TESIS DOCTORAL

2015

DGL: LENGUAJE DE DISEÑO GENÉRICO, MULTI-PARADIGMÁTICO Y EJECUTABLE

ISMAEL ABAD CARDIEL

Licenciado en Informática

DEPARTAMENTO
DE INGENIERÍA DE SOFTWARE
Y
SISTEMAS INFORMÁTICOS

E.T.S.I. INFORMÁTICA

UNIVERSIDAD NACIONAL
DE
EDUCACIÓN A DISTANCIA

JOSÉ ANTONIO CERRADA SOMOLINOS

EUGENIO ARELLANO ALAMEDA

Departamento de Ingeniería de Software y Sistemas Informáticos

Escuela Técnica Superior de Ingeniería Informática

DGL: Lenguaje de Diseño Genérico, Multi-paradigmático y Ejecutable

Autor: Ismael Abad Cardiel

Director: José Antonio Cerrada Somolinos

Codirector: Eugenio Arellano Alameda

Madrid
Noviembre 2015

Para Gabriel y Manoli

Índice general

Índice de figuras

ÍNDICE DE FIGURAS

Índice de tablas

A scientific theory should be as simple as possible, but no simpler

A. Einstein

CAPÍTULO

1

Introducción

En nuestros días resulta indudable afirmar que los cambios en las predicciones de las propuestas iniciales de la Computación Ubícua de Weiser [1] y del "Internet de las Cosas" propuesto por Ashton en su presentación en el MIT (*Massachusets Institute of Technology*) del laboratorio para la autoidentificación (AutoID Lab) [2] se han visto superadas tanto en las funciones como en las dimensiones definidas. La computación ubicua y oculta, el despliegue masivo de dispositivos, la extensión de dispositivos sensores y actuadores, la ocupación de Internet por datos automáticos, se ha intensificado de tal forma que cualquiera de las predicciones iniciales se han quedado cortas. Por ejemplo, en los primeros años de aparición del término "Internet de las cosas", 2005[3], 2008 [4], se estimaba que el número de objetos que estarían conectados a Internet en el año 2020 sería de 10.000 millones, esa misma estimación en el año 2011 [5] se corrigió a 50.000 millones, y en la actualidad se está considerando que el número de objetos conectados podría llegar a ser de 100.000 millones [6] para ese futuro cercano del 2020.

En este contexto, aparecen varios temas de investigación sobre los que trabajar que se pueden agrupar en las siguientes líneas generales:

- Estandarización: El despliegue masivo de "cosas" sobre el Internet actual exige la adopción de medidas que faciliten la integración. La aparición de consorcios de empresas y de grupos organizados en las diferentes áreas de

producción ha llevado al ISO al desarrollo de un informe preliminar [7] en el que se reconoce la falta de un estándar armonizado sobre el IoT y la existencia de más de 400 normas propias que se pueden vincular al desarrollo de una arquitectura de referencia. Además, en el comité de sistemas de la información han comenzado el desarrollo de un estándar de referencia denominado ISO/IEC AWI 30141 (IoT RA: Internet of Things Reference Architecture).

- Regulación. Las normas y leyes de los países no pueden adaptarse a las exigencias necesarias del marco del IoT. Las organizaciones supranacionales, como la Organización Mundial del Comercio (OMC)[8] y la Unión Internacional de las Telecomunicaciones (UIT-ONU) [9] han establecido comisiones de trabajo sobre la regulación del IoT.
- Registro de la propiedad de la información. La generación automática y oculta de información desde todo tipo de sensores establece nuevas exigencias de organización del registro de la propiedad [10]. Establecer la propiedad de esa información, y cómo garantizar que esa propiedad se proteja de alguna forma, plantea nuevos requisitos y necesidades a las aplicaciones que la manejen.
- Identificación automática: El uso de la identificación por radiofrecuencia (RFID) ha supuesto el primer, y necesario paso en el despliegue del IoT. Como se comenta en la presente tesis, la capacidad de autoidentificar cualquier objeto de forma simple, sencilla y de bajo coste, supone dotar a los objetos de la capacidad para establecer vínculos con el resto de "cosas" del entorno. Es necesario, por lo tanto, buscar lenguajes y modelos para integrar la información de autoidentificación y el resto de información heterogénea.
- Seguridad: Las comunicaciones entre los objetos del IoT se exponen a todo tipo de riesgos una vez que se establecen intercambios por redes de todo tipo. Es necesario establecer los mecanismos que garanticen la integridad y la validez de la información para poder garantizar un correcto funcionamiento.
- Integración. En nuestro entorno actual, la integración del mundo real y de Internet, se encuentra separado por esa frontera establecida del "acceso". Sin embargo cada vez más este acceso se comienza a difuminar debido a que factores como el coste, la movilidad y la seguridad se resuelven de una u otra forma por los recursos tecnológicos disponibles [11].

