparecen las técnicas basadas en WiFi y acelerometría. En el primer caso, el bajo tiempo de vida de la batería viene causado por el uso continuado del adaptador WiFi, el cual presenta un consumo considerable. Por otro lado, en el caso de la acelerometría, aunque bien es cierto que el acelerómetro a penas consume batería, el hecho de activar el GPS cada vez que se lleva a cabo un movimiento, influye negativamente en el tiempo de vida. En último lugar encontramos el método que mejor combina la eficiencia energética con la precisión en el reconocimiento, el método de técnicas combinadas. En este, el porcentaje de aciertos es muy elevado, rozando el 100% y, por otro lado, el consumo energético es reducido, llegando al punto de que sería posible realizar un geoposicionamiento continuado del usuario durante más de 24 horas.

Técnica	***WiFi***	***Acelerometría***	***GSM***	***WiFi + GSM + acelerometría***
% acierto	97%	100%	14%	98%
% falsos positivos	2%	87%	10%	2%
% falsos negativos	3%	0%	86%	0%
% total errores	5%	87%	96%	2%
Tiempo batería (min)	893	758	1939	1496

Tabla 13: Tabla de porcentajes de aciertos y errores de los diferentes métodos implementados de detección de salidas a exteriores

AHORRO ENERGÉTICO

Para comprender realmente la dimensión del ahorro energético para el caso del sistema de detección de salidas, estudiaremos la potencia en vatios ahorrada mediante el método propuesto. Realizaremos este estudio partiendo de las especificaciones de la batería de un dispositivo móvil marca HTC modelo Nexus One con una intensidad de 2400mAh y una corriente de 3,7 voltios. En primer lugar calcularemos los vatios por hora consumidos por el dispositivo con el que estamos trabajando. Para ello emplearemos la siguiente expresión que nos permite obtener la potencia ofrecida por la batería:

$$P = V \cdot I$$

donde P es la potencia del dispositivo, V el voltaje e I la intensidad. En nuestro caso, la ecuación anterior quedaría de la siguiente forma:

$$P = V \cdot I = 3{,}7 \cdot 2{,}4 = 8{,}9Wh$$

Es decir, nuestro dispositivo consumirá un total de aproximadamente 9 vatios por hora. Esta cantidad en sí no nos da demasiada información, por lo que realizaremos una comparativa con ciertos elementos electrónicos para dar una idea real de la potencia ahorrada.

Previamente, debemos determinar la diferencia entre el tiempo de vida de la batería para un uso continuado del GPS y el tiempo de uso aplicando la técnica desarrollada. Como se expuso en el apartado Comparación de la eficiencia de las técnicas, la duración de un ciclo de vida de la batería con un uso continuado del dispositivo GPS era de 500 minutos, es decir, aproximadamente 8 horas. En cambio, aplicando el método desarrollado, este tiempo asciende a 25 horas, por lo que el tiempo de baterías ahorrado mediante el uso de la técnica de detección de salidas es de 17 horas. Este ahorro traducido a vatios por hora hará un total dado por la siguiente expresión:

$$E_{ahorrada} = P \cdot \Delta t = 8{,}9W/h \cdot (25h - 8h) = 8{,}9W/h \cdot 17h/ciclo = 151W/ciclo$$

Esto se obtiene teniendo en cuenta que la duración de la recarga del dispositivo ha sido realizada en una hora. De esta forma, el uso del sistema de detección ahorra $151W$ por cada recarga realizada por el dispositivo. Teniendo en cuenta que el consumo de CO_2 a la atmósfera es de alrededor de 650gr por *KWh*, podemos calcular la cantidad de dióxido de carbono que el sistema de detección ha logrado evitar emitir a la atmósfera. Esto puede observarse en la Tabla 14, en la cual se hace un estudio de los gramos de CO_2 no emitidos.

Periodo de tiempo	Cantidad de CO_2 no emitido a la atmósfera (gr)[1]
Diario	98
Semanal	687
Mensual	2940
Anual	35280

Tabla 14: Tabla de emisiones ahorradas a la atmósfera mediante el uso del sistema de detección de salidas a exteriores

Como podemos ver, anualmente se podría reducir la emisión de dióxido de carbono a la atmósfera en 35 kg.

Además del estudio llevado a cabo anteriormente para el control de emisiones, a continuación se realizará una comparativa del consumo ahorrado y el consumo necesario por una serie de electrodomésticos, los cuales fueron publicados en 2010 por la Comisión Europea de Energía[2]. El objetivo de este estudio es analizar el tiempo que podríamos mantener en funcionamiento una serie de aparatos generalmente presentes en el hogar, con la energía ahorrada por cada una de las recargas mediante el uso del sistema de detección de salidas. El resultado de este estudio puede verse recogido en la Tabla 15, en la cual aparece cada uno de los electrodomésticos sometidos a comparación, la

[1] Suponiendo una recarga del dispositivo diaria
[2] http://ec.europa.eu/clima/sites/campaign/

potencia en *Wh* de cada uno de ellos, el coste de la energía necesaria para mantenerlos encendidos durante una hora y por último, los minutos de uso que podrían ser empleados por el dispositivos con la energía ahorrada al emplear el sistema de detección de salidas durante el geoposicionamiento continuo del usuario.

Electrodoméstico	Potencia (W) por hora	Coste (cent. €) por hora	Tiempo de uso por recarga (min)
Bombilla de 60 W	60	0,6	151
Bombilla de bajo consumo	11	0,11	824
Lámpara halógena	300	3	30
Televisor	80	0,8	113
Cadena de alta fidelidad	55	0,6	165
Ordenador portátil	80	0,8	113
Aspirador	700	7	13
Secador de pelo	800	8	11
Hervidor de agua	300	3	30
Microondas	700	7	13
Lavadora	500	5	18
Secadora	500	5	18
Lavavajillas	700	7	13
Radiador	500	5	18
Aire acondicionado	800	8	11

Tabla 15: Minutos de uso de ciertos electrodomésticos con la potencia ahorrada por el sistema de detección

En este sentido podemos observar que aplicando el sistema de detección de salidas, la energía ahorrada por éste en cada uno de los periodos de recarga del dispositivo en comparación con el uso del sensor GPS de manera continuada, sería suficiente como para mantener encendido un televisor casi 2 horas, o para conservar una bombilla de bajo consumo con una luz semejante a una bombilla normal de 60W durante 13 horas.

Como punto y final al capítulo actual, pondremos de manifiesto las numerosas ventajas del uso del sistema de detección de salidas no sólo en cuanto a eficiencia energética y contaminación se refiere, que hemos visto que los efectos son realmente significativos, sino desde el punto de vista de la calidad del servicio dado por el dispositivo cuando este se encuentra trabajando con un sistema de posicionamiento continuo. En ese caso, el hecho de recargar continuamente la batería, incluso 2 y 3 veces al día, podría significar un claro problema para el usuario, el cual queda solventado gracias al ahorro del sistema desarrollado que consigue aumentar el tiempo de vida útil de la batería a 25 horas por cada periodo de recarga.

CONCLUSIONES

El objetivo de este capítulo ha sido desarrollar un método para evitar el alto consumo de energía producida por los receptores GPS a la hora de llevar a cabo un geoposicionamiento continuo, concretamente en los dispositivos móviles. En la sección de resultados se puede comprobar que el objetivo se ha logrado. La duración de la batería de un dispositivo GPS en uso constante es normalmente de 7 horas aproximadamente. Sin embargo, gracias al uso de una técnica combinada

basada en WiFi, GSM y acelerometría, la duración de la batería ha ascendido a más de 25 horas. Este tiempo generalmente es suficiente para poder recargar el dispositivo sin que se agote totalmente la batería, por lo que podría realizarse sin ningún problema un geoposicionamiento continuo del individuo que lo transporta.

De esta forma, dispositivos y aplicaciones móviles, particularmente aquellas que hacen uso constante del GPS como técnica de geoposicionamiento, podrían reducir sus costes energéticos gracias a esa tecnología. En consecuencia, la vida de la batería no sería el principal factor limitador, por lo que deja el camino libre para que el desarrollador pueda aprovechar al máximo el uso de aplicaciones basadas en la ubicación del usuario en relación con los métodos de interconexión disponibles en dispositivos móviles.

RESUMEN

A lo largo de este capítulo se ha presentado un método innovador capaz de detectar las salidas a exteriores de usuarios. Este método ha sido aplicado para reducir el consumo energético producido por el sistema de posicionamiento GPS en dispositivos móviles. Concretamente, las bases de la reducción de consumo se han sentado en la desconexión del sensor GPS cuando sea detectada una entrada a interiores, donde la señal GPS no es válida. Por otro lado, el sensor se reactivará cuando gracias al sistema desarrollado, se determine que el usuario ha efectuado una salida. Para poner de manifiesto las ventajas en términos de eficiencia energética del sistema, se ha realizado una comparativa con los distintos criterios y métodos desarrollados para la detección de salidas a exteriores. El resultado de estas pruebas ha sido que, gracias al método generado es posible aumentar el tiempo de uso del dispositivo bajo un protocolo de localización permanente del usuario ha aumentado de 7 horas, sin el uso de ninguna técnica de reducción de consumo energético, a más de 25 horas.

CAPÍTULO 6: CONCLUSIONES Y TRABAJO FUTURO

Life is the art of drawing sufficient conclusions from insufficient premises.
- Smart Butler -

En esta tesis se ha abordado una línea de investigación original basada en la computación ubicua, la cual nació en la última década y ha sido en los últimos años cuando ha experimentado una difusión masiva entre los sistemas de computación. Ante este reto fueron analizadas las propuestas existentes y se detectaron una serie de carencias que han sido subsanadas a lo largo de esta tesis. En este capítulo se expondrán las conclusiones a las que se ha llegado tras elaborar la tesis, analizando los objetivos que deberán ser cubiertos durante los próximos años, así como las futuras líneas de trabajo que han surgido tras su elaboración.

CONCLUSIONES

El objetivo final de esta tesis ha sido elaborar un sistema capaz de ayudar a la interconexión entre aplicaciones independientes gracias a la distribución de la información entre las aplicaciones de una forma ordenada y coordinada por un middleware específico para este tipo de sistemas. Para la correcta utilización del middleware, también se ha definido una arquitectura completa para las aplicaciones basadas en contexto, la cual permite gracias a su organización en capas, la generación de aplicaciones con una distribución fijada que facilitará en gran medida la comunicación entre sistemas, el uso de sensores, la garantía de seguridad y el almacenamiento y procesamiento de la información obtenida.

Esto es especialmente importante en los últimos años, ya que el número de aplicaciones basadas en contexto se ha disparado, por lo que se hace extremadamente necesario el desarrollo de aplicaciones de una forma ordenada, que permita la reutilización de funcionalidad entre varias aplicaciones. Precisamente esto también puede ser conseguido a través del uso de la arquitectura propuesta.

Respecto al middleware que se ha desarrollado, facilita 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 se puede conseguir mediante el uso del middleware, ya que el procesamiento para la obtención de información a partir de la información raw 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.

En cuanto al sistema de reconocimiento de actividades, se ha conseguido realizar un sistema integral capaz de reconocer, clasificar y compartir información sobre la actividad física de un grupo de sujetos. Con ello se ha conseguido controlar la actividad de un usuario y monitorizar dicha actividad mediante el uso de una plataforma de difusión de la información. Esta monitorización se ha aplicado a

usuarios con capacidad de movilidad restringida como ha sido el caso de deportistas que están sometidos a un periodo de actividad restringida (por ejemplo reposo) y en ancianos, dado que gracias al sistema podemos controlar en todo momento su actividad y alertar en caso de incumplimiento de alguna indicación. Dicho de otro modo, sabemos qué actividad está haciendo la persona a la que se monitoriza sin necesidad de que el médico o el familiar esté presente, permitiendo posteriormente analizar la información sobre las actividades realizadas.

Además, todo el sistema se ha adaptado a un objetivo: ser instalado en un dispositivo móvil. Este requisito ha llevado a que todos los métodos de detección, filtrado y reconocimiento de las actividades deban tener un reducido coste computacional. Derivado de lo anterior, no sólo reducimos el riesgo de saturación del sistema del dispositivo móvil debido a excesivos cálculos, sino que reducimos el consumo energético al mínimo posible. Esto hace que el usuario deba recargar el dispositivo menos frecuentemente y en consecuencia, el sistema será más funcional y cómodo.

Por último, con el diseño del sistema propuesto hemos brindado la posibilidad de acceder a la información del familiar sometido a la asistencia mediante varias vías. De este modo damos cobertura a multitud de dispositivos distintos (móviles, PC's, PDA's, etc.) con el fin de poder acceder a la información en cualquier lugar y en cualquier momento.

Estrechamente relacionado con el sistema de monitorización, se ha planteado una novedosa técnica capaz de detectar salidas al exterior de un usuario gracias a la información obtenida a partir de los sensores WiFi, GSM y acelerómetro. Esto ha hecho que sea posible monitorizar no sólo la actividad física que el usuario está llevando a cabo, sino en qué lugar la está realizando. Mediante el posicionamiento continuo y eficiente es posible monitorizar en todo momento la posición del usuario. En esta tesis doctoral se ha resuelto de una manera eficaz el problema derivado del elevado coste energético de los métodos de posicionamiento mediante satélites. Además se ha llevado a cabo de una manera automática y sin necesidad de ninguna interacción por parte del usuario.

Por último, se ha demostrado el alto grado de actualidad que las temáticas tratadas a lo largo de esta tesis tienen en el presente, aunque aún es más evidente el interés para el futuro, lo cual justifica el objetivo de desarrollar un middleware y una arquitectura que soporte dicho middleware para aplicaciones basadas en contexto, así como la elaboración de un sistema eficaz de reconocimiento de actividades y de detección de salidas.

Sin embargo, uno de los principales problemas identificados en la temática es la amplitud y la generalidad del tema tratado. Los sistemas ubicuos no son un área concreta de la computación actual, sino que es una tendencia a la cual todos los sistemas de información personal están dirigidos. Por este motivo, las aplicaciones son innumerables y el hecho de proponer un middleware y una arquitectura para las aplicaciones contextuales en general es un hándicap que si bien es extremadamente necesario hoy en día, no ha sido en absoluto fácil de conseguir. Concretamente podemos observar este problema en la elaboración de la clasificación para información contextual. Si se hubiera cometido el error de elaborar una clasificación demasiado específica, multitud de sistemas contextuales hubieran quedado fuera de los metadatos generados, mientras que si se hubiera realizado una clasificación demasiado genérica, la adaptación a sistemas concretos sería demasiado compleja y existirían diversos problemas de integración de información entre aplicaciones.

Este documento presenta los objetivos y los fundamentos de la tesis que será llevada a cabo por el autor durante los próximos años. De este modo, las tareas que deben realizarse durante este tiempo se podrían dividir en varias secciones correspondientes a cada uno de los capítulos expuestos a lo largo de este documento.

- **Arquitectura para aplicaciones contextuales**. En este aspecto se debe completar la identificación de orígenes de información compatibles con la arquitectura, de manera que se consiga obtener una clasificación compatible con los orígenes actuales de información así como proporcionar una generalidad suficiente para soportar nuevos datos que no hayan sido tenidos en cuenta. Este punto de actualización será continuo en la plataforma, ya que el conjunto de contextos se actualiza constantemente con la aparición de nuevas aplicaciones que empleen y obtengan información relativa al entorno del usuario. Por otro lado se debe proporcionar una serie de soluciones óptimas para cada uno de los requisitos necesarios en las capas de la arquitectura. De esta forma se seguirá investigando en la obtención de nuevas estructuras de datos que permitan asegurar y minimizar la cantidad de datos que comparten las aplicaciones. Este conseguirá hacer de la plataforma un sistema más flexible y, sobre todo, más eficiente.

- **Middleware para aplicaciones contextuales**. El objetivo del middleware es proporcionar un medio que permita a las aplicaciones saber los servicios disponibles que ofrecen otras aplicaciones o servicios instalados en el dispositivo y emplear dichos servicios para obtener información adicional sin necesidad de procesar de nuevo la información procedente de los sensores. Es decir, se trata de un método de intercambio de información entre aplicaciones. Por este motivo, será necesario realizar una correcta gestión del descubrimiento de recursos, por lo que se deberá definir una serie de características que permitan identificar a las aplicaciones y los servicios que éstas ofrecen de cara al intercambio de información. Esta gestión hasta el momento, se basa en el simple descubrimiento de recursos, por lo que la evolución planteada es implementar una serie de políticas de seguridad que permita autogestionar un sistema de incompatibilidades entre aplicaciones y proveedores de contextos. Esta gestión automática favorecerá la seguridad del usuario, de manera que será posible determinar unos criterios mínimos de seguridad y compatibilidad.

- **Reconocimiento de actividades**. El objetivo que se plantea en el ámbito del reconocimiento de actividades es la identificación de nuevas actividades que permitan dotar al sistema de un mayor catálogo de reconocimiento. Para ello se deberá estudiar los diferentes métodos de discretización y reconocimiento para dichas actividades, ya que en la mayoría de los casos provocará un solapamiento con los datos de otras actividades reconocibles. Por último, el correcto modelado del problema del reconocimiento de actividades permitirá aumentar la precisión del sistema, por lo cual deberá ser estudiado en profundidad este aspecto, ampliando la funcionalidad del modelo de Markov y las probabilidades entre transiciones.

- **Sistema de detección de salidas**. A lo largo de este trabajo se ha expuesto un sistema capaz de detectar entradas y salidas a interiores, con el que se ha logrado reducir significativamente la energía consumida por el sensor GPS en aplicaciones de posicionamiento continuo. La evolución planteada en el campo de la detección de salidas es, partiendo del sistema desarrollado, adaptarlo para convertirlo en un sistema de posicionamiento en interiores. Gracias al sistema de marcado de intensidades propuesto, el sistema de posicionamiento se basará en el

etiquetado de lugares y asignación de información semántica identificativa, de manera que será posible el posicionamiento a nivel de habitaciones. Este nivel de precisión será suficiente para sistemas de seguimiento o de control de actividad de usuarios.

BIBLIOGRAFÍA

A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. **Dey, Anind, Abowd, Gregory and Salber, Daniel. 2001.** 2, s.l. : Human-Computer Interaction, 2001, Vol. 16.

A Framework for Ubiquitous Content Sharing. **Kröner, Alexander, Schneider, Michael and Mori, Junichiro. 2009.** 4, s.l. : IEEE Pervasive Computing, 2009, Vol. 8. 10.1109/MPRV.2009.65.

A home-based care model for outpatient cardiac rehabilitation based on mobile technologies. **Sarela, Antti Korhonen, et al. 2009.** Brisbane, Australia : s.n., 2009. 3rd International Conference on Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. pp. 1 - 8 .

A Markov model for driver turn prediction. **Krumm, J. 2008.** s.l. : Society of Automotive Engineers (SAE) World Congress, 2008.

A modified Chi2 algorithm for discretization. **Tay, F.E.H. and Shen, Lixiang. 2002.** 2002, IEEE Transactions on Knowledge and Data Engineering, pp. 666 - 670.

A Study on Saving Energy in Artificial Lighting by Making Smart Use of Wireless Sensor Networks and Actuators. **Alejandro Fernández-Montes González, Luis González Abril, Juan Antonio Ortega Ramirez, Francisco Velasco Morente. 2009.** 6, s.l. : IEEE Network, 2009, Vol. 23.

Activity Monitoring System using Dynamic Time Warping for the Elderly and Disabled people. **al., Sasiwan Paiyarom et. 2009.** s.l. : 2nd International Conference on Computer, Control and Communication, 2009. 978-1-60558-792-9.

Activity Recognition from Accelerometer Data. **al., Nishkam Ravi et. 2005.** s.l. : American Association for Artificial Intelligence, 2005. 1-57735-236-x.

Activity Recognition from Accelerometer Data on a Mobile Phone. **Brezmes, Tomas, Gorricho, Juan Luis and Cotrina, Josep. 2009.** s.l. : Lecture Notes in Computer Science, 2009. 978-3-642-02480-1.

Brezmes, Tomas, Gorricho, Juan-Luis and Cotrina, Josep. 2009. 2009, Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, pp. 796-799.

Activity Recognition using Wearable Sensors for Elder Care. **al., Yu-Jin Hong et. 2008.** s.l. : IEEE Future Generation Communication and Networking, 2008. 978-0-7695-3431-2.

Hong, Yu-Jin, et al. 2008. Seoul, South Korea : Second International Conference on Future Generation Communication and Networking. FGCN '08, 2008. 978-0-7695-3431-2.

AdMob. 2010. *AdMob Mobile Metrics Report.* s.l. : AdMob, 2010.

Alan Dix, Janet Finlay, Gregory D. Abows, Russell Beale. 2004. *Human-computer interaction.* Harlow, Inglaterra : Pearson Education Limited, 2004.

Ameva: an Autonomous Discretization Algorithm. **Abril, Luis González, et al. 2009.** 3, s.l. : Expert Systems With Applications, 2009, Vol. 36.

Ameva: An autonomous discretization algorithm. **Gonzalez-Abril, FJ Cuberos, F. Velasco, and JA Ortega. 2009.** 2009, Expert Systems with Applications, pp. 5327–5332.

An Architecture for Location Aware Applications. **Nord, James, Synnes, Kare and Parnes, Peter. 2002.** s.l. : Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), 2002.

An evaluation of mobile phone text input methods. **Lee Butts, Andy Cockburn. 2002.** 2002, Australian Computer Science Communications, pp. 55 - 59.

An extended Chi2 algorithm for discretization of real value attributes. **Su, Chao-Ton and Hsu, Jyh-Hwa. 2005.** 2005, IEEE Transactions on Knowledge and Data Engineering, pp. 437 - 441.

An infrastructure approach to context-aware computing. **Landay, Jason I. Hong and James A. 2001.** 2001, Human-Computer Interaction (HCI) Journal, pp. 16(2-3).

An Ontology for Context-Aware Pervasive Computing Environments. **Chen, Harry, Finin, Tim and Joshi, Anupam. 2004.** s.l. : Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2004.

An Ontology for Mobile Device Sensor-Based Context Awareness. **Korpipää, Panu and Mäntyjärvi, Jani. 2003.** s.l. : Lecture Notes in Computer Science, 2003, Vol. 2680/2003. 10.1007/3-540-44958-2_37.

Korpipää, Panu and Mäntyjärvi, Jani. 2003. s.l. : Proc. Context '03, LNAI, 2003.

Architectures for Context. **Winograd, Terry. 2001.** 2001, Human-Computer-Interaction, pp. 401-419.

Battery-Aware Embedded GPS Receiver Node. **Dejan Raskovic, David Giessel. 2007.** 2007, IEEE Computer Society.

Bayesian indoor positioning systems. **Madigan, David, et al. 2005.** 2005. Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. pp. 1217-1227.

Chen, Harry. 2003. *An Intelligent Broker Architecture for Context-Aware Systems.* 2003.

Classification Technique of Human Motion Context based on Wireless Sensor Network. **Hong, N. J. K. Joo Hyun, Cha, Eun Jong and Lee, Tae Soo. 2005.** s.l. : Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2005.

Comparative study on classifying human activities with miniature inertial and magnetic sensors. **Altun, Kerem, Barshan, Billur and Tunçel, Orkun. 2010.** 2010, Pattern Recognition, p. 43.

Context Awareness by Case-Based Reasoning in a Music Recommendation System. **Lee, Jae Sik and Lee, Jin Chun. 2008.** s.l. : Ubiquitous Computing System, 2008.

Context-aware computing applications. **Bill Schilit, Norman Adams, and Roy Want. 1994.** Santa Cruz, CA, US : s.n., 1994. IEEE Workshop on Mobile Computing Systems and Applications.

Context-Aware Computing Applications. **Schilit, Bill N., Adams, Norman and Want, Roy. 2003.** s.l. : Workshop on Mobile Computing Systems and Applications, 2003.

Context-awareness on mobile devices - the hydrogen approach. **Hofer, Thomas, et al. 2003.** s.l. : 36th Annual Hawaii International Conference on System Sciences, 2003.

Context-Awareness on Mobile Devices - the Hydrogen Approach. **Hofer, Thomas, et al. 2003.** Hawaii : Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), 2003. 0-7695-1874-5/03.

Crouter SE, Churilla JR, Bassett DR. 2006. *Estimating energy expenditure using accelerometers.* s.l. : Eur J Appl Physiol, 2006. 98:601-12.

CybreMinder: A Context- Aware System for Supporting Reminders. **Dey, A. K. and Abowd, G. D. 2000.** Bristol, UK : Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K), 2000.

Database correlation method for GSM location. **Laitinen, H., Lähteenmäki, J. and Nordström, T. 2001.** Rodas : IEEE VTC 2001 Spring Conference, 2001.

Device Positioning Using Radio Beacons in the Wild. **Anthony LaMarca, Yatin Chawathe. 2005.** s.l. : Springer, 2005. In Proceedings of the Third International Conference on Pervasive Computing. pp. 116-133.

Anthony LaMarca, Yatin Chawathe et al. 2005. 2005. proceedings of Pervasive Computing.

Extracting High-Level Information from Location Data: The W4 Diary Example in a Music Recommendation System. **Castelli, Gabriella, et al. 2009.** 1, s.l. : Mobile Networks and Applications, 2009, Vol. 14.

G, Plasqui, et al. *Measuring free-living energy expenditure and physical activity with triaxial accelerometry.* s.l. : Obes Res 2005. 13:1363-9.

Gaia: A Middleware Infrastructure to Enable Active Spaces. **Román, Manuel, et al. 2002.** s.l. : IEEE Pervasive Computing, 2002.

Game-Based Learning with Ubiquitous Technologies. **Chang, Wen-Chih, et al. 2009.** 2009, IEEE Internet Computing, pp. 26 - 33.

Chang, Wen-Chih, et al. 2009. 4, s.l. : IEEE Internet Computing, 2009, IEEE Internet Computing, Vol. 13, pp. 26 - 33. 10.1109/MIC.2009.81.

GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. **Espinoza, F., et al. 2001.** Berlin : Ubicomp 2001, International Conference on Ubiquitous Computing, 2001.

GSM standards activity on location. **Lopes, L., Villier, E. and Ludden, B. 1999.** Londres : IEEE colloquim on novel methods of location and tracking of cellular mobiles and their system applications, 1999.

Handheld and Ubiquitous Computing. **Thomas, P. and Gellersen, H.-W. 2000.** Bristol, Reino Unido : Lecture notes in computer science, 2000. Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing.

How to build smart appliances? **Schmidt, A. and van Laerhoven, K. 2001.** 2001, IEEE Personal Communications, pp. 66 - 71.

Ilia Mitov, Krassimira Ivanova, Krassimir Markov, Vitalii Velychko, Peter Stanchev, Koen Vanhoof. 2009. COMPARISON OF DISCRETIZATION METHODS FOR PREPROCESSING DATA FOR PYRAMIDAL GROWING NETWORK CLASSIFICATION METHOD. [book auth.] Ilia Mitov. *Information Science and Computing.* 2009, pp. 31-40.

Improving Human Computer Interaction through Spoken Natural Language. **Omar Florez-Choque, Ernesto Cuadros-Vargas. 2007.** 2007, IEEE Symposium on Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007, pp. 346 - 350.

Indoor wayfinding: developing a functional interface for individuals with cognitive impairments. **Liu, Alan L, et al. 2008.** 2008, Disability and rehabilitation. Assistive technology, pp. 69-81.

Information integration. **Hearst, M.A., et al. 1998.** 1998, IEEE Intelligent Systems and their Applications, pp. 12 - 24.

International Symposium on Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN '94. **Zue, Victor W. 1994.** 1994, Human computer interactions using language based technology, pp. I - VII vol.1 .

Investigating text input methods for mobile phones. **Kevin Curran, Derek Woods, Barry O. Riordan. 2006.** 2006, Telematics and Informatics, pp. 1 - 21.

ISWC 2010: The Latest in Wearable Computing Research. **Laerhoven, Kristof Van. 2011.** 2011, IEEE Pervasive Computing, pp. 8-10.

Kotz, Guanling Chen and David. 2000. *A survey of context-aware mobile computing research.* Hanover : Technical Report TR2000-381, Dartmouth College, Computer Science, 2000.

KY, Chen and DRJ, Basset. 2005. *The technology of accelerometry-based activity monitors: current and fture.* s.l. : Med Sci Sports Exerc, 2005. 37:S490-500.

Laitinen, Heikki. 2001. *Cellular Location Techniques.* s.l. : VIT INFORMATION TECHNOLOGY, 2001.

Large-Scale Localization fromWireless Signal Strength. **Julia Letchner, Dieter Fox, Anthony LaMarca. 2005.** 2005. National Conference on Artificial Intelligence. pp. 15-20.

Less is More: Energy-Efficient Mobile Sensing with SenseLess. **Fehmi Ben Absesslem, Andrew Phillips, Tristan Henderson. 2009.** 2009. 1st ACM workshop.

LifeTag: WiFi-Based Continuous Location Logging for Life Pattern Analysis. **J. Rekimoto, T. Miyaki, T. Ishizawa. 2007.** 2007, Lecture notes in computer science.

LMUs, Mobile location without network-based synchronization or how to do E-OTD without. **Drane, Chris R., et al. 2003.** Orlando, Estados Unidos : Location Services and Navigation Technologies , 2003. 5084.

Location Management in Pervasive Systems. **Jadwiga Indulska, Peter Sutton. 2003.** Darlinghurst, Australia : s.n., 2003. Proceedings of the Australasian information security workshop conference on ACSW frontiers.

Location Privacy in Pervasive Computing. **Beresford, Alastair R. and Stajano, Frank. 2003.** 1, s.l. : IEEE Pervasive Computing, 2003, Vol. 1.

Marinho, Roberto Irineu. 2009. *Digital citizenship in a hyper-connected society.* 2009.

Middleware for Mobile Context-Aware Applications. **Fahy, Patrick and Clarke, Siobhan. 2004.** s.l. : MobiSys, 2004.

Modeling context information in pervasive computing systems. **Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. 2002.** 2002, Lecture Notes in Computer Science, pp. 167–180.

Modeling of indoor positioning systems based on location fingerprinting. **Kaemarungsi, K. and Krishnamurthy, P. 2004.** s.l. : IEEE Society, 2004. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies . pp. 1012 - 1022.

Ness AR, Leary SD, Mattocks C, Blair SN, Reilly JJ, Wells J, Ingle. 2007. *Objectively measured physical activity and fat mass in a large cohort of children.* s.l. : PLoS Med, 2007. 4:e97.

Ontology based context modeling and reasoning using owl. **Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. 2004.** Washington, DC, USA : IEEE Computer Society, 2004. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops. p. 18.

OPF : A Distributed Context-Sensing Framework for Ubiquitous Computing Environments. **Kleek, Max Van, et al. 2006.** Seoul, Kore : Springer Verlag, 2006. International Symposium on Ubiquitous Computing Systems. pp. 82-97.

OPPORTUNITY: Towards opportunistic activity and context recognition systems. **Roggen, Daniel, et al. 2009.** 2009. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops. pp. 1-6.

PBIL PDR for scalable Bluetooth Indoor Localization. **Subramanian, Suguna P., et al. 2009.** s.l. : IEEE Computer Society, 2009. NGMAST '09 Proceedings of the 2009 Third International Conference on Next Generation Mobile Applications. pp. 170-175.

Pervasive Computing: Implications, Opportunities and Challenges for the Society. **Reddy, Y.V. 2006.** s.l. : Ieee Conferences, 2006. 1st International Symposium on Pervasive Computing and Applications. pp. 5-5.

Pervasive speech recognition. **Alewine, N., Ruback, H. and Deligne, S. 2004.** 2004, IEEE Pervasive Computing, pp. 78 - 81.

Positioning using Acceleration and Moving Direction. **Gu, Bongeun and Kwak, Yunsik. 2008.** 2008. 2008 International Conference on Information Security and Assurance. pp. 338-341.

Prediction model based on Gompertz function. **Yan, Zheng. 2009.** Beijing : IEEE, 2009. 2nd IEEE International Conference on Broadband Network & Multimedia Technology, 2009. IC-BNMT '09. pp. 893-898.

Providing location information in a ubiquitous computing environment. **Spreitzer, Mike and Theimer, Marvin. 1993.** s.l. : Proceedings of the Fourteenth ACMSymposiumon Operating System Principles, 1993.

Spreitzer, Mike and Theimer, Marvin. 1993. s.l. : Proceedings of the Fourteenth ACM Symposiumon Operating System Principles, 1993.

Recognizable-Image Selection for Fingerprint Recognition With a Mobile-Device Camera. **Lee, Dongjae, et al. 2008.** 2008, IEEE Transactions on Systems, Man, and Cybernetics, pp. 233 - 243.

Research on the dynamic comprehensive evaluation based on information integrated. **Yong-hui, Huang. 2010.** 2010, International Symposium on Computer Communication Control and Automation (3CA), pp. 225 - 228.

RFID indoor positioning using RBFNN with L-GEM. **Ding, H.-L., et al. 2010.** Qingdao : s.n., 2010. International Conference on Machine Learning and Cybernetics, ICMLC 2010.

Rough set based on modified ChiMerge algorithm and its application. **Xiao-Hong Gu, Di-Bo Hou, Ze-Kui Zhou, Dan Yu. 2003.** Guangzhou : IEEE, 2003. International Conference on Machine Learning and Cybernetics. pp. 1004 - 1008.

Schilit, William N. 1995. *A System Architecture for Context-Aware Mobile Computing. PhD thesis.* Columbia : Columbia University, 1995.

Security for Ubiquitous Computing. **Stajano, F. 2002.** Nueva York : John Wiley & Sons, 2002.

Smart Parking Applications Using RFID Technology. **Pala, Z. and Inanc, N. 2007.** s.l. : IEEE CONFERENCES, 2007. 1st Annual RFID Eurasia. pp. 1-3.

SmartBuckle: human activity recognition using a 3-axis accelerometer and a wearable camera. **Cho, Yongwon, et al. 2009.** Breckenridge, Colorado : International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments, 2009. 978-1-60558-199-6.

Some computer science issues in ubiquitous computing. **Weiser, Mark. 1993.** s.l. : Computer-Augmented Environments, 1993, Vol. Special Issue.

Soria-Morillo, L.M., et al. 2009. *Outdoor exit detection using combined techniques to increase GPS efficiency.* Sevilla, España : s.n., 2009.

Spiderweb: A social mobile network. **Sapuppo, Antonio. 2010.** 2010, European Wireless Conference (EW), pp. 475 - 481.

Standard ontology for ubiquitous and pervasive applications. **Harry Chen, Filip Perich, Timothy W. Finin, and Anupam Joshi. 2004.** s.l. : IEEE Computer Society, 2004. MobiQuitous. pp. 258–267.

Stratified/PCA: un método de preprocesamiento de datos y variables para la construcción de modelos de redes neuronales. **L., Gerardo A. Colmenares. 2003.** 16, Mérida : Revista Economía Universidad de Los Andes, 2003.

SurroundSense : Mobile Phone Localization via Ambience Fingerprinting. **Azizyan, Martin and Choudhury, Romit Roy. 2009.** New York : ACM, 2009. Proceedings of the 15th annual international conference on Mobile computing and networking. p. 261.

Sutherland, Ivan. 1963. *Ivan Sutherland's Sketchpad.* Hastings, Nebraska, United States : Carnegie Institute of Technology, 1963.

The acceptance of information, its subjective cost and the measurement of distortion. **Wilson, Roland G. 1982.** 1982, IEEE Transactions on Information Theory, pp. 967 - 971.

The coming age of calm technology. **Weiser, Mark and Seely-Brown, John. 1996.** s.l. : Xerox PARC, 1996.

The Computer for the 21st Century. **Weiser, Mark. 1991.** 1991, Scientific American, pp. 94-104.

The Context Toolkit: Aiding the Development of Context-Aware Applications. **Dey, A. K. and Abowd, G. D. 2000.** Limerick, Ireland : Workshop on Software Engineering for Wearable and Pervasive Computing, 2000.

The design and implementation of information integration framework on equipment domain based on ontology. **Li, Wang Panqing and Zengliang, Xiongwei Liu. 2010.** Shijiazhuang, China : s.n., 2010. 2nd International Conference on Networking and Digital Society (ICNDS), 2010. pp. 310 - 313.

The Java Context Awareness Framework (JCAF) - a service infrastructure and programming framework for context-aware applications. **Bardram, Jakob E. 2005.** 2005, Pervasive of Lecture Notes in Computer Science, pp. 98-115.

The study for the RFID with bluetooth positioning system. **Liu, C.-H. and Lo, J.-Y. 2010.** Chengdu : s.n., 2010. 3rd International Conference on Information Sciences and Interaction Sciences, ICIS 2010.

There is more to Context than Location. **Schmidt, Albrecht, Beigl, Michael and Gellersen, Hans-W. 1999.** 6, s.l. : Computers & Graphics Journal, Elsevier, 1999, Vol. 23.

Toward a Personal Health Society in Cardiology. **Fayn, J. and Rubel, P. 2010.** 2, s.l. : IEEE Transactions on Information Technology in Biomedicine, 2010, Vol. 14. 10.1109/TITB.2009.2037616 .

Transportation Research Record. **Toole05, L. Toole-Holt and S. Polzin and R. Pendyala:. 2005.** 2005.

Travel assistance device: utilising global positioning system-enabled mobile phones to aid transit riders with special needs. **Barbeau, S.J., et al. 2010.** 2010, Intelligent Transport Systems, IET , pp. 12-23.

Trip Destination Prediction Based on Past GPS Log Using a Hidden Markov Model. **Juan Antonio Álvarez García, Juan Antonio Ortega Ramirez, Luis González Abril, Francisco Velasco Morente. 2010.** 2010, Expert Systems With Applications, pp. 8166-8171 .

Trip destination prediction based on past GPS log using a Hidden Markov Model. **Álvarez, J. A., et al. 2009.** s.l. : Expert Systems and Applications, 2009.

Ubiquitous E-learning System for Dynamic Mini-courseware Assemblying and Delivering to Mobile Terminals. **Yushun Li, Hui Guo, Ge Gao, Ronghuai Huang, Xiaochun Cheng. 2009.** 2009, Conference on INC, IMS and IDC, NCM '09, pp. 1081 - 1086.

Understanding and Using Context. **Dey, A. K. 2001.** 2001 : Personal and Ubiquitous Computing, 2001. 0949-2054.

Universal Background Models for Real-Time Speaker Change Detection. **Wu, Ting-yao, et al. 2003.** s.l. : Proceedings of International Conference on Multimedia Modeling, 2003.

Unsupervised speaker segmentation and tracking in real-time audio content analysis. **Lu, Lie and Zhang, Hong-Jiang. 2005.** s.l. : Association for Computing Machinery, Inc., 2005, Vol. 14.

Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. **Yang, Jhun-Ying, Wang, Jeen-Shing and Chen, Yen-Ping. 2008.** s.l. : Pattern Recognition Letters, 2008. 0167-8655.

Using OWL in a Pervasive Computing Broker. **Chen, Harry, Finin, Tim and Joshi, Anupam.** 2003 : Workshop on Ontologies in Agent Systems AAMAS.

Wejchert, J. 2000. *The Disappearing Computer.* s.l. : Information Document, IST Call for proposals, European Commission, Future and Emerging Technologies, 2000.

What GPS doesn't tell you: determining one's context with low-level sensors. **Schmidt, Albrecht and Forbess, Jessica. 1999.** s.l. : Proceedings of ICECS '99. 6th IEEE International Conference on Electronics, Circuits and Systems, 1999.

Where will they turn: predicting turn proportions at intersections. **Krumm, J. 2009.** 7, s.l. : Personal and Ubiquitous Computing, 2009, Vol. 14. 10.1007/s00779-009-0248-1.

XML Meta Data. **Ahmed, K. and al, et. 2001.** s.l. : Wrox Press, 2001.

ESTRUCTURA DE LOS FICHEROS DE ALMACENAMIENTO DE INFORMACIÓN

En este anexo se recogen las estructuras de los diferentes ficheros *XML* empleados para almacenar la información contextual necesaria por parte del middleware desarrollado. Concretamente esta información será almacenada por los módulos *Data Repository* y *Context Repository*.