- Coordinación. El funcionamiento de los objetos en una estructura global sólo será posible si se establecen mecanismos de coordinación entre ellos como generadores/receptores de información, los servicios que los usan y las personas que se benefician.

Las propuestas que se incluyen en este trabajo se enmarcan en las cuestiones relacionadas con la integración, la coordinación y la identificación automática, con el objetivo fundamental de aportar herramientas alternativas y diferentes a las propuestas existentes.

1.1 Motivación

Uno de los objetivos tradicionales de la computación ubicua es la integración de la realidad física con procesos de computación. El desarrollo de los dispositivos de adquisición de información y de actuación automáticos, embebidos en todo tipo de objetos de nuestra vida cotidiana, y la incorporación de nuevos mecanismos de comunicación, sobre todo inalámbricos, nos conducen hacia un nuevo entorno que se conoce con el término inglés de: *"pervasive computing"*. Este término se traduce en los diccionarios como *"computación omnipresente"*. El adjetivo omnipresente[1] tiene poco sentido aplicado al término computación en nuestro idioma, por lo que es necesario incorporar a esta traducción algún adjetivo alternativo. El término inglés quiere, sobre todo, reflejar tres características asociadas a los nuevos entornos de proceso: la continuidad, la transparencia y la inconsciencia u ocultación. La continuidad quiere significar que el uso de dispositivos en nuestro entorno y que el proceso de la información que generan es incesante. La transparencia refleja la ausencia de una necesaria interacción con los dispositivos. El funcionamiento de estos dispositivos no exige la interacción de un usuario, pueden funcionar de forma autónoma y automática. Y por último, la transparencia, refleja ese entorno real en el que cada vez más nos encontramos, rodeados de dispositivos que interactúan y que recogen información sobre los contextos en los que vivimos. Por ejemplo, las redes de sensores ambientales recogen información para crear aplicaciones de supervisión del medio ambiente en las ciudades inteligentes [12], o las aplicaciones domóticas que integran sensores y actuadores sobre nuestras casas para controlar desde nuestros dispositivos multimedia, a los sistemas de calefacción y aire acondicionado, o incluso la gestión de los sistemas eléctricos [13], [14], [15].

Estas nuevas características y orientaciones de los sistemas de proceso de información han provocado que las cuestiones relacionadas con el IoT se hayan incluido en las listas de metas tecnológicas de investigación a nivel mundial. Así, en Estados Unidos, el *National Intelligence Council* de la Fundación Americana de la Ciencia (*NSF: National Science Foundation*), ya incluía en su lista de objetivos de tecnologías de interés para el marco 2025 [16], el IoT como uno de los seis principales

[1] El diccionario de la RAE incorpora dos significados a este término: que está presente en todas partes y que procura acudir deprisa a todos los sitios.

objetivos; y en 2010 el *President's Council of Advisor and Technology*, en el informe titulado "Diseño del futuro digital: financiación federal de la investigación y el desarrollo"[17], incluía como objetivo prioritario del programa de investigación de sistemas ubícuos y computación móvil *Cyber-Physical* debido tanto al impacto estratégico como económico. De igual forma, la Unión Europea incluyó en el programa marco de investigación europea FP7 (*The Seventh Framework Programme 2007 to 2013*) varios proyectos relacionados con el IoT [18] [19], y que dieron lugar al *IERC (Internet of things European Research Cluster)* como centro para recapitular los resultados de alrededor de los treinta proyectos que se han desarrollado. Y ha incluido en las nuevas ofertas de investigación del programa FP8, desde las primeras versiones aparecidas en 2011, entre los objetivos de investigación para el periodo que va de 2014 a 2020 temas relacionados directamente con el desarrollo y las tecnologías vinculadas de una u otra forma con el IoT.

El objetivo final del IoT es poder conectar en una red a cualquier tipo de dispositivo y a cualquier usuario que quiera y pueda acceder a la información. El problema que aparece al intentar resolver este simple enunciado es justamente la variedad de dispositivos y de formas de acceso a la información: que no importe el sistema de los dispositivos, cuál sea la plataforma empleada, qué función se realiza, o cómo se conectan; esto es lo que conocemos como la heterogeneidad del IoT [20], [21]. Esta heterogeneidad del entorno del IoT está restringida por dos funciones que debe poseer cualquier objeto que se integra en la red. Primero, cada objeto debe ser identificable [22]. Según cuál sea el escenario, esta identificación puede ser única o puede pertenecer a una clase o tipo de objeto, en donde se identifique de forma adicional. La identificación en el IoT actual procede en gran medida de sistemas de identificación tradicionales basados en códigos de barras, códigos QR o similares, y cada vez más se incluyen sistemas RFID. La segunda característica que deben incluir los objetos es tener la capacidad de adquisición o actuación(*sensing/actuation*). En la actualidad esto se resuelve con sistemas de adquisición y de actuación realizados ad-hoc para aplicaciones particulares y funciones propias. De forma resumida podemos expresar el IoT con las siguientes tres características:

- Todo comunica: independientemente del modo de comunicación, los objetos deben tener la capacidad de comunicar entre ellos y con otros sistemas.

- Todo está identificado: las relaciones entre los objetos pueden especificarse en un dominio en el que se puede diferenciar entre ellos ya sea con identificación propia o asignada por otros sistemas.
- Todo interactúa: los objetos generan información en el entorno o producen efectos sobre él.

En los últimos años se han acumulado propuestas sobre cómo implementar arquitecturas relacionadas con el IoT. Entre estas propuestas, es necesario destacar las relacionadas con organizaciones de estandarización como la del consorcio OGC (*Open Geospatial Consortium*), la del W3C-SSN (*Semantic Sensor Network*), y el EPCGlobal Network promovido por la OMC (Organización Mundial de Consumo). Algunas de estas arquitecturas incluyen componentes para resolver la comunicación entre objetos, plataformas *middleware* para facilitar el desarrollo de aplicaciones, plataformas para proporcionar servicios, y metodologías y herramientas para facilitar el despliegue de sus propuestas. Sin embargo, en estas propuestas no se ha hecho hincapié en resolver y proporcionar herramientas sencillas que faciliten el desarrollo de aplicaciones para el IoT de forma independiente de la plataforma que se utilice. De hecho, en la mayor parte de las propuestas existentes, la incorporación de aplicaciones viene dada por el uso de los mecanismos propios y particulares de cada una de ellas, y que suele incluir sólo herramientas de configuración sin que exista capacidad para dar soporte a nuevos elementos [23]. Y en el caso de querer introducir nuevas funcionalidades o componentes exige invertir gran cantidad de esfuerzo en el diseño y desarrollo a incorporar. Por lo tanto, es necesario investigar y proponer nuevas opciones en el desarrollo del IoT, que faciliten y simplifiquen el despliegue de objetos sensores y actuadores sin las exigencias de esfuerzo y coste actuales.

1.2 Objetivos

A partir del entorno heterogéneo del IoT comentado en la motivación, surgen dos retos principales: el primero de ellos es buscar mecanismos que permitan gestionar la heterogeneidad del entorno y el otro es proporcionar mecanismos para facilitar el desarrollo de aplicaciones en el mismo.