Comenzaremos presentando recogiendo un ejemplo correspondiente al fichero *XML* empleado por el *Data Repository* para almacenar las información procedente de los proveedores de contexto, la cual será empleada porteriormente por las diferentes aplicaciones asociadas al sistema.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <ContextProviders>
  - <Context>
      <Id>cpActivity</Id>
    - <Functions>
      - <Function>
          <idFunction>function1</idFunction>
        - <DataCollection>
          - <Data>
              <timeStamp>1222324897</timeStamp>
              <accuracy>0.7</accuracy>
              <value>4.75</value>
            </Data>
          - <Data>
              <timeStamp>1223456917</timeStamp>
              <accuracy>0.8</accuracy>
              <value>5.15</value>
            </Data>
          - <Data>
              <timeStamp>1223457121</timeStamp>
              <accuracy>0.8</accuracy>
              <value>6.20</value>
            </Data>
        </DataCollection>
      </Function>
      - <Function>
          <idFunction>function3</idFunction>
        - <DataCollection>
          - <Data>
              <timeStamp>1222488987</timeStamp>
              <accuracy>0.9</accuracy>
              <value>Walk</value>
            </Data>
          - <Data>
              <timeStamp>1222489852</timeStamp>
              <accuracy>0.9</accuracy>
              <value>Walk</value>
            </Data>
          - <Data>
              <timeStamp>1222489912</timeStamp>
              <accuracy>0.8</accuracy>
              <value>Run</value>
            </Data>
        </DataCollection>
      </Function>
    </Functions>
  </Context>
```

```
- <Context>
      <Id>cpLocation</Id>
    - <Functions>
      - <Function>
            <idFunction>functionLat</idFunction>
          - <DataCollection>
            - <Data>
                  <timeStamp>1222324897</timeStamp>
                  <accuracy>0.8</accuracy>
                  <value>37.214232</value>
              </Data>
            - <Data>
                  <timeStamp>1223456917</timeStamp>
                  <accuracy>0.9</accuracy>
                  <value>37.212123</value>
              </Data>
            - <Data>
                  <timeStamp>1223457121</timeStamp>
                  <accuracy>0.9</accuracy>
                  <value>37.213124</value>
              </Data>
            </DataCollection>
        </Function>
      - <Function>
            <idFunction>functionLongitud</idFunction>
          - <DataCollection>
            - <Data>
                  <timeStamp>1222488987</timeStamp>
                  <accuracy>0.7</accuracy>
                  <value>-6.75456</value>
              </Data>
            - <Data>
                  <timeStamp>1222489912</timeStamp>
                  <accuracy>0.9</accuracy>
                  <value>-6.89787</value>
              </Data>
            </DataCollection>
        </Function>
      </Functions>
  </Context>
</ContextProviders>
```

A continuación, se presenta un ejemplo del fichero *XML* encargado de almacenar la información relativa a los proveedores de contexto que empleada, entre otros, por las aplicaciones que realizarán consultas sobre dicha información para obtener aquellos proveedores de contexto que mejor se adapten a sus necesidades concretas.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <ContextProviders>
  - <ContextProvider>
      <Id>cpActivity</Id>
      <Name>Activity recognition using accelerometers</Name>
      <Type>Activity_recognition</Type>
      <Description>This context provider carries out an user physical activity recognition by
          using the accelerometer sensor embedded </Description>
    - <Permisions>
        <Permision>Read_acc</Permision>
        <Permision>Internet</Permision>
        <Permision>TouchScreen</Permision>
        <Permision>UI</Permision>
      </Permisions>
```

```
    - <Sensors>
        - <Sensor>
            <SensorId>accelerometer</SensorId>
            <SensorDescription>Accelerometer embedded</SensorDescription>
            <MinFrecuency>-1</MinFrecuency>
            <MaxFrecuency>100</MaxFrecuency>
            <Accuracy>0.98</Accuracy>
        </Sensor>
    </Sensors>
    - <Functions>
        - <Function>
            <FunctionId>function1</FunctionId>
            <FunctionName>Activity recognition</FunctionName>
            <FunctionDescription>Recognizes the physical activity carried out by an
                user</FunctionDescription>
            <MinFrecuency>0.1</MinFrecuency>
            <MaxFrecuency>1</MaxFrecuency>
        </Function>
        - <Function>
            <FunctionId>function2</FunctionId>
            <FunctionName>Fall detection</FunctionName>
            <FunctionDescription>Recognizes users' falls using the
                accelerometer</FunctionDescription>
            <MinFrecuency>0.1</MinFrecuency>
            <MaxFrecuency>2</MaxFrecuency>
        </Function>
        - <Function>
            <FunctionId>function3</FunctionId>
            <FunctionName>Activity frecuency</FunctionName>
            <FunctionDescription>Recognizes physical activity
                frecuency</FunctionDescription>
            <MinFrecuency>0.1</MinFrecuency>
            <MaxFrecuency>1</MaxFrecuency>
        </Function>
    </Functions>
</ContextProvider>
- <ContextProvider>
    <Id>cpLocation</Id>
    <Name>User location</Name>
    <Type>GPS_Location</Type>
    <Description>This context provider locates the user using GPS sensors</Description>
    - <Permisions>
        <Permision>Read_GPS</Permision>
        <Permision>TouchScreen</Permision>
        <Permision>UI</Permision>
    </Permisions>
    - <Sensors>
        - <Sensor>
            <SensorId>gps</SensorId>
            <SensorDescription>GPS embedded</SensorDescription>
            <MinFrecuency>-1</MinFrecuency>
            <MaxFrecuency>100</MaxFrecuency>
            <Accuracy>0.97</Accuracy>
        </Sensor>
    </Sensors>
    - <Functions>
        - <Function>
            <FunctionId>function1</FunctionId>
            <FunctionName>Latitude</FunctionName>
            <FunctionDescription>Determines the latitude component of the user's
                location</FunctionDescription>
            <MinFrecuency>-1</MinFrecuency>
            <MaxFrecuency>100</MaxFrecuency>
        </Function>
```

```
        - <Function>
            <FunctionId>function2</FunctionId>
            <FunctionName>Longitude</FunctionName>
            <FunctionDescription>Determines the longitude component of the user's
                location</FunctionDescription>
            <MinFrecuency>-1</MinFrecuency>
            <MaxFrecuency>100</MaxFrecuency>
          </Function>
        - <Function>
            <FunctionId>function3</FunctionId>
            <FunctionName>Altitude</FunctionName>
            <FunctionDescription>Determines the altitude component of the user's
                location</FunctionDescription>
            <MinFrecuency>-1</MinFrecuency>
            <MaxFrecuency>100</MaxFrecuency>
          </Function>
      </Functions>
    </ContextProvider>
</ContextProviders>
```

ANEXO II

En este segundo anexo se especifican las diversas interfaces de comunicación de la arquitectura desarrollada, las cuales permiten el paso de información entre los distintos componentes del sistema expuesto. Comenzaremos en primer lugar identificando las interfaces asociadas a la sección de *Proveedores de Contexto*, para seguidamente exponer las relativas al *Núcleo* de la arquitectura y finalizando así con las asociadas a las *Aplicaciones* que hacen uso del middleware para obtener la información requerida.

PROVEEDORES DE CONTEXTO

Interfaces

Con el fin de definir una arquitectura general para todos los *ContextProviders*, es necesario crear una serie de interfaces que faciliten la comunicación con el núcleo. Durante esta sección describiremos en profundidad cada una de las interfaces de datos presentes en el módulo *proveedor de contexto* del middleware propuesto.

En primer lugar será necesario definir una interfaz que permita la comunicación entre el *ContextProvider* y el *Core*. La definición de esta interfaz es realmente importante puesto que el *ContextProvider* necesitará enviar información relativa a los valores del contexto hacia el *Core*, para que este último pueda distribuir los datos entre las aplicaciones que lo requieran. La interfaz recibirá el nombre de *IContextProvider* y contendrá los métodos y atributos recogidos en la Figura 123.

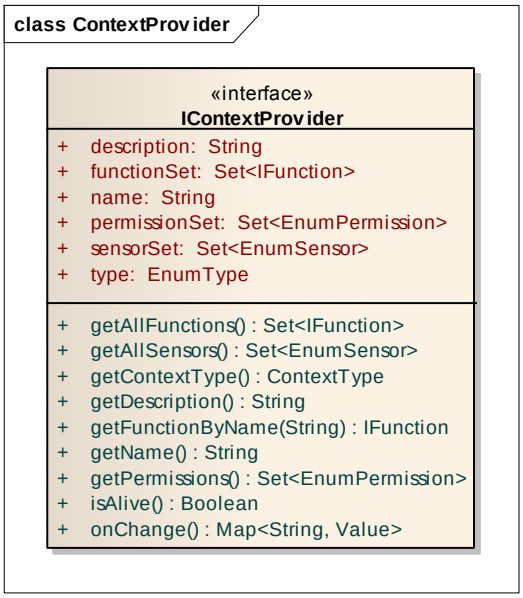


Figura 123: Interfaz IContextProvider que proporciona los métodos necesarios para la comunicación entre el Core y el proveedor de contexto concreto implementando

Para ello se define la interfaz *IContextProvider* que consta de los siguientes atributos:

- *Description*. Aporta una descripción sobre el *ContextProvider* concreto que se implementa. Dicha descripción suele ser interesante a la hora de determinar, por parte de la aplicación, el proveedor de contexto que mejor satisface sus necesidades.

- *FunctionSet*. Este atributo contendrá el conjunto de funciones que el *ContextProvider* ofrece, las cuales, como se comentó anteriormente, deben formar parte del ámbito del contexto que se está implementando. Gracias al atributo *functionSet* es posible ejecutar diferentes funciones sobre los valores del contexto a partir de una serie de parámetros de entrada que se definirán en la propia *Function*.

- *Name*. Contiene el nombre que recibe el *ContextProvider* desarrollado. Este nombre deberá ser identificativo del ámbito contextual sobre el cual trabaja.

- *PermissionSet*. Todo *ContextProvider* tendrá una serie de permisos asociados, los cuales se definirán en este atributo. Puesto que el conjunto de permisos está previamente tipificado, el proveedor de contexto puede hacer uso de ellos para ser descrito. Desde el punto de vista del *Core*, este conjunto de permisos será necesario para identificar *ContextProviders* potencialmente peligrosos o incompatibilidades entre los distintos *ContextProviders* empleados por una determinada aplicación. Claro ejemplo de ello sería un proveedor de contexto que haga uso de acceso a redes sociales, acceso a internet y acceso al sensor de posicionamiento, lo cual podría ser un indicio de que podrían ser publicadas las localizaciones del usuario en una determinada red social, violando la privacidad del usuario. Aunque generalmente las políticas de seguridad no son estrictas, se informará al usuario de esta incompatibilidad, siendo éste el encargado de aceptar o no la responsabilidad. De esta función se encargará el *Core* del middleware.

- *SensorSet*. El conjunto de sensores que emplea un determinado *ContextProvider* es de especial interés a la hora de realizar búsquedas sobre proveedores de contexto que ofrezcan una determinada funcionalidad. Esta búsqueda generalmente será iniciada a través de una petición de la capa de *Applications*. Por otro lado, el conocer los diferentes sensores que emplea un *ContextProvider* dará la oportunidad al *Core* de detener o activar automáticamente los proveedores de contexto en función de si está o no presente el sensor empleado en el dispositivo. Esto es extremadamente útil en el sector de los dispositivos móviles, ya que las características de cada uno de ellos varía en gran medida, pudiendo producirse ciertas incompatibilidades entre un *ContextProvider* y un dispositivo que no integre los sensores necesarios.

- *Type*. Los diferentes *ContextProviders* se integran dentro de un conjunto de tipos de contexto, de manera que la información ofrecida por estos proveedores se relacionan con un determinado área contextual. La finalidad de determinar el tipo de contexto sobre el cual actúa el *ContextProvider* facilitará a la aplicación la búsqueda de proveedores que ofrezcan funcionalidad sobre un determinado área.

Por otro lado se definen una serie de funciones que dan acceso a los atributos definidos anteriormente, así como otras que permiten la comunicación con el *Core* del middleware. A continuación se describen las funciones que dan acceso al *Core* para gestionar las distintas actualizaciones sobre los valores contextuales así como para determinar si el *ContextProvider* está o no en un estado funcional.

- *isAlive()*. La función *isAlive* permite al *Core* determinar si un *ContextProvider* está activo o si, por el contrario, se ha desconectado de la plataforma. Gracias a esta función se podrá

eliminar el proveedor de contexto de la lista de éstos que mantiene internamente el *Core* con el fin de ofrecer información consulta a la capa de *Applications*.

- *onChange()*. Será la función encargada de alertar al *Core* cuando se produzca un cambio en uno o varios de los valores de contexto que maneja el proveedor. Estos valores de contexto serán concretamente los datos ofrecidos por el conjunto de funciones que implementa el *ContextProvider* o por otro lado, información generada internamente por el *ContextProvider* que deba ser gestionada por el *Core*.

Una vez expuestos las diferentes funcionalidades que ofrece la interfaz *IContextProvider*, se describirá la interfaz *IFunction*, la cual se muestra en la Figura 124, encargada de representar a todas y cada una de las funciones que pueden ser empleadas por las *Applications* a través del *Core*. Dichas funciones serán ofrecidas por los *ContextProviders* que forman parte de la arquitectura.

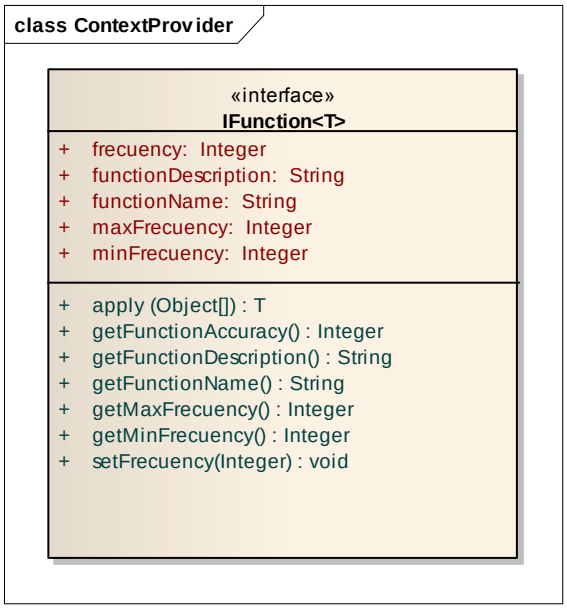


Figura 124: Interfaz IFunction que debe ser implementada por las diferentes funciones existentes en el ContextProvider

En cuanto a los diferentes atributos presentes en la interfaz *IFunction*, al igual que en el caso anterior, se describen a continuación con una mayor profundidad:

- *Frecuency*. La frecuencia a la que se actualiza una determinada función es importante a la hora de realizar actualizaciones automáticas de la misma. Esto tiene especial interés en datos que son susceptibles a cambios constantes como, por ejemplo, la localización de un usuario o la actividad física que está realizando. Generalmente la frecuencia de actualización puede ser ajustada por la propia *Application* con el fin de adaptarla a sus propias necesidades. Esto hace que cada aplicación trabaje con un conjunto de datos obtenidos con una frecuencia diferente en función de los requisitos necesarios.

- *FunctionDescription.* Al igual que ocurría en el caso de la interfaz *IContextProvider*, las diferentes aplicaciones deben conocer en detalle la funcionalidad ofrecida por un determinado módulo de contexto. En este caso, la información será relativa a la propia función, indicando de manera textual los datos obtenidos por la misma.

- *FunctionName*. El nombre de la función es importante para identificarla entre el conjunto de funciones ofrecidas por el *ContextProvider*. Gracias al nombre, la función puede ser llamada desde las *Applications* con tan sólo conocer el *ContextProvider* al que pertenece y su nombre.

- *MaxFrecuency*. Este atributo determinará cuál es la frecuencia máxima a la que está permitido trabajar con una determinada función. Generalmente, esta frecuencia máxima de actualización vendrá determinada por la frecuencia de actualización de los sensores empleados o de la complejidad computacional de la propia *Function*.

- *MinFrecuency*. Relacionada con la anterior, en este caso el atributo indica la frecuencia mínima a la que es posible trabajar con la función. Junto con *MaxFrecuency*, permiten determinar el rango frecuencial con el que puede ser actualizada la función descrita.

Paralelamente a los diferentes métodos que permiten consultar o modificar los valores de los atributos de la *Function* descrita, existe un método a través del cual es posible obtener el valor de la función para un determinado conjunto de parámetros de entrada, la cual se describe a continuación.

- *Apply(Object[])*. La función *Apply* ejecuta la función descrita con los parámetros pasados por argumento. El uso de esta función puede deberse a dos motivos principalmente, una actualización programada por el *Core* del middleware, o bien una llamada explícita por parte de una *Application* para obtener un valor puntual. En general, sea cual sea el origen de la llamada, la función A*pply* procesará los datos contextuales con los cuales trabaja y retornará un tipo de dato *T* genérico, el cual viene dado por la propia definición del objeto *Function*, que deberá concretar el tipo de dato genérico para que pueda ser empleado por las distintas *Applications* que lo requieran.

- *SetFrecuency(int)*. Se trata del único método modificador de la interfaz *IFunction*, y se encarga de determinar la frecuencia a la que trabajará la función descrita. Es estrictamente necesario que el parámetro de entrada esté comprendido entre la frecuencia máxima y la mínima de la función, ya que de lo contrario se establecerá automáticamente al límite más cercano al valor introducido.

Definición de clasificaciones contextuales

En el caso del middleware, la definición de una clasificación que permita compartir información entre las diferentes aplicaciones y proveedores de contexto es indispensable. En esta sección se pretende desarrollar una clasificación contextual flexible y eficiente que cubra la amplia gama de posibles contextos que maneja el sistema. Concretamente se diseñaran aquellas clases ontológicas que forman parte del módulo de provisión de contexto.

En general, los objetivos más importantes en el diseño de una ontología contextual son (An Ontology for Mobile Device Sensor-Based Context Awareness, 2003):

- **Simplicidad**: las expresiones utilizadas y las relaciones deben ser tan simple como sea posible para simplificar el trabajo de los desarrolladores de aplicaciones y del propio middleware.

- **Flexibilidad y extensibilidad**: la ontología debe facilitar la adición de nuevos elementos contexto y nuevas relaciones entre ellos si fuera necesario.

- **Generalidad**: la ontología no debe limitarse a un tipo especial de contexto, sino que debe soportar diferentes tipos de contexto.

- **Expresividad**: la ontología debe permitir describir los estados contextuales con el mayor nivel de detalle posible.

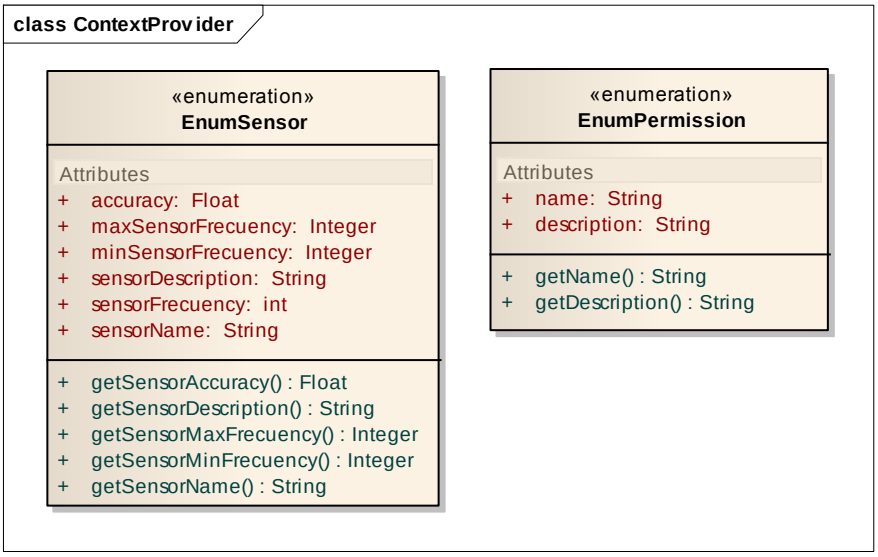


Figura 125: Clasificación de sensores y permisos para los Context Providers

Con el fin de permitir a los proveedores de contexto especificar los diferentes sensores empleados para obtener la información, se definen los siguientes atributos que permitirán englobar la actividad del *Context Provider* en función del conjunto de sensores empleados. Además, esto permitirá determinar si es posible ejecutar un determinado proveedor de contexto en un dispositivo concreto, en base a que dicho dispositivo contenga o no lo sensores necesarios para la ejecución.

Nombre
Location
Speed
Temperature
Humidity
Luminosity
Time
Accelerometer
Pression
Electromagnetism

Sound

Tabla 16: Nombre de los diferentes sensores que pueden ser empleados por los Context Providers

Los valores de los distintos permisos presentes en el sistema se recogen en la Tabla 17. Estos valores permitirán al *Core* y a las propias aplicaciones tener un mejor concepto del *Context Provider* que proporciona los datos, sabiendo así las acciones que llevará a cabo sobre el sistema conociendo sus permisos. Además, permitirá detectar diferentes incompatibilidades entre los proveedores de contexto, gracias a la línea de investigación abierta basada en la búsqueda de configuraciones compatibles del sistema.

Nombre	Tipo
Hardware	Physical
Speed	Physical
Temperature	Physical
Humidity	Physical
Luminosity	Physical
GPS	Physical
NFC	Physical
Time	Physical
Acceleration	Physical
Pression	Physical
Barometric altitude	Physical
Electromagnetism	Physical
Sound	Physical
Camera	Physical
Flashlight	Physical
Software	Logical
Lenguage	Logical
Web Navigation	Logical
Playlist	Logical
Calendar	Logical
Phone	Logical
Labeled Location	Logical
Locals	Logical
Physical Activiry	Logical
Destiny prediction	Logical

On-line	Logical
Proximity	Logical
Traffic states	Logical
Socials	Logical
Weather	Logical
Communication	Logical
Contacts/Phonebook	Logical

Tabla 17: Valores de la clasificación de permisos

NÚCLEO

Interfaces

A continuación analizaremos las diferentes interfaces que conforman la información manejada por el *Core*. Será precisamente gracias a la interfaces descritas a continuación mediante las cuales se podrá realizar el intercambio de información entre los distintos módulos que componen el *Core*.