En el presente trabajo se plantean los siguientes objetivos principales para resolver estas dos cuestiones. El primer objetivo es aportar un modelo que permita la definición de redes de dispositivos para el IoT basadas en redes de adquisición RFID. A este modelo lo denominaremos metaDEPCAS (que utiliza como base particular el *middleware* DEPCAS (*Data EPC Acquisition System [24]*) para la adquisición de información. El segundo objetivo consiste en la definición del lenguaje de dominio DGL (*Design Generic Language*) que permita la definición de sistemas del IoT tanto en su configuración estática como en su configuración dinámica basados en el modelado de metaDEPCAS. Y tercero, establecer el proceso de uso del DGL para generar aplicaciones para el IoT.

El primer objetivo, la definición de metaDEPCAS permite disponer de una representación de aplicaciones para el IoT. El modelo metaDEPCAS posibilita poder utilizar una herramienta que tiene como principal objetivo representar el funcionamiento de sistemas heterogéneos en el IoT integrando la información procedente de sensores o que se envía a actuadores, con la información de identificación de tipo RFID, en un modelo único. En este trabajo, nos centramos en utilizar únicamente información de identificación basada en RFID dado que se considera que este será el recurso tecnológico fundamental que dará soporte de identificación automática en el IoT.

El segundo objetivo, consiste en la propuesta de un lenguaje específico de dominio para el diseño de aplicaciones para el IoT sobre el modelo y el metamodelo antes comentado. Este lenguaje intenta aportar soluciones a la falta de herramientas propias para la creación de aplicaciones para el IoT tomando como base los elementos específicos del dominio particular del IoT.

Por último, el tercer objetivo fundamental consiste en la descripción del proceso para el uso del DGL de forma que se facilite la incorporación del DGL como herramienta para el desarrollo de aplicaciones. Esta herramienta se caracteriza por

proporcionar un mecanismo de alto nivel que reduce la complejidad y el esfuerzo de desarrollo de nuevas aplicaciones, a la vez que aporta un mecanismo de automatización en la generación de las aplicaciones para el IoT.

1.3 Organización de la tesis

La tesis se organiza en los siguientes tres bloques:

- **Capítulo 2:** repasa las propuestas existentes relacionadas con el IoT. Esta revisión del estado del arte incluye apartados específicos relacionados con la RFID, las redes de sensores, las características de heterogeneidad de las redes en el IoT, y el desarrollo de aplicaciones para el IoT.

- **Capítulo 3:** define el modelo denominado *metaDEPCAS* para el modelado de sistemas heterogéneos de adquisición basados en identificación por RFID. En este capítulo se incluyen los conceptos utilizados para el modelado del IoT, una descripción de los sistemas DEPCAS y los elementos del metamodelo.

- **Capítulo 4:** incluye la descripción del lenguaje de dominio específico, un DSL (*Domain Specific Language*) para el diseño de aplicaciones para el IoT sobre DEPCAS y metaDEPCAS, que hemos denominado DGL. En este capítulo se incluyen las secciones sobre los conceptos utilizados para la definición del lenguaje y los elementos del propio DGL. En el último apartado se presenta la descripción del proceso de desarrollo de aplicaciones con el DGL para el IoT.

Para terminar, el **Capítulo 5** incluye las conclusiones más relevantes que aporta este trabajo y describe las posibles líneas de trabajo a continuar en la misma línea de investigación.

Personally, I think it does help, that it makes a beneficial difference, but the scientific literature on the subject is very messy.

Jeanne Petrek

CAPÍTULO

2

Estado del arte

Este capítulo presenta una revisión de los principales elementos que conforman el IoT. Comenzamos con una revisión de las definiciones sobre el IoT y las principales propuestas actualizadas sobre arquitecturas para el IoT. En esta revisión también se incluyen los conceptos fundamentales relacionados con la RFID y las redes de sensores. También ser revisan los elementos heterogéneos que caracterizan los sistemas que se incluyen en el IoT. Y por último, se comentan diversas herramientas y propuestas relacionadas con el desarrollo de las aplicaciones en el IoT.