Context Section

Comenzaremos analizando las diferentes interfaces empleadas en el *Context Section*, permitiendo así analizar con detalle la funcionalidad de cada una de ellas. Para ello, se procederá analizando una a una el conjunto de interfaces describiendo los atributos y los distintos métodos que poseen cada una de ellas.

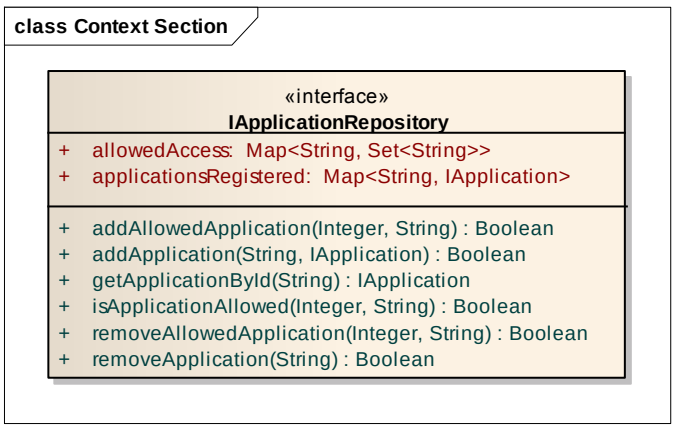


Figura 126: Diagrama de la interfaz IApplicationRepsitory

La interfaz *IApplicationRepository* será la interfaz de la clase encargada de almacenar toda la información relativa a las aplicaciones que hará uso del middleware. El objeto de ésta es gestionar los permisos de acceso a los diferentes *ContextProviders* que se asocian a cada una de las aplicaciones, ofreciendo en este sentido diferentes métodos para añadir, eliminar y consultar dicho conjunto de permisos.

La interfaz *IApplicationRepository* consta de los siguientes atributos:

- *Map<String, Set<String>> allowedAccess*. Este atributo será el encargado de almacenar un par clave-valor que permite relacionar cada con los diferentes *ContextProviders* a los que se le ha permitido el acceso. De esta forma, se mantiene los requisitos de seguridad del sistema, puesto que una aplicación tan sólo podrá acceder a aquella información contextual que sea generada por aquellos *ContextProviders* a los que se haya suscrito previamente a través del módulo *Context Discover*.

- *Map<String, IApplication> applicationsRegistered*. Para asociar las diferentes aplicaciones a sus identificadores, el atributo actual almacenará pares clave-valor para proveer al sistema de la lista de aplicaciones instaladas en el sistema asociadas a algún contexto junto al identificador unívoco generado a la hora del registro. Dicho identificador será la manera en la que se hará referencia a la aplicación a lo largo de todo el *Core*. Es especialmente interesante observar que junto al identificador, el valor almacenado es un objeto que debe implementar la interfaz *IApplication*. Esto se debe a que en numerosas ocasiones a lo largo del *Core*, es necesario hacer llamadas a funciones de las aplicaciones con el fin de establecer una interacción con ellas. Esta interacción se conseguirá precisamente gracias al almacenamiento de una referencia a ellas a través de la interfaz *IApplication*. Un ejemplo claro de esta interacción sería el momento en el que se debe comunicar a la aplicación la activación de un determinado evento.

Los métodos encargados de dar funcionalidad a la interfaz descrita serán los siguientes:

- *Boolean isApplicationAllowed(String idApplication, String idContext)*. Este método determina si una determinada aplicación tiene permitido el acceso a un determinado contexto. Para ello, utiliza un identificador de Proveedor de Contexto y otro de Aplicación, ambos pasados por parámetro, y comprueba si existe el par clave-valor que relacione ambos elementos en el mapa definido por el atributo *allowedAccess*. Como se comentó anteriormente, para identificar una aplicación se empleará el identificador global asignado a la aplicación a la hora de la realización del registro, mientras que para identificar del *ContextProvider* se usará el identificador global determinado por el sistema, determinado por el módulo *Context Repository*.

- *Boolean addAllowedApplication(IApplication application, String idContext)*. El método actual añadirá un contexto al conjunto de contextos a los que se permite el acceso por parte de la aplicación. Una vez una aplicación solicite la suscripción a un *ContextProvider* y obtenga permiso por parte del *Context Discover*, el cual comprobará que no existen incompatibilidades, será necesario añadir el par contexto-aplicación al conjunto de contextos permitidos por la aplicación. Esto se realizará añadiendo una nueva entrada al atributo *allowedAccess* que asocie la aplicación con el nuevo contexto registrado. Si el contexto ya se encontraba asociado a la aplicación, no será necesario modificar la estructura, aunque el valor devuelto por el método será "*true"*.
  Podría darse el caso que fuera necesario añadir la aplicación al conjunto de aplicaciones registradas por el sistema, lo cual se produciría la primera vez que dicha aplicación se suscribe a un contexto. En tal caso, se deberá registrar previamente la aplicación en el atributo *applicationsRegistered*.
  En caso de que no haya sido posible añadir la asociación entre la aplicación y el contexto por cualquier motivo será devuelto el valor "*false"*.

- *Boolean removeAllowedApplication(String idApplication, String idContextProvider)*. Esta función brinda la posibilidad a las aplicaciones de que se elimine la suscripción a un determinado contexto. Esto se realizará generalmente cuando se finalice el uso del proveedor de contexto o cuando la aplicación se suscriba a uno nuevo cuya información es incompatible con la anterior. Además, el propio sistema puede hacer uso de esta función con el fin de liberar suscripciones a contextos de aplicaciones que se eliminen del sistema.
  En caso de que exista un par clave-valor que corresponda a la aplicación y al contexto que se desea eliminar, el proceso consiste en remover dicho par del atributo *allowedAccess*. Con el fin de reducir el espacio necesario para el almacenamiento de información, si el elemento a eliminar se trata del último *ContextProvider* asociado a la aplicación, se eliminará completamente el registro de la aplicación en el sistema, es decir, se procederá a la eliminación tanto del par clave-valor del atributo *allowedAccess* como del atributo *applicationsRegistered*.
  Este método devolverá "*true"* siempre y cuando se haya eliminado la asociación entre el contexto y la aplicación indicados. En caso de que se produzca un error o el contexto no se encontrara previamente asociado a la aplicación, el valor devuelto será "*false"*.

- *IApplication getApplicationById(String idApplication)*. Otra funcionalidad básica que ofrece este módulo al resto del sistema es la posibilidad de acceder a una determinada aplicación a través de su interfaz. La razón de esta funcionalidad se debe a que en numerosos puntos del *Core*, es necesario acceder a ciertos métodos de la aplicación que permitirán la comunicación con ésta. Puesto que en el resto de la aplicación tan sólo se hará referencia al identificador de la aplicación, será necesario dotar de un método que, a partir de dicho identificador, devuelva la interfaz de la aplicación asociada.
  Esta funcionalidad se ofrece a partir del método *getApplicationById*, el cual será el encargado de obtener el objeto asociado al identificador de la aplicación pasado por parámetro en el atributo *applicationsRegistered*.
  En caso de que exista el identificador de la aplicación en el atributo *applicationsRegistered*, el método actual devolverá la referencia a la interfaz. En caso contrario el valor devuelto será nulo.

- *Boolean addApplication(IApplication Application)*. Cuando una aplicación solicite la suscripción por primera vez a un *ContextProvider* y obtenga permiso por parte del módulo *Context Discover*, será necesario añadir una referencia a esta aplicación a la lista de aplicaciones del sistema, de modo que quede registrado dentro del mismo el par compuesto por el identificador de la aplicación y el propio objeto que hacer referencia a la aplicación, es decir, el *IApplication*. En este caso, el identificador de la aplicación será obtenido por este módulo en función del resto de identificadores, asegurando por tanto que el identificador generado sea único en todo el sistema. De esta forma, será totalmente transparente a la aplicación la generación de los identificadores.
  Si la aplicación se ha añadido correctamente al repositorio o bien existía previamente, se devolverá el valor "*true"*. Por el contrario, si se ha producido algún error a la hora de añadir la aplicación, el valor devuelto será "*false"*.

- *Boolean removeApplication(String idApplication)*. Cuando una aplicación sea eliminada del sistema, ya sea porque la propia aplicación lo ha pedido expresamente o porque no tiene asociado ningún *Context Provider*, esta debe ser eliminada completamente del *Core*, por lo que será necesario remover las entradas oportunas en los atributos que componen la

interfaz actual. En caso de que el método actual sea llamado desde *removeAllowedApplication*, tan sólo será necesario eliminar el registro asociado en el atributo *applicationsRegistered*. Por el contrario, si se trata de una solicitud de eliminación expresa, es necesario eliminar los registros de ambos atributos, por lo que en primer lugar se eliminará todas las referencias a los contextos empleados, las cuales se encuentran en el atributo allowedApplications y posteriormente, la referencia a la propia aplicación existente en el mapa definido por el atributo *applicationsRegistered*.
El método actual devolverá un valor lógico, el cual se ajustará a "*true"* siempre y cuando la eliminación de la aplicación y sus contextos se realice sin problemas. En caso de que se produzca un error al eliminar la aplicación del repositorio o si no existe la aplicación en el sistema, se comunicará devolviendo un valor "*false"* a la salida.

Continuamos el análisis con la interfaz *IContextDiscover*, cuyos métodos y atributos se muestran a continuación.

Figura 127: Diagrama de la interfaz IContextDiscover

Esta interfaz permite realizar consultas sobre los diferentes *ContextProviders* que se encuentran registrados en la aplicación. Estas consultas son de utilidad de cara a brindar a las aplicaciones un método sencillo y eficaz a través del cual obtener un *ContextProvider* que se acoja a sus necesidades, evitando así la creación de *ContextProviders* ad-hoc para determinadas aplicaciones.

Internamente, la clase que implemente dicha interfaz deberá acceder al módulo *Context* Repository, el cual estará compuesto por un almacén de datos en formato *XML* que permitirá almacenar toda la información relativa a los proveedores de contexto. El resultado de los diferentes métodos del *Context* Discover que se estudiarán a continuación será una vista sobre dicho conjunto de datos.

En general, podríamos clasificar los métodos de la interfaz *IContextDiscover* en dos grande grupos; por un lado aquellos métodos que devuelve un fichero *XML* resultado de filtrar el conjunto de datos original por los parámetros de búsqueda y, por otro lado, aquellos métodos que devuelven una interfaz de información en la que se recogen los diferentes contextos o funciones resultantes de la búsqueda. El primero de estos grupos tiene la ventaja de que es posible realizar filtros en la propia aplicación, obteniendo el conjunto completo de proveedores de contexto. Sin embargo, el uso a nivel básico de estos datos podría complicarse debido al procesamiento necesario que se debe realizar sobre los datos en *XML*. Este problema se soluciona gracias al segundo grupo de métodos, cuyos resultados son objetos que envuelven los valores de los atributos de las funciones o contextos resultantes del filtrado.

Veamos los diferentes métodos de la interfaz *IContextDiscover*.

- *Boolean isContextOnRepository(String idContextProvider)*. Este método será empleado por las aplicaciones y por el propio sistema para determinar si un *ContextProvider* determinado por su identificador está o no presente en el conjunto de proveedores inscritos en la plataforma. En ocasiones este método es útil para comprobar que un *ContextProvider* sigue estando activo en el sistema.

- *getAllContext*. En ambas versiones, es decir, devolviendo un *XML* o un conjunto de elementos *IContextProviderInformation*, esta función obtendrá todos los contextos activos en el sistema en el momento de la llamada. No se recomienda su uso debido a la elevada cantidad de información devuelta, por lo que en la mayoría de las situaciones se podría sustituir por un método que realiza un filtrado de los contextos.

- *getContextByFrecuency(Integer frecuency)*. Devuelve un filtrado de aquellos contextos cuya frecuencia de actualización sea igual a la que se pasa por parámetro.

- *getContextByFunctionName(String functionName)*. Devuelve un contexto que contenga la función cuyo nombre sea igual al que se pasa por parámetro. En este caso el elemento devuelto será un único objeto de tipo *IContextInformation* o *XML*, en función de si se trabaja con la información en bruto o con la clase envoltura.

- *getContextByName(String name)*. Devuelve un contexto cuyo nombre sea igual al que se pasa por parámetro. En este caso, al igual que en el anterior, el elemento devuelto será un único objeto de tipo *IContextInformation* o *XML*, en función de si se trabaja con la información en bruto o con la clase envoltura.

- *getContextByPermission(Set<EnumPermission> permission)*. Devuelve un conjunto de contextos que contengan al menos los permisos que se pasan por parámetro.

- *getContextBySensorName(String sensorName)*. Devuelve un conjunto de contextos que cumplan la restricción de que trabajen al menos con el sensor cuyo nombre se pasa por parámetro.

- *getContextBySubDescription(String subdescription)*. Devuelve aquellos contextos cuya descripción contenga la subcadena que se pasa por parámetro.

- *getContextByType(EnumContextType type)*. Como se comentó previamente, los contextos tienen asociado un tipo, el cual será el campo contextual en el que trabajen. Esta función realizará una búsqueda de todos aquellos contextos cuyo tipo se corresponda con el pasado por parámetro.

- *getFunctionByFrecuency(Integer frecuency)*. Devuelve un conjunto de funciones que cumplan con la condición de que la frecuencia de actualización coincida con la que se pasa por parámetro. En el caso de trabajar con el objeto envoltura, el tipo devuelto será un Set<*IFuncionInformation*>.

- *getFunctionByMaxFrecuency(Integer frecuency)*. Devuelve un conjunto de funciones que cumplan con la condición de que la frecuencia de máxima de actualización coincida con la que se pasa por parámetro. En el caso de trabajar con el objeto envoltura, el tipo devuelto será un Set<*IFuncionInformation*>.

- *getFunctionByMinFrecuency(Integer frecuency)*. Devuelve un conjunto de funciones que cumplan con la condición de que la frecuencia mínima de actualización coincida con la que se pasa por parámetro. En el caso de trabajar con el objeto envoltura, el tipo devuelto será un Set<*IFuncionInformation*>.

- *getFunctionByReturnedType(Class clase)*. Devuelve un conjunto de funciones que cumplan con la condición de que el tipo del valor devuelto coincida con el que se pasa por parámetro.

- *getFunctionBySubDescription(String subdescription)*. Devuelve aquellas funciones cuya descripción contenga la subcadena que se pasa por parámetro.

- *getSensorsByAccuracy(Double accuracy)*. Devuelve un conjunto de sensores cuya precisión sea mayor o igual a la que se pasa por parámetro.

- *getSensorsByMaxFrecuency(Integer frecuency)*. Devuelve un conjunto de sensores cuya frecuencia máxima de actualización sea igual a la que se pasa por parámetro.

- *getSensorsByMinFrecuency(Integer frecuency)*. Devuelve un conjunto de sensores cuya frecuencia mínima de actualización sea igual a la que se pasa por parámetro.

- *getSensorsByName(String name)*. Devuelve el sensor cuyo nombre sea igual al que se pasa por parámetro.

- *Boolean registryApplication(IApplication application, String idContextProvider)*. Cuando una aplicación desea suscribirse a un *Context Provider* (lo cual es el primer paso si quiere

acceder al Core, ya sea para suscribirse a un evento o para obtener los datos), el sistema debe asegurarse de que cumple las políticas de seguridad establecidas. Para ello, se ofrece a la aplicación el método *registryApplication*, que será el punto de entrada al *Core* para la suscripción a un contexto.
Este método internamente ejecutará *canSubscribe* para comprobar si una aplicación puede suscribirse a un contexto y, en caso afirmativo, registra la aplicación en el módulo *Application Repository*, introduciendo una copia del objeto que representa a la aplicación en la estructura adecuada, así como la asociación establecida entre la aplicación y el *Context Provider*. Este método, a su vez, hará uso del *Application Repository* para llevar a cabo su funcionalidad.
El método actual devolverá un valor lógico ajustado en función de si se ha llevado a cabo adecuadamente el registro o si por el contrario no se ha podido registrar la aplicación. Este segundo caso se podría dar cuando no se satisfagan las condiciones de seguridad establecidas en el módulo debido a, por ejemplo, una incompatibilidad entre el contexto al que se desea suscribir la aplicación y otro contexto al que se encuentra suscrita anteriormente.

- *Boolean unregistryApplication(IApplication application, String idContextProvider)*. Del mismo modo que una aplicación tiene la posibilidad de suscribirse a un *Context Provider*, este método permite al sistema ofrecer la posibilidad de eliminar este registro del repositorio. Este método puede ser invocado desde el propio *Core*, debido por ejemplo al hecho de que una aplicación haya dejado ser válida y se deba borrar su registro, o desde la propia aplicación, cuando esta desea dejar de hacer uso de un *Context Provider*.
  Este método ejecutará internamente el método *removeAllowedApplications* del módulo *Application Repository* para eliminar el registro de la aplicación en el contexto que se pasa como parámetro.
  El método *unregistryApplication* devolverá "*true"* si la eliminación se ha realizado correctamente y "*false"* en caso contrario.

- *Boolean canSubscribe(IApplication application, String idContextProvider)*. Cuando una aplicación quiere suscribirse a un *Context Provider* (lo cual es el primer paso si quiere acceder al *Core* para obtener la información contextual procedente de los proveedores), debe asegurarse de que se cumplen todas las políticas de seguridad establecidas en el sistema. Para ello, se ofrece a la aplicación el método *canSubscribe*, el cual devolverá un valor lógico a *true* si no existen incompatibilidades entre el contexto al que se desea suscribir la aplicación y el conjunto de contextos a los que se encuentras inscrita, y *false* en caso contrario.

- *String onDeleteContextProvider(String idContextProvider)*. Cuando el módulo *Context Subscription* detecta que un *Context Provider* ha dejado de estar disponible en el sistema, eliminará el *Context Provider* del módulo *Context Repository* y además, se enviará un evento a través del módulo *Context Listener*. Dicho evento podrá ser capturado por cualquier módulo interno del Core.

Con el fin de almacenar los distintos proveedores de contextos dados de alta en el sistema así como para acceder a dichos contextos por parte del resto de las aplicaciones, se define la interfaz *IContextRepository*. Siguiendo el diseño del resto de repositorios, contará de un sistema central de almacenamiento compuesto por un sistema de ficheros *XML*. De esta forma, la interfaz encargada de almacenar dichos valores constará de los siguientes métodos y atributos.

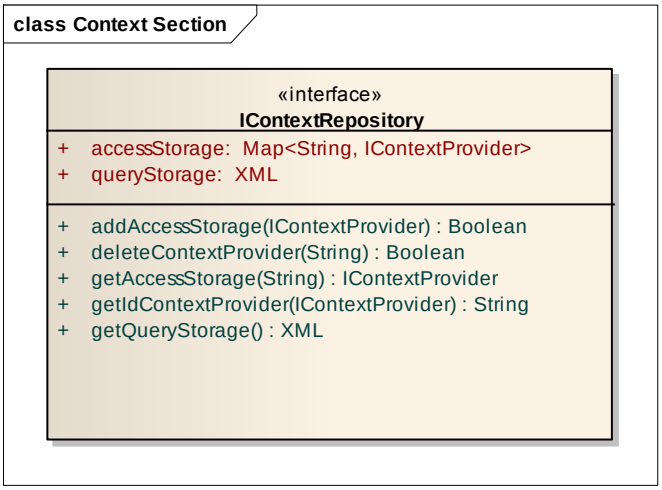


Figura 128: Diagrama de la interfaz IContextRepository

A continuación se describen los atributos de la interfaz:

- *XML queryStorage*. Con el fin de almacenar la información relativa a los contextos, se define este atributo que permitirá realizar esta funcionalidad mediante el almacenamiento en ficheros *XML*. La estructura de este fichero *XML* se puede ver representada en el *Anexo II*.

- *Map<String, IContextProvider> accessStorage*. Este atributo permitirá almacenar los diferentes *ContextProviders* asociando al identificador del mismo una referencia al propio proveedor. Esto será de gran utilidad a la hora de iniciar el protocolo de comunicación con la capa inferior de contextos.

Con el fin de acceder y gestionar los atributos internos de la clase encargada de implementar la interfaz descrita, se declaran los siguientes métodos.

- *Boolean addAccessStorage(IContextProvider contextProvider)*. Este método será el encargado de añadir un nuevo *ContextProvider* al repositorio de contextos de la aplicación. Para ello será necesario comprobar previamente el conjunto de permisos del contexto para asegurar que no exista ninguna incompatibilidad. Una vez realizado este proceso se incluirá la descripción del *ContextProvider* en el repositorio *XML* así como la referencia al propio proveedor de contextos en el atributo *accessStorage*.
  Si la adición del proveedor de contexto se ha realizado correctamente, la llamada al método devolverá "*true*", en cambio, si se ha producido un error o ya existía el proveedor, el resultado será "*false*".