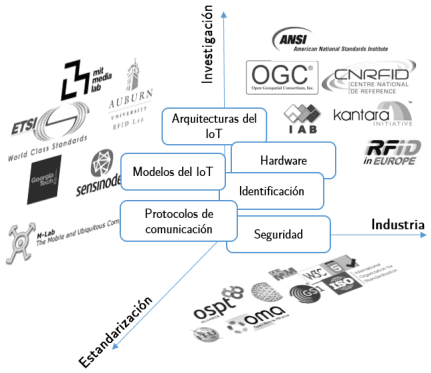

Figura 2.1: Vistas del IoT.

Elementos y roles involucrados en el IoT

2. ESTADO DEL ARTE

Las diferentes aristas que se presentan en el IoT dificultan disponer de una visión única. Las perspectivas que se pueden afrontar según el usuario, el desarrollador o la tecnología plantean cuestiones muy diferentes. Además, según los objetivos que se quieran conseguir, los productos finales que podemos obtener serán distintos para la investigación, la industria o las organizaciones de estandarización. Y también es necesario considerar que los elementos que se involucran en el IoT disponen de su propia perspectiva según se pretendan resolver. Así surgen cuestiones para resolver: la arquitectura del IoT, el modelado del IoT, la identificación y el direccionamiento de objetos, la implementación y la integración de protocolos de comunicación, el uso de hardware, o los aspectos relacionados con la seguridad y la privacidad de la información.

2.1 IoT. El internet de las cosas

La primera mención al término "Internet de las cosas" refiere a la presentación de K. Ashton en 1999 [2]. En esta presentación, el director del AutoID Lab del MIT, defendió que el uso de las etiquetas de identificación por radio frecuencia, RFID, incorporadas a cualquier "cosa" de nuestro entorno daría lugar a un nuevo sistema para el que utilizó el término "Internet de las cosas". La visión inicial del IoT presentaba una cadena de producción, desde las fábricas hasta los consumidores en la que los productos eran etiquetados con RFID y la información asociada a ellos se almacenaba en bases de datos distribuidas por Internet para automatizar los procesos de toda la cadena. A pesar de que esta visión inicial sólo se ha alcanzado de forma parcial y, sólo en determinados entornos, el término original ha ido incrementando su significado con la integración de otros muchos objetos del mundo real en Internet. De esta forma, el IoT ha incluido como objetivo principal integrar, en la medida de lo posible, cualquier interacción entre los objetos del mundo real, en el mundo de virtual de Internet. Los dispositivos sensores, los usuarios introduciendo y obteniendo información, y los equipos actuadores modificando el entorno buscan cómo quedar incluidos en el entorno virtual, y se distribuyen poco a poco por toda nuestra vida cotidiana en: los coches, los teléfonos móviles, las tabletas, la ropa inteligente, los equipos del hogar, etc. Esta visión ampliada del IoT ha provocado que, a día de hoy, no exista una definición única y ampliamente aceptada del concepto de "Internet de las cosas". En las siguientes secciones se presentan las definiciones más relevantes de este concepto y las propuestas de arquitectura del IoT más extendidas.

2.1.1 Definiciones del IoT

A continuación se presentan cuatro definiciones del IoT desde diferentes puntos de vista como ejemplo de la variedad de planteamientos que existen sobre el mismo tema.

1. Definición ITU. El organismo de la ONU para las telecomunicaciones, en la reunión de 2011 sobre el IoT, acordó la siguiente definición:

Versión corta: *El IoT es una red dinámica global y una infraestructura de servicios para conectar cosas.*

Versión larga: *El IoT es una red dinámica global y una infraestructura de servicios con capacidad de configuración propia basada en protocolos estándar e interconectables para conectar objetos heterogéneos que tienen identidad, atributos físicos y virtuales, de forma segura e integrada en Internet.*

2. Definición Proyecto CASAGRAS. El proyecto CASAGRAS pertenece al grupo de proyectos financiados por el FP7 de la Unión Europea. Este proyecto aporta la siguiente definición del IoT:

   El IoT es una infraestructura de red global que enlaza el mundo físico y el mundo virtual de las cosas a través de la explotación de los datos capturados y de las capacidades actuales de comunicación. Esta infraestructura incluye las aplicaciones actuales y las futuras evoluciones de Internet. Incluye la identificación de los objetos, la capacidad de integrar sensores y actuadores con conexión para el desarrollo de aplicaciones y servicios cooperativos e independientes. Caracterizado por un alto grado de autonomía en la captura de datos, transferencia de eventos, conectividad e interoperabilidad.

3. Definición SAP. La visión de la empresa privada dedicada a los grandes sistemas corporativos presenta la siguiente definición:

   El IoT es el entorno en que los objetos físicos autónomos se integran en una red de información para aportar información relevante al negocio. Los servicios de esos "objetos inteligentes"se ofrecen sobre la infraestructura de Internet.

4. Definición del FIA (*Future Internet Association*). La FIA es una asociación sin ánimo de lucro dedicada a dar pronósticos sobre las futuras vías y aplicaciones en Internet, que presenta el IoT como:

> *El IoT que inicialmente se propuso alrededor del RFID y de los WSAN (Wireless Sensor and Actuators Networks), es en nuestros días un conjunto variado y amplio de dispositivos con diferentes capacidades de proceso y comunicación interconectados en una red NED (Nework Embedded Devices) propia o integrada en Internet.*

En estas definiciones se muestran las diferencias existentes en los planteamientos. Mientras que la definición del ITU y del proyecto CASAGRAS se centran en las infraestructuras para conectar cosas con capacidad de adquirir información, capacidad de proceso y de conexión, la definición de la FIA se centra en las propiedades de las cosas y como distribuirlas por Internet. Mientras que la de SAP, pone el foco sobre el impacto en los procesos de las empresas. Con esta revisión de la literatura podemos plantear tres niveles en los que representar los desarrollos en el IoT: el primer nivel sería el que se dedica a resolver las cuestiones físicas y lógicas relacionadas con los protocolos y los dispositivos (RFID, sensores/actuadores, ..); el segundo nivel sería el relacionado con la integración de la información en la estructura de Internet actual; y el tercer nivel trataría de las aplicaciones que emplean esta información.

A modo de resumen y conclusión sobre las definiciones podemos concentrarlas con las siguientes dos definiciones básicas para las "cosas"del IoT y el "IoT":

Cosa. *Una "cosa" es cualquier objeto identificable con capacidad de proceso y comunicación con sensores y actuadores para interactuar con el entorno real.*

IoT. *El IoT es una extensión del Internet, que conecta el mundo real con el mundo virtual, a partir de los datos capturados, de las acciones realizadas y de los procesos ejecutados por las cosas.*

A partir de estas definiciones en los últimos años se han ido proponiendo y desarrollando diversas plataformas para dar soporte al crecimiento del IoT que se comentan en el siguiente apartado.

Plataforma	a) Soporte para dispositivos heterogéneos	b) Tipo	c) Arquitectura	d) Código libre	e) Protocolo REST	f) Forma de acceso a la información
Arkessa	Si	M2M PaaS	Basada en la nube	No	Si	Política propia de usuarios
DeviceCloud	Si	PaaS	Basada en la nube	No	Si	No incluye gestión propia
IoT-framework	Si	Servidor	Centralizada	Si	Si	Local
Nimbits	Si	Servidor	Centralizada y basada en la nube	Parcial	Si	3 niveles propios
OpenIoT	Si	Hub	Descentralizada	Si, LGPL	No	Privilegios de usuario
The Thing System	Solo dispositivos domóticos	Servidor	Centralizada	Software del M.I.T.	Si	Privilegios de usuario
ThingWorx	Si	M2M PaaS	Basada en la nube	No	Si	-

Tabla 2.1: Características de las plataformas del IoT

2.1.2 Plataformas del IoT

En la actualidad podemos recoger información sobre alrededor de más de cincuenta propuestas de plataformas relacionadas con el IoT [25], [26]. La mayor parte de ellas presentan características comunes en cuanto a la funcionalidad básica: soporte para dispositivos heterogéneos, algún tipo de función para el M2M (*Machine To Machine*), arquitectura basada en Internet (la nube), algún tipo de implementación con el protocolo REST (*Representational State Transfer*), y algún mecanismo de configuración más o menos automático. La tabla 2.1 incluye alguna de las plataformas más relevantes en la actualidad según su utilización.