- *Boolean deleteContextProvider(String idContextProvider)*. Método encargado de eliminar del repositorio el *ContextProvider* cuyo identificador se pasa por parámetro. Este método será ejecutado en caso de que el propio *ContextProvider* desee eliminar su suscripción del middleware, o bien podría ser llamado desde el propio *Core* tras recibir una serie de señales *ping* sin respuesta, lo que podría indicar que el *ContextProvider* no está disponible.

El resultado de este método será "*true"* o "*false*" en función de si ha sido posible o no respectivamente la eliminación. Si el proveedor cuyo identificador se pasa por parámetro no existe, el resultado será *false*.

- *IContextProvider getAccessStorage(String idContextProvider)*. El objetivo de este método es proveer a los módulos que así lo deseen una referencia a la interfaz del *ContextProvider* cuyo identificador se pasa por parámetro. Como se comentó anteriormente, esto es especialmente útil cuando se desea realizar una comunicación con el proveedor de contexto.

- *String getIdContextProvider(IContextProvider contextProvider)*. Dado que el resto de módulos del sistema hará referencia a los diferentes *ContextProviders* a través de su identificador, este método será el encargado de obtener el identificador asociado al proveedor que se pasa por parámetro. En caso de que no existiera el proveedor de contexto en el sistema, devolverá el siguiente identificador disponible.

- *XML getQueryStorage()*. Este método tan sólo es accesible desde el interior del *Context Repository*, ya que será el encargado de devolver la base de datos *XML* que soporta el almacenamiento de los proveedores de contexto registrados.

Para permitir la actualización de información contextual por parte de los diferentes *ContextProviders* que forman parte del sistema, es necesario una suscripción al *middleware* a través del *Core*. Esta suscripción se llevará a cabo mediante un objeto que implementará la interfaz *IContextSubscription*, la cual se describe a continuación.

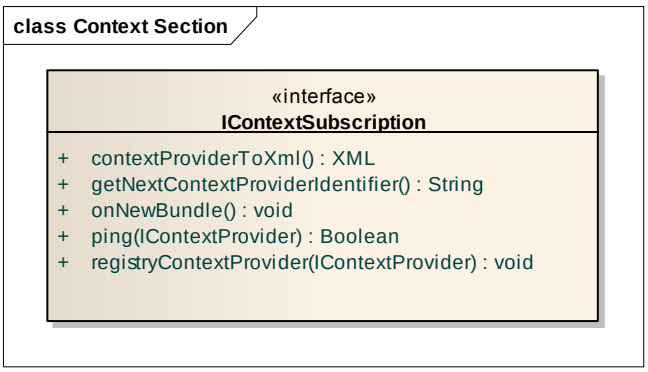


Figura 129: Diagrama de la interfaz IContextSubscription

Esta interfaz no posee ningún atributo, ya que toda la funcionalidad de almacenamiento de información se delegará en el *Context Repository*, el cual será el encargado de gestionar toda la información relativa a los *ContextProviders* inscritos en el sistema.

En cuanto a los métodos, la interfaz *IContextSubscription* se compone de las siguientes funciones que permitirán suscribir los *ContextProviders* que así lo requieran.

- *XML contextProviderToXml()*. Este método permite generar una descripción en formato *XML* de toda la información relativa al *ContextProvider* que realiza la suscripción.

- *String getNextContextProviderIdentifier()*. Con el fin de generar un identificador unívoco para cada uno de los *ContextProviders* suscritos a la plataforma, este método será el único encargado de generar dichos identificadores en base a la colección de elementos generados previamente.

- *Void onNewBundle()*. Este método será llamado por parte del framework de *OSGi* cuando se inscribe un nuevo *Bundle* en el sistema. Esto será importante a la hora de detectar los diferentes *ContextProviders* que se añaden al sistema y, en ese caso, iniciar la comunicación con dichos proveedores de contexto para incluirlos en el middleware desarrollado.

- *Boolean ping(IContextProvider contextProvider)*. Al igual que ocurría en el caso de las aplicaciones, con el fin de determinar si un *Context Provider* previamente inscrito en la plataforma sigue activo, la función *ping* llamará al método *isAlive* del proveedor de contexto que recibe por parámetro. Esto permitirá conocer el estado del *Context Provider* sobre el que se llama.

- *Void registryContextProvider(IContextProvider contextProvider)*. Una vez el sistema ha detectado un nuevo proveedor de contexto instalado en el dispositivo, deberá encargarse de registrarlo en el repositorio creado a tal efecto (*Context Repository*). De esta manera, el *Core* podrá trabajar con dicho *Context Provider* para generar información contextual. Este proceso se realizará a través del método *registryContextProvider*, en el cual se delega la escritura de la información asociada al proveedor de contexto en el módulo Context Repository. Hace uso del método *getNextContextProviderIdentifier* para obtener un identificador de contexto global, el cual será un dato necesario para el almacenamiento de la información del Proveedor de Contexto.

Con el fin de monitorizar los cambios que se produzcan en el repositorio descrito anteriormente, la clase encargada de implementar la interfaz *IContextListener*, la cual se describe a continuación, proveerá de la lógica necesaria para la suscripción a los eventos de cambios del repositorio.

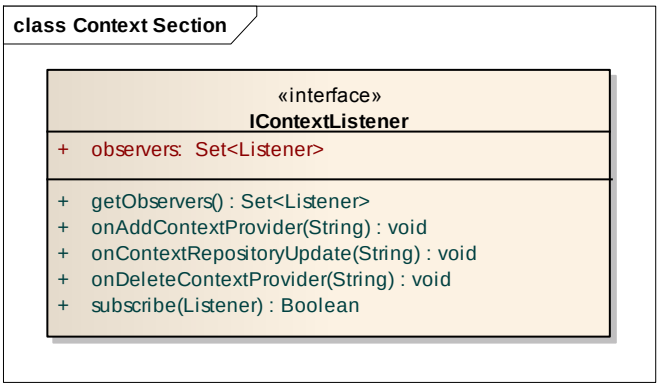


Figura 130: Diagrama de la interfaz IContextListener

La interfaz descrita se compone de un solo atributo, que permitirá almacenar la lista de elementos suscritos a los eventos.

- *Set<Listener> observers*. Este atributo es el encargado de almacenar las clases inscritas a la difusión de eventos relacionados con el *Context Repository*.

Para permitir a los componentes del *Core* la suscripción a los eventos con el objetivo de facilitar el uso del *Context Listener* al *Context Repository*, se definen los siguiente métodos:

- *Set<Listener> getObservers()*. Método encargado de devolver todos los *Listener* inscritos en el proveedor de eventos asociado al repositorio de contextos. Dichos *Listeners* serán diferentes clases del *Core* que implementen la interfaz *Listener* adecuada.

- *void onAddContextProvider(String idContextListener)*. Método que será ejecutado cuando se añada un nuevo proveedor de contextos al sistema de almacenamiento implementado por el módulo *Context Repository*.

- *void onDeleteContextProvider(String idContextListener)*. Método que será ejecutado cuando se elimine un proveedor de contextos del sistema de almacenamiento implementado por el módulo *Context Repository*.

- *void onContextProviderUpdate(String idContextListener)*. Método que será ejecutado cuando se realice alguna modificación sobre un proveedor de contextos almacenado en el módulo *Context Repository*.

- *Boolean subscribe(Listener listener)*. Este método se encargará de suscribir una nueva clase a la lista de elementos sobre los que se distribuirán los eventos producidos en el *Context Repository*.

Para permitir a los *ContextProviders* actualizar la información contextual con el fin de hacerla accesible a las diferentes *Applications* registradas en el sistema, se define la siguiente interfaz llamada *IContextUpdater*. Este será el punto de acceso al *Core* para la actualización de información.

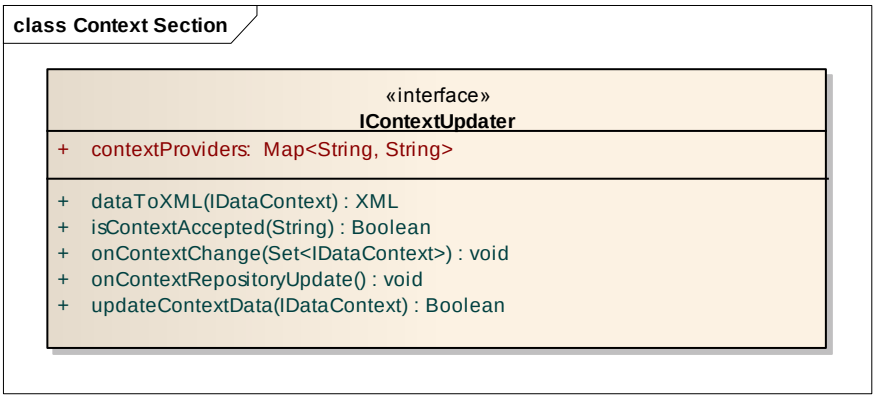


Figura 131: Diagrama de la interfaz IContextUpdater

La siguiente interfaz estará compuesta por tan sólo un atributo descrito a continuación:

- *Map<String, String> ContextProviders*. Se trata de una propiedad derivada resultante de almacenar temporalmente los identificadores del *Bundle* y el identificador del *Context Provider* asociado en la plataforma. Esto se empleará para evitar llamadas masivas al módulo *Context Repository*, incrementando de esta forma la eficiencia del sistema. Este atributo será consultado cada vez que un *Context Provider* decida actualizar la información asociada.

Con el objetivo de manejar los datos procedentes del atributo anterior y gestionar las actualizaciones realizadas por los proveedores de contexto, se definen los siguientes métodos:

- *XML dataToXml(IDataContext data)*. Este método realiza el proceso de conversión entre un elemento de tipo IDataContext, que contendrá la información contextual que se dispone a ser actualizada, a un objeto de tipo XML que contiene dicha información en formato textual. El objetivo de este método es independizar al máximo el sistema de actualización de información contextual del resto de módulos del sistema, de esta manera es posible realizar cambios en el formato del almacenamiento XML del módulo Data Repository sin excesiva complejidad.

- *Boolean isContextAccepted(String idBundle).* Este método comprobará si el *Context Provider* cuyo identificador de *Bundle* se pasa como argumento está registrado como proveedor de contexto confiable dentro del sistema. Para ello, realizará una búsqueda dentro del atributo *contextProviders* y, en caso de que se encuentre en este, el resultado será positivo. En caso contrario devolverá un valor negativo, ya que el atributo se mantiene actualizado respecto al repositorio a través del *Context Listener*.

- *Void onContextChange(Set<IDataContext> data).* A la hora de solicitar la actualización de un nuevo dato por parte del *Context Provider*, este método será el encargado de difundir entre el resto de módulos de la aplicación los datos de actualización. Dichos datos constarán del identificador del contexto, el cual determinará el contexto concreto encargado de realizar la actualización, y un conjunto de objetos de tipo *IDataContext* con el valor concreto de los datos obtenidos por el proveedor contextual. Una vez sea recibido un nuevo dato, este método se encargará de comprobar en primer lugar si el *Context Provider* identificado por su identificador de *Bundle* está incluido en el sistema como permitido llamando al método *isContextAccepted*. En caso afirmativo, la escritura será delegada en el módulo *Data Repository* a través del método *updateContextData*.

- *Void onContextRepositoryUpdate()*. Este método será llamado por el *Context Listener* si se produce alguna actualización en el repositorio de contextos. De este modo, cuando se reciba el evento del *Context Listener* se realizará la conexión con el repositorio y la actualización del atributo *contextProviders*, con el fin de seguir manteniendo la integridad del sistema.

- *Void updateContextData(IDataContext data)*. Se encarga de enviar una solicitud de actualización de datos al módulo *Data Repository* con la información contenida en el objeto *IDataContext* que se pasa por parámetro.

Para permitir el envío y recepción de información contextual por parte de los *Context Provider*, el *Core* facilita una interfaz que permite encapsular dichos datos. La interfaz en cuestión se denomina *IDataContext* y se compone por un conjunto de atributos y métodos.

Respecto a los atributos, serán necesarios para almacenar no sólo el dato contextual en sí, sino una serie de información asociada a este como por ejemplo la precisión o el proveedor de contexto que lo proporciona. Concretamente, los atributos de la clase *IDataContext* son los siguientes:

- *Integer accuracy*. Precisión del dato encapsulado en el objeto actual. Este dato es de especial interés para aquellas aplicaciones que necesitan trabajar con datos con una

determinada precisión. Será labor del propio *ContextProvider* asignar un valor adecuado a esta propiedad, siendo "-1" si la precisión no está disponible.

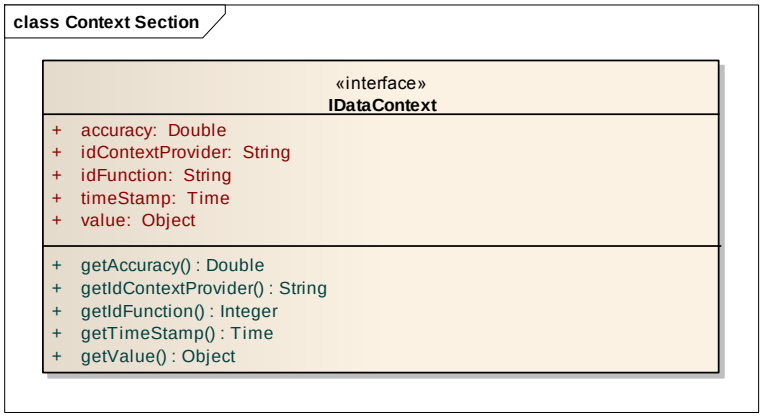


Figura 132: Diagrama de la interfaz IDataContext

- *String idContextProvider*. Un dato estará asociado a una *Function*, la cual es la que provee al Core del nuevo valor. Por otro lado, una función se identifica conociendo tanto su descriptor, el cual coincide con su atributo *name*, como el *Context Provider* al cual está asociado. Por este motivo es condición necesaria que un objeto de tipo *IDataContext* contenga ambos valores para quedar completamente determinado el origen de los datos.

- *Integer idFunction*. Este atributo contendrá el identificador de función que se encarga de generar el dato. Junto con el atributo *idContextProvider* identifica completamente a la función que provee el dato.

- *Time timeStamp*. Dado que se ha de mantener un histórico de datos dentro del módulo *Data Repository*, un objeto de tipo *IDataContext* podría tener varios valores asociados a la misma función y proveedor de contexto. Por este motivo, para que quede completamente definido el valor será necesario declarar el atributo que identifique la marca de tiempo en que se generó la información.

- *Object value*. Este atributo es el encargado de almacenar la información contextual que proporciona el *Context Provider*. En general, el valor almacenado en este atributo será el resultado de aplicar una función a un origen de datos concreto. Se trata de un objeto genérico, puesto que a priori no será posible conocer el tipo concreto de la información.

Para la gestión de los atributos descritos anteriormente, se definen los siguientes métodos que permitirán acceder y modificar sus valores:

- *String getIdContextProvider()*. Método encargado de devolver el identificador de proveedor de contexto que genera la información contextual almacenada en el objeto *IDataContext*.

- *Integer getIdFunction()*. Método consultor para el identificador de función. Devuelve un objeto de tipo Integer, que coincide con el valor del atributo 'idFunction' y, a su vez, con el identificador asociado al objeto *Function* del *Context Provider* que ha creado el valor del dato.

- *Time getTimeStamp()*. Este método será el encargado de retornar la marca de tiempo asociada a la información. Devolverá un objeto de tipo *Time*, que coincide con el valor del atributo *timeStamp* e identifica el momento en que se crea el objeto de tipo *IDataContext*, según el reloj del sistema.

- *Double getAccuracy()*. Método encargado de devolver la precisión de la medida de la información contextual almacenada en el objeto. Devuelve un objeto de tipo *Double*, que coincide con el valor del atributo *accuracy* e identifica la precisión asociada al valor del dato.

- *Object getValue()*. Es el método consultor más importante de la interfaz *IDataContext*, dado que devolverá el valor de la propia información contextual generada por la función asociada al proveedor de contexto determinado. Devuelve un objeto de tipo *Object*, el cual coincide con el valor del atributo v*alue* de la clase.

Llegados a este punto, se ha realizado una descripción en profundidad de todas las interfaces que componen el *Context Section* de la capa intermedia del middleware desarrollado. A modo de resumen, se podría decir que la sección descrita ofrece a las *Applications* y a los *ContextProviders* todos los puntos de acceso necesarios al *Core* para la suscripción a la plataforma, así como para labores de búsqueda y actualización de información contextual.

Esta sección será la base para los siguientes conjuntos de módulos que se describirán a continuación, los cuales permitirán a la aplicación trabajar con toda la información generada por la capa inferior de la arquitectura, es decir, la capa de *ContextProviders*.

Data Section

La principal funcionalidad de esta sección es almacenar los datos provenientes de los *ContextProviders*, así como posibilitar a las *Applications* la consulta de dichos datos. El *Data Section* es considerablemente más sencillo que la sección explicada anteriormente, dado que partirá de la precondición que tanto *ContextProviders* como *Applications* ya se encuentras suscritos en la plataforma de gestión de contexto.

Comenzaremos describiendo la interfaz en torno a la cual se organiza el *Data Section*, puesto que el objeto que la implemente será el encargado de almacenar toda la información contextual procedente de los proveedores de contexto, con el fin de que las diferentes *Applications* puedan llevar a cabo las consultas necesarias.

La interfaz *IDataRepository*, como se comentó anteriormente en la sección de diseño, posee dos métodos de almacenamiento de datos con el fin de facilitar el uso de estos a las aplicaciones y a los *ContextProviders*. Por un lado, la información será almacenada de manera permanente en formato *XML*, a partir del cual se aumenta la capacidad del sistema para la realización de consultas sobre la propia información. Por otro lado y basado en el propio fichero *XML* de almacenamiento, la información podrá ser accesible mediante objetos de tipo *IDataContext*, el cual contendrá todas las propiedades relativas a un dato contextual procedente de un *ContextProvider*.

Para conocer más profundamente la manera en que el módulo *IDataRepository* almacena la información, en el *Anexo II* se detalla la estructura del fichero *XML* donde se almacenan los datos de contexto.

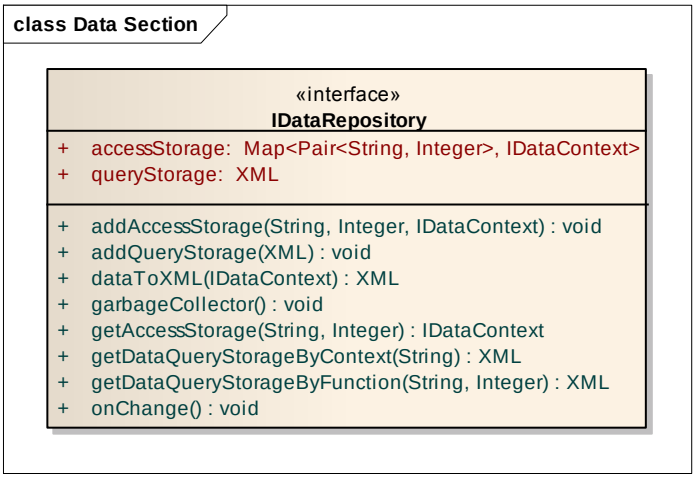


Figura 133: Diagrama de la interfaz IDataRepository

Una vez descrito el objetivo de la interfaz *IDataRepository*, procederemos a describir los diferentes atributos que la componen:

- *Map<Pair<String,Integer>, IDataContext> accessStorage*. Este atributo será el encargado de almacenar, asociado a un par de identificadores consistente el identificador del *Context Provider* y la *Function* (*idContextProvider, idFunction*) de la función que proporciona la información contextual, el dato generado por el proveedor de contexto. El objetivo del atributo *accessStorage* es conseguir una eficiencia elevada a la hora de consultar los últimos valores asociados a una función y proveedor de contexto determinado.

- *XML queryStorage*. El atributo actual permite almacenar la información contextual en el formato descrito en el *Anexo II* en el fichero *XML* creado para tal fin. En principio, esta información no es accesible tal cual para las *Applications*, sino que las diferentes consultas que se deseen realizar sobre los datos, deberán llevarse a cabo a través del módulo *DataQuery*.

Tras describir los atributos de los que consta la interfaz *IDataRespository*, se procederá a describir los diferentes métodos que permiten el acceso a estos atributos:

- *Void addAccessStorage(String idContext, Integer idFunction, IDataContext data)*. Este método añadirá una nueva entrada al repositorio de información asociada a los identificadores de contexto y de función que se pasa por parámetro.

- *Void addQueryStorage(XML data)*. Adicionalmente al método anterior, es posible agregar un nuevo valor contextual a partir del *XML* asociado. El objetivo de este medio de inserción es garantizar la eficiencia en aquellos momentos en los que no sea necesario realizar un acceso al objeto *IDataContext* de información.

- *XML dataToXml(IDataContext data)*. A través de este método se posibilita la conversión de un objeto de tipo *IDataContext* a su *XML* asociado. Esta funcionalidad puede ser empleada por las *Applications* para llevar a cabo consultas personalizadas sobre los datos contextuales procedentes del *Core*.

- *Void garbageCollector()*. El recolector de basura permitirá eliminar del repositorio de datos aquellos elementos que tengan una antigüedad excesiva en el sistema. El objetivo de esta eliminación es el control sobre el espacio de ocupación del repositorio de contexto, ya que no hay que perder de vista que el trabajo con dispositivos móviles implica el hecho de controlar el uso del soporte de almacenamiento. Este método será llamado de manera automática por el *Core* cuando el tamaño de los datos sobrepasen un determinado valor que dependerá del propio sistema donde se ejecute el middleware.

- *IDataContext getAccessStorage(String idContextProvider, Integer idFunction)*. Este método permite obtener el último dato proporcionado por la función asociado al *ContextProvider* cuyos identificadores se pasan por parámetro. Este método podría interpretarse como un método de consulta básico sobre la información contextual.

- *XML getDataQueryStorageByFunction(String idContextProvider, Integer idFunction)*. El método actual permitirá obtener toda la información contextual almacenada relativa al identificador de contexto y función que se pasan por parámetro.

- *XML getDataQueryStorageByContext(String idContextProvider)*. Este método es semejante al anterior, salvo que en este caso se devolverá toda la información contextual almacenada relativa al *ContextProvider* cuyo identificador se pasa por parámetro.

- *Void onChange()*. Este método permitirá enviar un evento al resto de módulos del sistema con el fin de indicar que se ha producido un cambio en el *Context Repository*.

En relación al último método de la interfaz *IDataRepository*, para dar la posibilidad al resto de módulos de realizar una suscripción a los eventos generados por el repositorio de datos, se define la interfaz *IDataListener*. El objetivo de esta interfaz es interceptar la llamada al método *onChange* del *Data Repository* y realizar la llamada a los métodos activadores del evento de cada una de las clases suscritas al mismo.

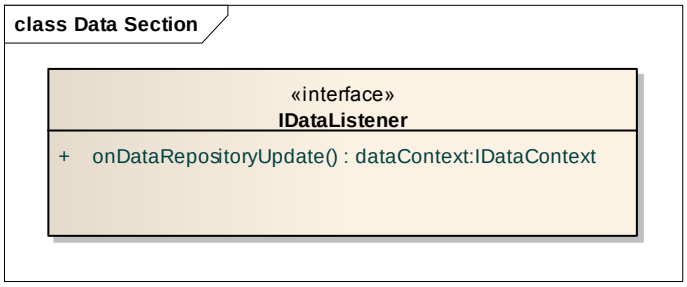


Figura 134: Diagrama de la interfaz IDataListener

La interfaz *IDataListener* se compone únicamente de un método denominado *onDataRepository* que devolverá un objeto *IDataContext* con el dato del repositorio modificado. El principal objetivo de este método es notificar al módulo *Aggregation Repository* al producirse una modificación en el

repositorio de datos, lo cual permitirá a este módulo reevaluar aquellos agregados involucrados en la actualización.

Como se comentó anteriormente, el módulo *Data Repository* no permite realizar consultas por parte de las *Applications*, por lo que esta funcionalidad es labor de la interfaz *IDataQuery*. Esta interfaz poseerá todos los métodos necesarios para realizar las consultas oportunas sobre la información contextual proporcionada por los diferentes *ContextProviders*. De este modo, la interfaz *IDataQuery* estará compuesta por dos tipos de métodos de consultas, los encargados de generar un objeto *XML* con el resultado de la consulta y aquellos que generan elementos de tipo *IDataContext*.

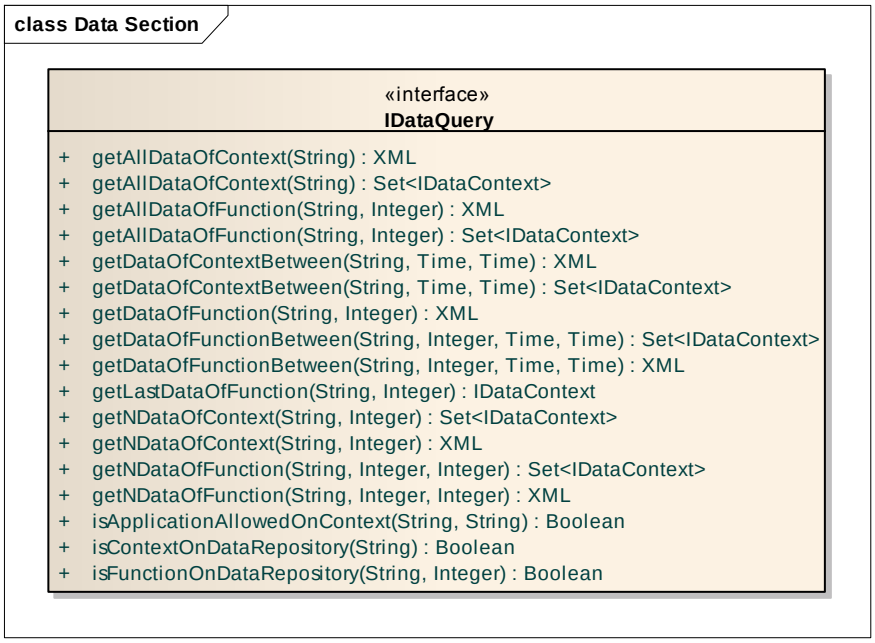


Figura 135: Diagrama de la interfaz IDataQuery

Debido al elevado número de métodos y a lo descriptivo de los mismos, se describirá tan sólo aquellos más empleados por las *Applications* implementadas para las pruebas del middleware desarrollado:

- *Set<IDataContext> getAllDataOfContext(String idContextProvider)*. Método encargado de generar un conjunto de objetos de tipo *IDataContext* con aquellos elementos de información contextual asociados al *ContextProvider* cuyo identificador se pasa por parámetro. Recordemos que en este caso se podrían devolver elementos duplicados para la misma función, ya que el repositorio almacena un histórico con los valores de la información contextual.

- *Set<IDataContext> getDataOfContextBetween(String idContextProvider, Time startTime, Time endTime)*. En este caso, la funcionalidad del método es semejante al anterior, salvo que el conjunto de elementos devueltos serán aquellos cuya generación se haya producido entre la fecha y hora inicial y final que se pasan por parámetro.

- *Set<IDataContext> getDataOfFunction(String idContextProvider, Integer idFunction)*. Este método devolverá un conjunto de *IDataContext* que contiene aquella información

provista por la *Function* cuyo identificador de contexto y de función se pasan por parámetro.

Con este elemento finalizamos la descripción de interfaces de las que se compone el *Data Section*, a través de las cuales se permitirá el almacenamiento y la consulta de información contextual por parte de los *ContextProviders* y las *Applications* respectivamente.

Event Section

La ventaja más importante del uso del middleware propuesto es evitar que las aplicaciones deban consultar de manera reiterada una determinada información contextual en busca de que se dé un valor concreto, o en general, que se produzca una condición concreta en dicha información. Gracias a la sección de eventos es posible inscribir una determinada *Application* en un evento o agregado, de manera que será el propio *Core* el encargado de evaluar las diferentes expresiones en busca de un valor concreto, el cual será informado al *Core* mediante la aplicación.

Esta inscripción se realizará a través de módulo *Event Subscription* como se comentó en la sección de diseño. De este modo, comenzaremos esta sección describiendo la interfaz que representa a este módulo y que permitirá la comunicación entre las *Applications* y el *Core* a nivel de suscripción de eventos. La interfaz encargada de especificar la funcionalidad anterior recibe el nombre de *IEventSubscription.*

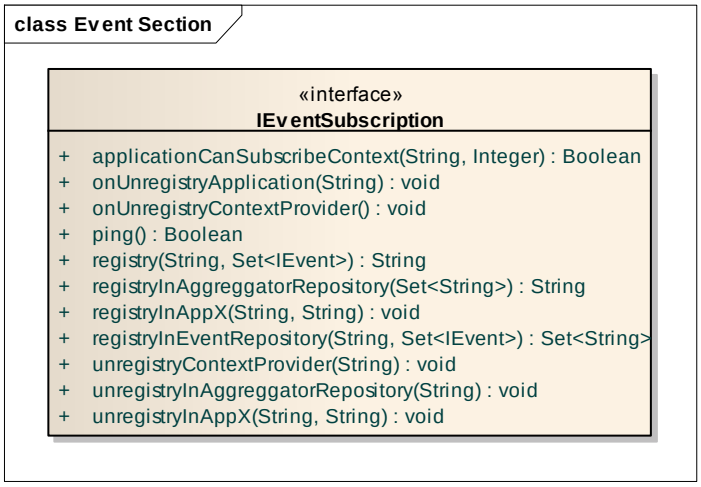


Figura 136: Diagrama de la interfaz IEventSubscription

Los métodos que permitirán la suscripción a los distintos eventos auto-configurados por la propia aplicación son los siguientes:

- *Boolean applicationCanSubscribeContext(String idContext, String idApplication)*. El Core implementa un sistema de seguridad interno basado en un módulo de políticas de seguridad, de tal manera que una aplicación solamente puede tener acceso a aquella información relacionada con los diferentes ContextProviders a los que se haya suscrito correctamente a través del módulo Context Discover. Por lo tanto, cualquier petición de acceso a la información almacenada en el sistema, lo cual incluye la suscripción a agregados, debe tener en cuenta que se haya realizado previamente el proceso de suscripción. Este método será el encargado de realizar la comprobación de adscripción de la aplicación a un Context Provider, a partir de

una consulta al módulo Application Repository.

- *Boolean onUnregistryApplication()*. Del mismo modo que una aplicación puede suscribirse a un agregado o a un evento simple, esta suscripción también puede ser eliminada en el momento en que se desee cambiar dicha suscripción o simplemente dejar de estar suscrito al sistema. Este método puede ser usado del mismo modo por el Core, en el caso en el que sea detectada una "caída" de la Application asociada al evento.

- *String registry(String idApplication, Set<IEvent> events)*. Cuando una aplicación desea suscribirse a un conjunto de eventos, debe hacerlo a través de un punto de entrada al módulo Event Subscription definido precisamente por la función registry. Dicha función, gracias a los parámetros recibidos, permite asociar un conjunto de elementos de tipo IEvent que definen los eventos a los que se desea suscribir la aplicación, a la aplicación que realiza la petición. Dicho conjunto de eventos será interpretado como un agregado de condiciones AND, es decir, el evento se activará cuando se evalúen positivamente todos y cada uno de los eventos que componen el conjunto.
  El parámetro devuelto por este método es una cadena que contiene el identificador del agregado que se ha dado de alta en el Core. Dicho agregado será empleado por la aplicación para identificar el evento que se ha activado en el caso de estar suscrita a más de uno.

Existe otros métodos en la interfaz *IEventSubscription* pero debido a que su uso se realiza de manera interna al *Core*, carece de sentido realizar una análisis profundo de los mismo. En general, el resto de métodos serán lanzados desde los métodos *registry* y *onUnregistryApplication* como funciones de apoyo a los mismos.

La manera en que las aplicaciones configuran los diferentes eventos de los que desean recibir notificaciones pasa por el uso de la interfaz *IEvent*, la cual deberá ser comprendida correctamente para configurar los distintos eventos de manera adecuada.

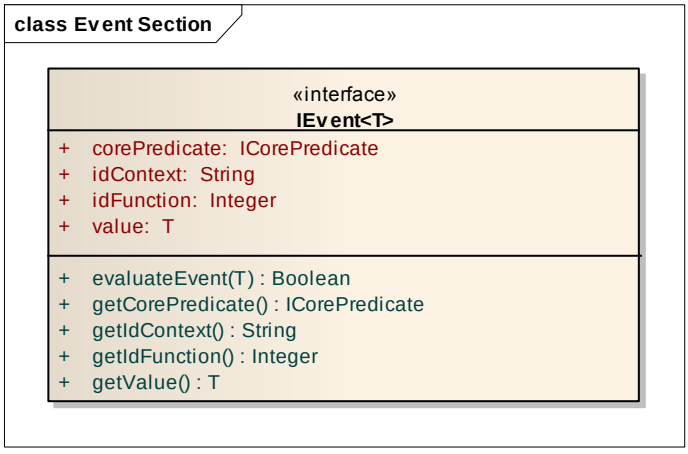


Figura 137: Diagrama de la interfaz IEvent<T>

El objetivo de la interfaz *IEvent* es envolver un determinado evento y todos los parámetros que lo configuran. De este modo, los elementos necesarios para configurar dicho evento son los siguientes:

- *String idContext*. Hace referencia al identificador del *Context Provider* que proporciona el valor de referencia asociado al evento.

- *Integer idFunction*. Atributo que identifica de manera unívoca junto al anterior el origen de los datos. Recordemos que una información contextual determinada se identifica de manera unívoca conociendo los identificadores de contexto y función que la proporcionan.

- *T value*. Valor de referencia con el que se comparará el valor obtenido por la función identificada anteriormente para determinar si se debe activar el evento configurado.

- *ICorePredicate corePredicate*. La relación entre valor de la función y valor de referencia no tiene por qué ser siempre la igualdad. Podremos comparar si el valor devuelto es mayor, menor, mayor o igual, etc. que el valor de referencia. Este conjunto de comparadores será proporcionado por los objetos de la interfaz *ICorePredicate*, la cual permite aplicar un determinado operador a un conjunto de datos.

Asociado a los atributos anteriores se definen una serie de métodos que permiten consultar sus valores. De esta forma, no existen métodos con funcionalidad adicional, ya que el único fin de esta interfaz es encapsular los valores de configuración del evento.

Sin embargo, la interfaz *IEvent<T>* debe proporcionar un método que permita al *Core* realizar la evaluación de un determinado evento a partir del valor de referencia. Este método recibe el nombre de *evaluateEvent*. El resultado del mismo será un valor lógico que indica si el valor de referencia pasado por parámetro cumple la relación indicada por el atributo *corePredicate* y el valor *value*. En caso de evaluarse a cierto indicará que la condición definida por el evento simple se ha satisfecho.

Como se ha explicado anteriormente, la relación entre el valor de referencia y el valor real de la información contextual queda identificada a través de un objeto de tipo *ICorePredicate<T>*. Esta interfaz permite ampliar la funcionalidad del *Core* mediante la inclusión de nuevas relaciones entre valores con tan sólo crear una nueva instancia de dicha interfaz que relacione los valores de la manera deseada.

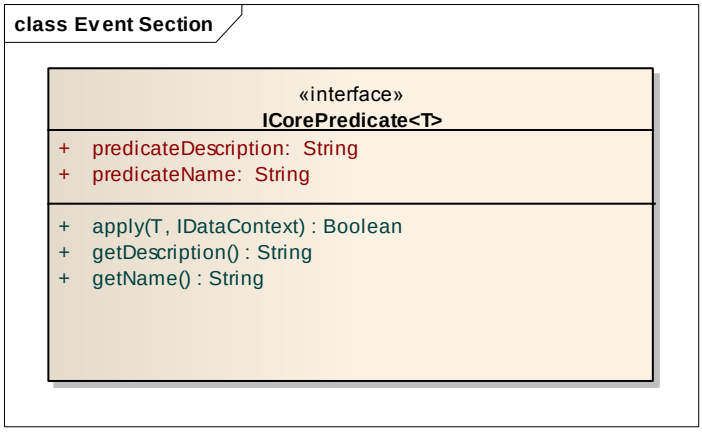


Figura 138: Diagrama de la interfaz ICorePredicate

Para poder hacer referencia a los diferentes predicados, se necesita que el usuario defina a cada uno de ellos un nombre y una descripción, con el fin de reutilizar al máximo dichas relaciones. El

método más relevante de esta interfaz es el denominado *apply*. Este método recibirá por parámetro un objeto de tipo *T* genérico que identificará el tipo del objeto de referencia, y un objeto de tipo *IDataContext* que contendrá el valor real de la información contextual proporcionada por el *Core* a través del *Context Provider*.

Internamente, lo diferentes eventos serán referenciados a través de un identificador que los representa. Dicho identificador debe ser único y por otro lado, el *Core* deberá poseer un elemento que permita almacenar la relación entre los identificadores y los objetos *IEvent* al que representan. Dicha estructura está incluida en el módulo *Event Repository* y poseerá un objeto que permita el almacenamiento de esta información. El objeto de almacenamiento deberá implementar la interfaz *IEventRepository* que se describe a continuación.

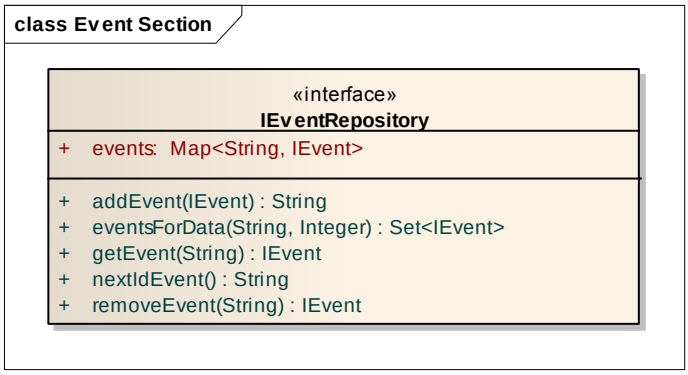


Figura 139: Diagrama de la interfaz IEventRepository

La principal propiedad de la interfaz *IEventRepository* es el mapa que permite almacenar los pares clave-valor que representan al identificador del evento y al propio evento respectivamente. A través de los siguientes métodos, el *Core* podrá gestionar los diferentes identificadores de eventos de forma que será posible añadir nuevos eventos al repositorio, eliminarlos o consultarlos.

- *String nextIdEvent()*. Este método devolverá el siguiente identificador disponible, el cual será auto-generado por el objeto descrito asegurando que es único para todos los eventos registrados en la plataforma.

- *IEvent getEvent(String idEvent)*. A partir del identificador es posible obtener el objeto *IEvent* asociado. Esto será útil a la hora de evaluar los eventos de los que se compone un determinado agregado.

- *String addEvent(IEvent event)*. Este método permitirá almacenar los eventos en el repositorio. Previo a la adición del evento, este método realizará una búsqueda en el repositorio con el fin de obtener un evento idéntico al que se desea añadir. La relación de igualdad entre eventos se basará en las propiedades *idContextProvider, idFunction, data y corePredicate*. En caso de que sea hallado un elemento con dichas características, el método actual devolverá el identificador asociado a dicho evento, en caso contrario se añadirá el nuevo evento al repositorio tras generar un identificador unívoco.

- *IEvent removeEvent(String idEvent)*. Este método eliminará del repositorio el evento cuyo identificador se pasa por parámetro. El método *removeEvent* tan sólo deberá ser llamado por el *Core* si no existe ninguna aplicación que se encuentre usando el evento a eliminar.

- *Set<IEvent> eventsForData(String idContext, Integer idFunction)*. Cuando se produce un proceso de evaluación de agregados (véase *Context Aggregator*) el módulo *Context Aggregator* accede al repositorio *Event Repository* para recuperar todos los eventos asociados a una determinada función correspondiente a un contexto. Esta funcionalidad se provee a través del método *eventsForData*. Este método se encarga de recorrer el repositorio buscando todos aquellos registros cuyo valor (objeto IEvent) está generado por el identificador de *Context Provider* y de *Function* pasados por parámetro.

Una vez descrita la interfaz que brinda la posibilidad al *Core* de almacenar los diferentes eventos generados por las aplicaciones, expondremos otra interfaz que permitirá en este caso enviar una serie de notificaciones a las aplicaciones con objeto de informar que se ha evaluado positivamente un determinado evento.

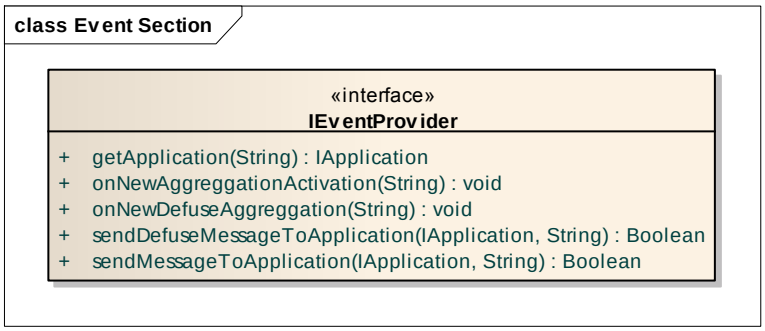


Figura 140: Diagrama de la interfaz IEventProvider

Esta interfaz recibirá el nombre de *IEventProvider* y consta de los siguiente métodos que permitirán al *Core* enviar eventos hacia las *Applications* que así lo requieran:

- *Void onNewAggregationActivation(String idAggregation).* El módulo *Event Provider* debe responder al mensaje de activación de agregado del módulo *Context Aggregator*, de manera que tan sólo se ejecutará cuando este lo solicite.
  Por otro lado, con el objetivo de determinar los eventos que deben ser lanzados se hace uso de los métodos de acceso a los repositorios *AppX* y *Application Repository*, esto permite conocer las aplicaciones a las que debe ser enviado el evento de alerta.
  Todo este procesamiento será realizado por la función *onNewAggreggationActivation*, la cual actúa como punto de entrada al módulo por parte del *Context Aggregator*.

- *Application getApplication*(*String idAggregation*). Una vez el módulo obtiene el listado de identificadores de aplicación a las cuales debe enviar un mensaje, para poder enviar un mensaje a la aplicación correspondiente es necesario conocer la referencia al objeto *IApplication*, para poder ejecutar su método de aviso. Para ello se diseña el método *getApplication*, el cual se encarga de recuperar el objeto asociado al identificador de aplicación que se pasa como parámetro de entrada, utilizando el método *getApplicationById* del módulo *Application Repository*.

- *Boolean sendMessageToApplication(IApplication application, String idAggregation).* El módulo *Event Provider* se encarga de alertar a las aplicaciones cuando se ha producido un nuevo agregado, de manera que ejecuta el método 'onNewEvent' de la interfaz *IApplication* cuando se evalúe un agregado como cierto, enviando a este el identificador de agregados activado.

Para ello es necesario diseñar un método que se encargue de enviar dichos mensajes a las aplicaciones. El método *sendMessageToApplication* implementa esta funcionalidad, enviando un mensaje de activación a la aplicación que se le pasa como parámetro de entrada, indicando que se ha activado el agregado cuyo identificador también se pasa como parámetro.

- *Boolean onNewDefuseAggreggation(String idAggregation)*. Paralelamente al procesamiento de detección de contextos realizado en el método *onNewAggregationActivation*, se realiza el de desactivación de agregados. Todo este procesamiento se realiza en el módulo *Context Aggreggator*, aunque cuando este evalúa un agregado como "desactivado" (lo cual indica que antes del nuevo valor contextual ek evento se encontraba activo), se encarga de enviar un aviso al módulo *Event Provider* a través del método *onNewDefuseAggregation* con el identificador de agregado correspondiente.

- *Boolean sendDefuseMessageToApplication(IApplication application, String idAggregation)*. Cuando el módulo *Context Aggregator* detecta un cambio de estado de un agregado el cual cambia de una evaluación cierta a otra falsa envía un aviso al módulo *Event Provider* el cual se encarga de gestionar el envío del mensaje a las aplicaciones correspondientes.

Se ha reservado para el último lugar la interfaz que permitirá al *Core* registrar los diferentes agregados a los que se encuentran suscritos las *Applications*. Para ello será necesario gestionar los pares agregado-aplicación con el fin de determinar, cuando se evalúe positivamente la condición de un agregado, qué aplicaciones se encontraban suscritas al mismo. Un vez obtenidas estas aplicaciones, el *Core* deberá enviar los eventos correspondientes a las mismas a través del objeto encargado de implementar la interfaz *IEventProvider* descrita anteriormente.

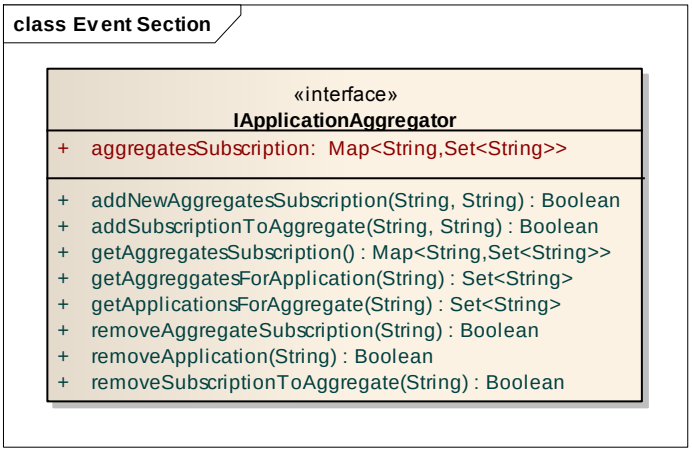


Figura 141: Diagrama de la interfaz IApplicationAggregator

Esta interfaz cuenta tan sólo con un atributo, el cual será un conjunto de pares clave-valor, donde las claves serán los identificadores de agregado y los valores serán conjuntos que contendrán los identificadores de las aplicaciones asociadas al agregado correspondiente. Gracias a este atributo denominado *aggregatesSubscription*, el *Core* conocerá en todo momento las aplicaciones suscritas a los diferentes agregados del sistema.

En cuanto a los métodos registrados en esta interfaz, cubrirán los requisitos impuestos por el repositorio entre los que se encuentran la inserción, eliminación y consulta de contextos asociados a aplicaciones y viceversa. Concretamente, entre los métodos definidos se encuentran los siguientes.

- *Set<String> getApplicationsForAggregate(String idAggregate)*. Devuelve un conjunto de identificadores de aplicaciones asociadas al agregado cuyo identificador se pasa por parámetro. Este método es útil a la hora de consultar las aplicaciones asociadas a un evento a la hora de determinar a cuales de dichas se deberá notificar la activación del mismo.

- *Set<String> getAggreggatesForApplication(String idApplication)*. La lógica de este método es semejante al anterior, salvo que en este caso el parámetro de consulta es el identificador de una aplicación. Por tanto, el resultado de la llamada a este método será el conjunto de identificadores de agregados asociados a una determinada aplicación.

- *Boolean addNewAggregatesSubscription(String idAggregate, String idApplication)*. Este método añadirá un nuevo par clave valor que asocie el agregado y la aplicación cuyos identificadores se pasan por parámetro. El agregado al cual se quiere añadir el identificador de aplicación puede haber sido añadido previamente dentro del repositorio, por lo que en este caso no será necesario añadir el agregado, sino simplemente la referencia a la aplicación. En caso de que el agregado no haya sido introducido aun en el repositorio, este método se encargará de llamar al *addNewAggreggatesSubscription*, el cual añade un nuevo agregado al sistema.

Aggregation Section

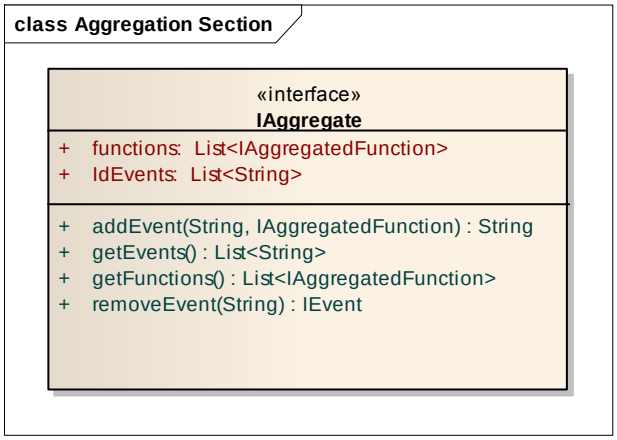
Mediante el middleware desarrollado es posible crear eventos complejos, los cuales consisten en la composición de varios eventos simples empleando funciones lógicas para su unión. En este caso, las interfaces del *Aggregation Section* permitirán al *Core* almacenar los diferentes agregados creados por las *Applications* así como encapsular la composición de un conjunto de eventos simples en un agregado.

Comenzaremos definiendo la interfaz *IAggregate*, la cual permite almacenar la relación existente entre varios eventos simples para encapsular a un agregado. Para este fin, los objetos que implementen dicha interfaz necesitarán almacenar, por un lado la lista de eventos simples que compondrán el agregado y por otro, la lista de funciones que relacionarán a los distintos eventos. Para ello se definen dos atributos:

- *List<String> idEvents*. Almacena los identificadores de los eventos implicados en la agregación. De esta manera, cuando se desea evaluar un determinado agregado, el *Core* enviará la lista con todos los eventos que componen el agregado a evaluar al *DataRepository* y devolverá una lista de valores booleanos indicando cuáles de estos eventos simples se cumplen y cuáles no. Una vez realizado esto, será el propio *IAggregate* el encargado de determinar si se cumplen las relaciones lógicas definidas por los *IAggregatedFunctions* que la aplicación configuró.

- *List<IAggreggatedFunction> functions*. Almacena la relación entre los eventos recogidos en el anterior atributo. Con el fin de permitir ampliar la funcionalidad del middleware, estas relaciones se basan en la implementación de la interfaz *IAggregatedFunction*. Así,

una función de agregado permitirá unir dos eventos simples consecutivos, siendo la longitud de la lista *Functions* igual a la longitud del atributo *idEvents* menos uno. Por este motivo, la evaluación del agregado se realizará mediante la evaluación de los pares de eventos a los que unen las funciones de manera recursiva.



- *Figura 142: Diagrama de la interfaz IAggregate*

Una vez definidos los atributos que conforman la interfaz *IAggregate*, describiremos el conjunto de funciones que permiten acceder y gestionar los atributos anteriores.

- *String addEvent(String idEvent, IAggregateFunction function)*. Este método es el encargado de añadir un evento a la lista de eventos presentes anteriormente en el agregado. Para añadir un evento a dicho conjunto se necesitará tanto el identificador del evento que previamente se encuentra añadido en el *EventRepository* y la función que relacionará el último evento añadido con el anterior. En caso de que se trate del primer evento que se añada al agregado, esta función no tendrá validez alguna y no será almacenada en el atributo *functions*. En cambio, para el segundo y sucesivos eventos, la función definirá la relación entre el último evento y el anterior. El resultado de este método es el identificador del agregado generado.

- *List<String> getEvents()*. Este método devuelve la lista de eventos que forman el agregado de manera ordenada. El orden de dichos eventos será la secuencia de inserción que siguió la aplicación a la hora de crearlo.

- *List<IAggregateFunction> getFunctions()*. En este caso, la función *getFunctions* devolverá las funciones que relacionan los eventos simples añadidos al agregado.

- *IEvent removeEvent(String idEvent)*. Elimina el evento cuyo identificador se pasa por parámetro del agregado actual. El resultado del método será el objeto *IEvent* que define al evento eliminado.

Además, es necesario definir una estructura que permita almacenar los diferentes agregados con los que trabaja el middleware. Para ello se define la interfaz *IAggregateRepository* la cual, a través de un atributo que contendrá el conjunto de agregados del sistema, ofrecerá una serie de métodos encargados de gestionar dicho repositorio.

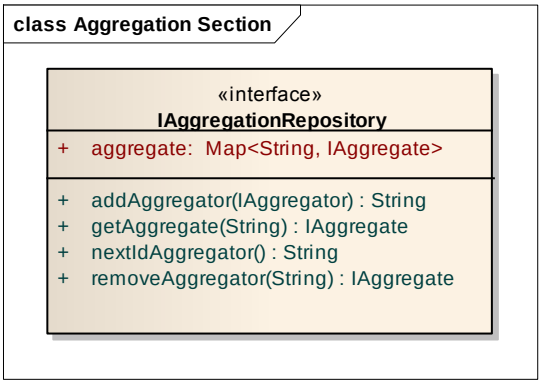


Figura 143: Diagrama de la interfaz IAggregationRespository

Los objetos que implementen la interfaz *IAggregationRepository* tan sólo tendrán que disponer necesariamente de un atributo, el cual se describe a continuación.

- *Map<String, IAggreggate> aggreggates*. El módulo *Aggreggation Repository* se diseña para ofrecer al *Aggregation Section* un sistema de almacenamiento que debe ofrecer al sistema un acceso rápido y sencillo a los diferentes agregados generados y sus identificadores, para lo que resulta básico un atributo que permita almacenar pares clave-valor con el identificador y el propio agregado. De esta función se encargará el atributo *aggreggates*, con el fin de que, cuando el módulo *Context Aggregator* acceda al repositorio para obtener un agregado, podrá obtener en un tiempo constante el agregado a partir del identificador, el cual representa de manera unívoca al agregado en el sistema.

Para gestionar los diferentes agregados se definen una serie de métodos descritos a continuación.

- *String addAggregate(IAggregate aggregate)*. Este método permite añadir un agregado al conjunto de agregados manejados por el sistema. Dado que el agregado en sí no posee un identificador, será este método el encargado de informar a la clase que lo invoca del identificador generado al añadir el agregado al repositorio. Este identificador será el objeto devuelto por el método actual.

- *String nextIdAggregate()*. Con el fin de reducir y centralizar la lógica de asignación de identificadores a los agregados, este método permite obtener el siguiente identificador de agregado disponible para el sistema. Este método será llamado por *addAggregate* para generar el identificador del agregado.

- *IAggregate getAggregate(String idAggregate)*. Puesto que el sistema tan sólo manejará el agregado a través de su identificador, este método permite obtener el objeto *IAggregate* asociado al identificador que se pasa por parámetro.

- *IAggregate removeAggregate(String idAggregate)*. Cuando una determinada aplicación elimina la suscripción a un agregado de eventos, este método es invocado para remover dicho agregado de la lista de elementos del sistema. El objeto devuelto por este método será el agregado que se ha eliminado a partir de su identificador.

Para llevar a cabo la evaluación de agregados es necesario llevar a cabo un nuevo elemento en la estructura del *Aggregation Section*. En este caso, la interfaz *IContextAggregator* permitirá gestionar en

todo momento el disparo de eventos cuando se evalúe satisfactoriamente las condiciones que conforman un determinado agregado. Además, no sólo es interesante para las *Applications* conocer cuando se activa un agregado, sino cuando se desactiva. Un ejemplo para este caso se obtiene en la ejecución de una determinada aplicación basada en la detección de actividades. Si, por ejemplo, el usuario desea informar a una aplicación encargada de reproducir un determinado 'playlist' el momento en el que cumple una determinada condición, en este caso que la actividad detectada para el usuario sea *correr*. Mientras se cumpla esta condición, la aplicación mantendrá activa la lista de reproducción actual, hasta el momento en el que cambie la actividad llevada a cabo por el usuario, que será detenido el reproductor.

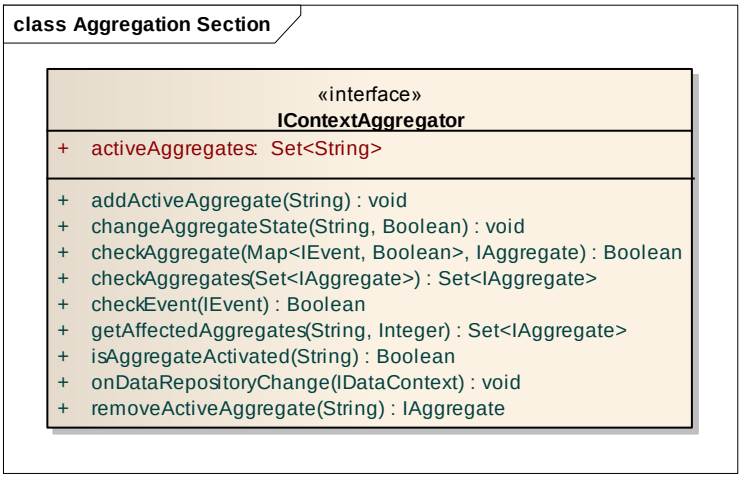


Figura 144: Diagrama de la interfaz IContextAggregator

En consecuencia es necesario almacenar los agregados activos en cada momento, de manera que cuando el sistema evalúe satisfactoriamente un agregado que no está presente en la lista o bien se evalúa negativamente un agregado que estaba presente, alertar a la aplicación correspondiente que ha sido activado o desactivado respectivamente el agregado.

Para ello se presenta el atributo *activeAggregates*, el cual se utiliza para el almacenamiento del conjunto de los últimos agregados activos, pudiendo el *Core*, en base a este atributo, determinar qué agregados han cambiado de estado.

Con el fin de gestionar adecuadamente la evaluación de agregados, se definen los siguientes métodos.

- *Void addActiveAggregate(String idAggregate)*. Añade a la lista de agregados activados el agregado cuyo identificador se pasa por parámetro. Además de añadir el agregado al conjunto de activos, se llamará además al evento *onNewEvent* de las aplicaciones suscritas a este evento.

- *Void changeAggregateState(String idAggregate, Boolean state)*. Cambia el estado del agregado cuyo identificador se pasa por parámetro. En caso de que el agregado estuviera en la lista de activos, se eliminará de ésta llamando a los eventos correspondientes, mientras que si el agregado no estaba activo, se añadirá al conjunto.

- *Boolean checkAggregate(Map<String, Boolean> events, IAggregate aggregate)*. En esta función recae el peso de la evaluación de agregados. Gracias a ella es posible, a partir de un *Map* que contiene los eventos del sistema relacionados con el agregado junto a su evaluación (positiva o negativa), determinar si las condiciones de composición contenidas en el agregado se satisfacen o no en función de los valores que toman los diferentes eventos.

- *Set<IAggregate> checkAggregates(Set<IAggregate> aggregates)*. Esta función es semejante a la anterior, salvo que en lugar de recibir un único agregado para su evaluación, recibe un conjunto de ellos. Además, en este caso no se reciben las evaluaciones de los eventos, sino que será necesario acudir al *Event Repository* para realizar dicha evaluación. En general, la implementación de este método llevará asociada una serie de llamadas al método *checkAggregate*, concretamente una por cada agregado contenido en el conjunto de entrada. El resultado de esta función será el conjunto de agregados que se satisfacen en base a los valores actuales de los datos.

- *Set<IAggregate> getAffectedAggregates(String idContext, Integer idFunction)*. El cambio de cierta información contextual llevará asociado una serie de reevaluaciones de los diferentes agregados que forman el sistema. De este modo, cuando cambie esta información contextual provista por un determinado *Context Provider*, será necesario determinar qué agregados pueden verse afectados por este cambio. Para ello, la función *getAffectedAggregates* se encarga de obtener el conjunto de agregados para los cuales, alguno de sus eventos dependen de la información proporcionada por la función y el contexto cuyos identificadores se pasan por parámetro.

- *Boolean isAggregateActivated(String idAggregate)*. Determina si un determinado agregado está o no activo en el momento de la llamada.

- *Void onDataRepositoryChange(IDataContext dataContext)*. Este método será llamado por el *Data Listener* cuando se produzca alguna modificación sobre la información contextual de alguna función. Este será el punto de partida para la reevaluación de los agregados relacionados con esta información, así como para el disparo de los eventos necesarios en caso de que se satisfagan las restricciones de dichos agregados.

- *IAggregate removeActiveAggregate(String idAggregate)*. El método actual es el encargado de desactivar un agregado en caso de que estuviera evaluado satisfactoriamente en el momento en que se llama. Para ello recibirá por parámetro el identificador del agregado que se desea eliminar del conjunto de agregados activos. El resultado será el agregado que se ha eliminado y, en caso de no haberse producido ninguna eliminación, el método devolverá un valor nulo.

Con la anterior, se ha realizado una descripción de todas las interfaces que forman parte del *Core* del middleware desarrollado. Debemos denotar que no todas las interfaces serán accesibles desde las aplicaciones o desde los proveedores de contexto, sino simplemente se trata de interfaces internas del *Core* que favorecen la comunicación entre los distintos módulos que lo componen. El hecho de haber optado por una descripción a nivel de interfaz, con la leve modificación de denotar los atributos básicos involucrados, se debe a que aporta una idea lo suficientemente específica, sin tratar la implementación concreta, de las diferentes estructuras y tipos de datos que se emplean para llevar a cabo el núcleo del middleware. Por otro lado, muchas de las interfaces proporcionadas brindan la

posibilidad de convertir el middleware desarrollado en un elemento expandible y en continua evolución, puesto que se deja abierta la posibilidad de generar nuevos tipos concretos que implementen las interfaces descritas, adaptando así el middleware a las necesidades que se produzcan.

A continuación, a modo de resumen, se presentan las diferentes interfaces que han sido diseñadas para el núcleo del middleware. Dichas interfaces se agrupan en los diferentes módulos que conforman el *Core* del middleware.

Figura 145: Conjunto de interfaces correspondientes al Context Section del Core del middleware desarrollado

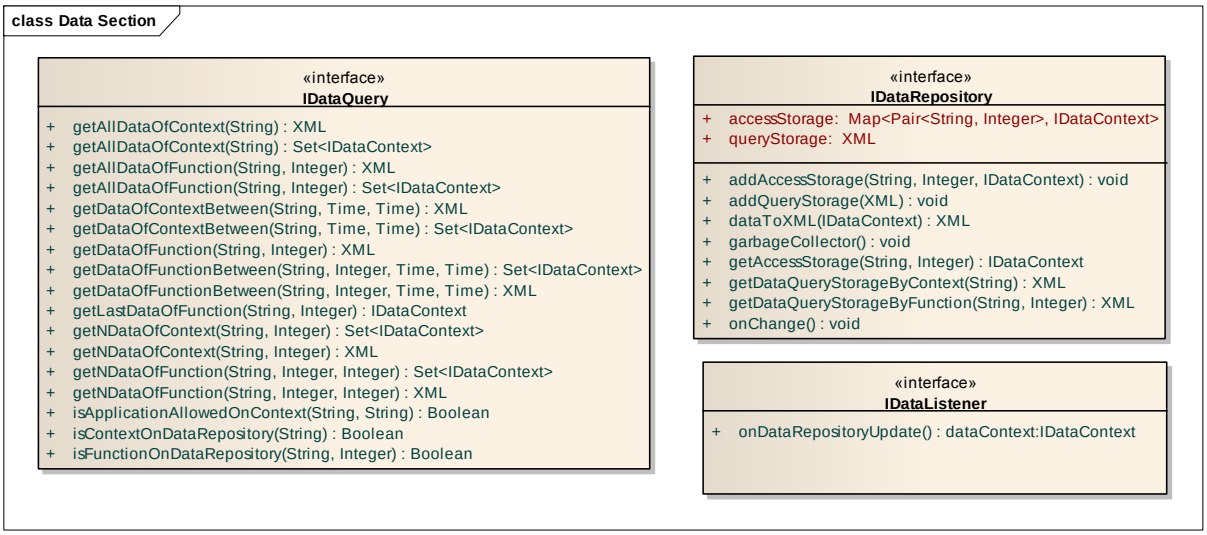


Figura 146: Conjunto de interfaces correspondientes al Data Section del Core del middleware desarrollado

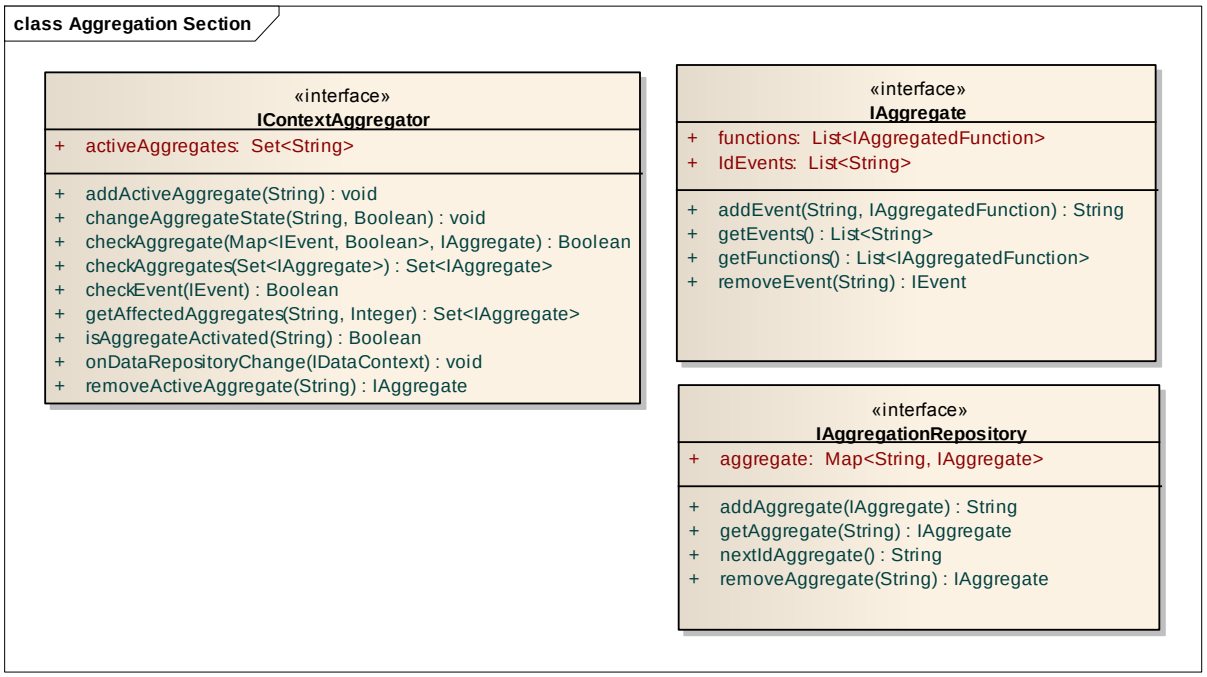


Figura 147: Conjunto de interfaces correspondientes al Aggregation Section del Core del middleware desarrollado

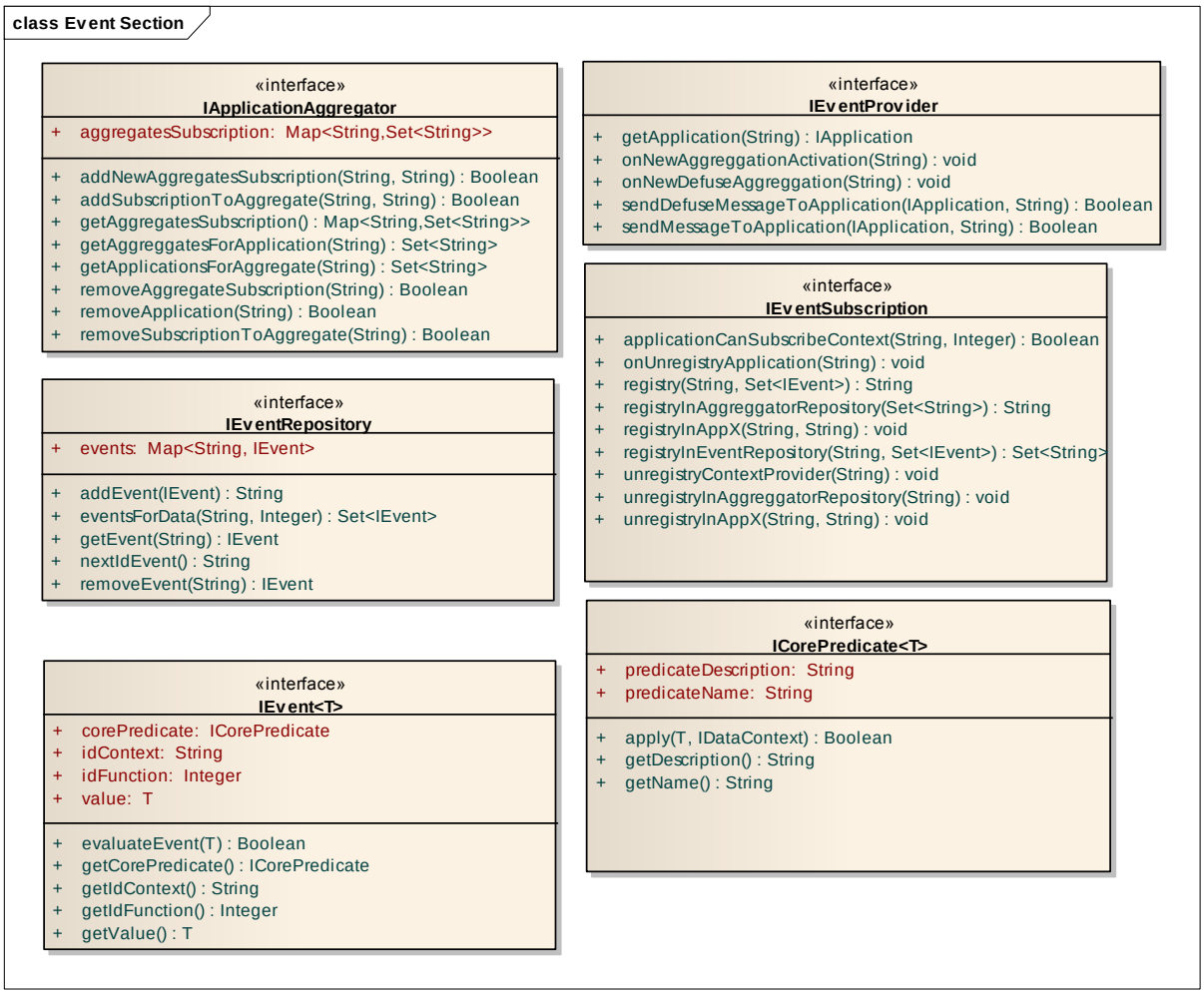


Figura 148: Conjunto de interfaces correspondientes al Event Section del Core del middleware desarrollado

APLICACIÓN

Interfaces

Con el fin de estandarizar el acceso a las *Applications* que usan el middleware, se define la siguiente interfaz denominada *IApplication*. El uso de esta interfaz por parte de las aplicaciones permitirá al *Core* acceder a los métodos necesarios en caso de que la aplicación realice una comunicación asíncrona mediante eventos.

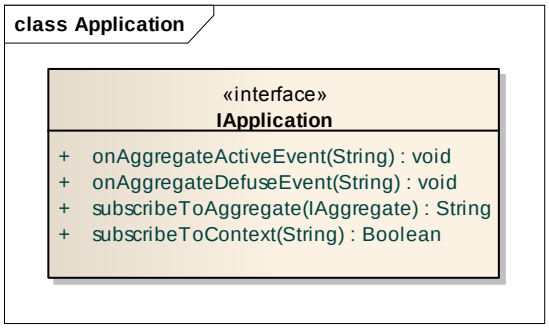


Figura 149: Diagrama de la interfaz IApplication

La interfaz *IApplication* consta de los siguientes métodos:

- *Void onAggregateActiveEvent(String idAggregate)*. Este método será llamado por el módulo *Event Subscription* del *Core* cuando se evalúe satisfactoriamente las condiciones del agregado cuyo identificador se pasa por parámetro. Para que este método sea llamado, la aplicación debe haber suscrito previamente dicho agregado en el núcleo del sistema.

- *Void onAggregateDefuseEvent(String idAggregate)*. Semejante al método anterior, este método será llamado por el módulo *Event Subscription* del *Core* cuando se evalúe negativamente las condiciones del agregado cuyo identificador se pasa por parámetro. Para que este método sea llamado, además de cumplirse que la aplicación suscribiera previamente el agregado, es necesario que el agregado fuera evaluado positivamente en el instante anterior. De este modo, el método actual informará a la aplicación cuando un agregado que estaba activo, ha dejado de estarlo.

- *String subscribeToAggregate(IAggregate aggregate)*. El método actual suscribirá un agregado generado por la aplicación en el núcleo del sistema. De este modo, cuando se evalúe las condiciones de los eventos que forman el agregado de manera positiva, se informará a la aplicación a través de los métodos anteriores. La cadena que devuelve la ejecución de este método será el identificador del agregado una vez inscrito en el sistema. Este identificador será útil a la hora de la comunicación entre el *Core* y la *Application*, puesto que cuando se activa o desactiva un agregado, el núcleo tan sólo enviará a la aplicación el identificador del mismo.

- *Boolean subscribeToContext(String idContext)*. Este método se define con el fin de inscribir la aplicación en un contexto del que podrá entonces obtener información. Este método, a su vez, llamará al punto de acceso ofrecido por el *Core* en el *Context Discover* para llevar a cabo la suscripción en el contexto. El parámetro del método actual es el identificador del contexto al que se desea inscribir, mientras que el valor devuelto será un elemento lógico que indica si la inscripción se ha realizado correctamente.

Con los métodos anteriores, una aplicación tendrá la capacidad de comunicarse con el *Core* para la inscripción a eventos y la obtención de información y, lo que es aún más importante, el núcleo podrá comunicarse con la aplicación informando de los eventos activados y desactivados.

En este caso, a diferencia del *Core* y el *Context Provider*, no será necesario definir ninguna clasificación adicional a las existentes. Las aplicaciones tan sólo harán uso de las clasificaciones definidas anteriormente para realizar consultas sobre los contextos disponibles así como para llevar a cabo los agregados en base a las relaciones entre eventos.

ANEXO III

CURRICULUM DEL AUTOR

Durante los últimos años han sido realizadas un conjunto de aportaciones científicas en el ámbito de sistemas ubicuos y contextuales, concretamente en el ámbito del reconocimiento físico de actividades, posicionamiento y de los middlewares para sistemas móviles. Además, gracias a la línea de investigación del desarrollo de aplicaciones contextuales a través de middlewares, se están realizando actualmente numerosas aportaciones al campo de la computación en dispositivos móviles.

2009

Durante este año el investigador participó en el proyecto de investigación andaluz *CUBICO (Sistema de CUidados UBICuOs y asistencia controlado por familiares y centros médicos para personas con dependencia)*, en el cual se realizaron aportaciones en el ámbito de la obtención y procesamiento de datos procedentes de sensores electrocardiógrafos (ECG). Realizando el procesamiento de dichos datos en el propio dispositivo móvil.

Este mismo año, el investigador formó parte del comité organizador de las XI Jornadas ARCA (*Sistemas Cualitativos, Diagnosis, Robótica, Sistemas Domóticos y Computación Ubicua*) llevadas a cabo en Almuñécar (Granada).

Respecto a publicaciones de carácter científico, el investigador llevó a cabo las siguientes.

- **Título:** An approach for saving energy in smart environments
  Autores: A. Fernández-Montes, J.A. Ortega, L. González-Abril, M.L. Vilchez y L.M. Soria.
  Publicado en: XI Jornadas en Sistemas Cualitativos, Diagnosis, Robótica, Sistemas Domóticos y Computación Ubicua, JARCA ‘09, Granada, Junio 2009. ISBN: 978-84-613-71587

2010

En el año 2010 el investigador participó en el proyecto del Ministerio de Educación y Ciencia denominado proyecto *InCare-Famenet (plataforma integral para la interacción entre familiares y centros médicos para la atención telemática a personas con dependencias)*. En dicho proyecto se inició la investigación en nuevos métodos de detección, mediante dispositivos móviles, de actividades físicas llevadas a cabo por usuarios con dependencias, con el fin de monitorizar y publicar dichas actividades y ponerlas a disposición del equipo sanitario o los familiares del usuario.

Junto al anterior, el investigador participó en el proyecto del Ministerio de Educación y Ciencia llamado *ARTEMISA (arquitectura para la eficiencia energética y sostenibilidad en entornos residenciales)*. En el mismo, el autor se familiarizó con las diferentes técnicas de eficiencia energética y la necesidad de su aplicación en dispositivos autónomos, como es el caso de los dispositivos móviles.

Por último, entre los proyecto más destacados en los que participó el investigador en este periodo, se encuentra el denominado *TIM (televisión interactiva multimedia)* realizado a través del Centro de Informática Científico de Andalucía (*CICA*). En este proyecto se colaboró con la empresa *Guadaltel* para el desarrollo de una plataforma de multimedia para entornos televisivos. Gracias a este proyecto se comenzó el desarrollo de una aplicación basada en el reconocimiento de actividades físicas, con el fin de ofrecer a los familiares de usuarios con dependencias o al propio usuario, la capacidad de consultar el histórico y diferentes estadísticos de las actividades físicas realizadas a través de aplicaciones televisivas.

Así mismo se realizaron las siguientes aportaciones en congresos nacionales:

- **Título:** Context-triggered applications for mobile devices
  Autores: L.M. Soria, J.A. Álvarez, J.A. Ortega y L. González-Abril
  Publicado en: Ubiquitous Compunting & Ambient Intelligence, UCAmI 2010 (CEDI 2010), Valencia, Septiembre 2010. ISBN: 978-84-92812-61-5

- **Título:** Framework for systems based on context
  Autores: L.M. Soria, J.A. Álvarez y J.A. Ortega
  Publicado en: XII Conference on Energy Efficiency and Sustainability in Intelligence Environments, JARCA 2010, Palma de Mallorca, Junio 2010. ISBN: 978-84-614-6457-9

- **Título:** An Approach to the Implementation of Web TV Architecture with Interactive Services
  Autores: A.M. Bellido, M.A. Álvarez, L.M. Soria, J.A. Ortega, J. Torres
  Publicado en: XII Conference on Energy Efficiency and Sustainability in Intelligence Environments, JARCA 2010, Palma de Mallorca, Junio 2010. ISBN: 978-84-614-6457-9

En cuanto a congresos internacionales, el autor ha realizado las siguientes publicaciones:

- **Título:** Tracking system based on accelerometry for users with restricted physical activity
  Autores: L.M. Soria, J.A. Álvarez, J.A. Ortega y L. González-Abril
  Publicado en: XXIII International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, IEA/AIE ‘10, Córdoba, Junio 2010. ISBN: 978-84-613-71587

A lo largo del año 2010, se solicitó la inscripción del trabajo *SCI (Simple Cluster Interface): Arquitectura para la gestión de tareas de usuario en un clúster a través de la web*, en el registro de patentes del Ministerio de Industria.

Respecto a otros méritos científicos conseguidos por el investigador, se recoge la aportación como revisor colaborador en el congreso internacional *IEA/AIE 2010 (Twenty Third International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems)*.

2011

Durante el presente año, el investigador siguió trabajando en los proyectos citados anteriormente, además de completar su participación en otros proyectos de investigación que se recogerán más adelante. Gracias a dichos proyecto ha sido posible el desarrollo del middleware de aplicaciones basadas en contexto, el cual sigue actualmente su desarrollo con numerosas ampliaciones sobre la idea original.

En relación a las publicaciones en revistas resultantes de esta tesis doctoral se encuentran las siguientes:

- **Título:** Efficient purchasing
  Autores: L.M. Soria, J.A. Álvarez y J.A. Ortega
  Publicado en: IEEE Pervasive Computing, Abril 2011. D.O.I.: 10.1109/MPRV.2011.24

En cuanto a congresos internacionales, el autor ha realizado las siguientes publicaciones:

- **Título:** Mobile architecture for communication and development of applications based on context
  Autores: L.M. Soria, J.A. Ortega, L. González-Abril y J.A. Álvarez
  Publicado en: XXIV International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, IEA/AIE '11, Syracuse, EE.UU., Junio 2011.

A lo largo del año 2011, se solicitó la inscripción del trabajo *DiLoS: Dispositivo de localización y seguimiento energéticamente eficiente*, en el registro de patentes del Ministerio de Industria. La patente anterior fue el resultado del trabajo llevado a cabo en torno al sistema de detección de salidas y su aplicación a la reducción energética en sistemas de posicionamiento.

Respecto a otros méritos científicos conseguidos por el investigador, éste es miembro revisor de la revista *Journal of Pattern Recognition Research*.

Por último, durante el año 2011, el autor realizó una estancia predoctoral de cuatro meses de duración en el centro de investigación alemán DFKI Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (Centro Alemán de Investiagación para la Inteligencia Artificial).

Proyectos de investigación

Esta tesis doctoral se ha desarrollado en el marco de los siguientes proyectos de investigación:

- **Nombre:** INCUBA (P025-09/E05)
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Sociedad Incubadora de Emprendedores de Andalucía, S.L.U.
  Periodo de duración: 2009-2010

- **Nombre:** INTEGR@:Integración de Servicios y Nuevas Tecnologías en el Alojamiento del Cliente dentro de un Espacio Turístico (P033-09/E05)
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Junta de Andalucía
  Periodo de duración: 2008-2009

- **Nombre:** ATENEA, Arquitectura Middleware y Herramientas. (FIT-340503-2007-1)
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Ministerio de Industria, Turismo y Comercio
  Periodo de duración: 2006-2009

- **Nombre:** TIM: Televisión Interactiva Multimedia (CTA- TIM.09)
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Guadaltel S.A.
  Periodo de duración: 2009-2010

- **Nombre:** OSAMI-Commons Open Source Infraestructure Ambient Inteligence Commons (P024-08/E05)
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Ministerio de Industria, Turismo y Comercio
  Periodo de duración: 2008-2011

- **Nombre:** InCare: Plataforma abierta para la integración en el hogar de servicios cooperativos de teleasistencia y telemedicina (TSI2006-13390-C02-02).
  Investigador principal: Ralf E.D. Seepold.
  Organismo financiador: Ministerio de Ciencia e Innovación
  Periodo de duración: 2007-2010

- **Nombre:** Sistema de cuidados ubicuos y asistencia controlado por familiares y centros médicos para personas con dependencias - CUBICO (TIC2141).
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Junta de Andalucía
  Periodo de duración: 2007-2010

- **Nombre:** Arquitectura para la eficiencia energética y sostenibilidad en entornos residenciales (TIN2009-14378-C02-01)
  Investigador principal: Juan Antonio Ortega Ramírez
  Organismo financiador: Ministerio de Ciencia e Innovación
  Periodo de duración: 2009-2011

LUIS MIGUEL SORIA MORILLO

Sevilla, 28 de junio de 2011