La tabla 2.1 sólo presenta las características más habituales de las plataformas

actuales del IoT sin indicar otros aspectos de mejora o de carencia que se pueden encontrar. Un análisis crítico de las plataformas nos permite establecer los diferentes aspectos relacionados con: la forma de integración de los sensores, los actuadores y las comunicaciones entre ellos con M2M o con otros recursos, la gestión que se realiza de la información que se genera, los procesos de desarrollo de aplicaciones y la integración con otros sistemas de las organizaciones como BPM (*Business Process Management*), ERP (*Enterprise Resource Planning*), CRM(*Custormer RelationShip Management*), BI (*Business Intelligence*), etc.

La integración de los equipos de adquisición de información y de actuación en las plataformas del IoT es uno de los aspectos fundamentales que las caracteriza. La tendencia inicial de sistemas ad-hoc y propietarios tanto en software como en hardware ha ido evolucionando hacia sistemas que intentan facilitar la incorporación de todo tipo de dispositivos. Esta resolución inicial ha hecho uso de los mecanismos más habituales de comunicación en Internet basando la comunicación en: el protocolo HTTP (*Hyper Text Tranfer Protocol*), el uso de websockets o gateways propios. Sin embargo estas opciones iniciales no resuelven las nuevas necesidades de los entornos que deben aportar soluciones para incluir todo tipo de dispositivos propietarios y restringidos de la manera más sencilla posible y con algún mecanismo de nombrado o direccionamiento que identifique cada elementos en la plataforma. Las propuestas más novedosas se dirigen al establecimiento de nuevos estándares que aportan mejores mecanismos para las redes heterogéneas y para resolver la comunicación M2M como pueden ser CoAP (*Constrained Application Protocol*), LwM2M (*Lightweight M2M*) o MQTT (*MQ Telemetry Transport*).

Las características de la estructura del IoT implican la generación y el manejo de una cantidad de información totalmente desconocida en los sistemas actuales (por ejemplo, los sensores de los motores de los aviones Boing se pueden generar hasta 20 TB de información por hora [27]) que plantea desafíos tanto en la gestión de la propia información como en la privacidad de la misma. La mayor parte de las propuestas actuales incluyen mecanismos de control, ya sea por el acceso a la información de los dispositivos, o por el acceso que hagan las aplicaciones al menos para la lectura y la escritura de información. Y no existe una capacidad de gestión de la información de grano fino para asegurar la propiedad y el acceso a la información de los dispositivos, lo que plantea problemas parar garantizar la

seguridad e integridad de la información. La tendencia del futuro próximo plantea el establecimiento de almacenamientos locales a los propios entornos a modo de concentrador para permitir que se puedan instaura mecanismos que aseguren la información tanto en cuanto a la propiedad como la integridad.

Las plataformas existentes en la actualidad proporcionan mecanismos tipo API (*Application Programming Interface*) basados en REST para facilitar el uso de sus servicios o *widgets* para poder incluirse en otras aplicaciones, siempre con limitadas funciones de acceso y uso de la información, puesto que son aportadas por cada plataforma para utilizar sus propias capacidades. Las alternativas más deseables serían que las plataformas integrasen SDK (*Software Development Kit*) o DSLs propios de los problemas del IoT que faciliten el desarrollo y la reutilización del software sin depender de una u otra capacidad de la plataforma específica.

Las plataformas que podemos manejar en estos momentos son útiles en el contexto de aplicaciones externas que pueden emplear la información que se genera en el IoT. Con esta tendencia las soluciones sobre el IoT se centran en aproximaciones verticales a cada sector, sin que existan opciones de reutilizar en la medida de lo posible el trabajo desarrollado en una solución en otra nueva [28]. Como solución a corto plazo se espera que la estandarización del ISO aporte un marco que pueda servir para unificar las arquitecturas.

2.2 RFID y software RFID

Este apartado incluye la descripción de los elementos de la tecnología RFID. Esta tecnología disponible desde la Segunda Guerra Mundial [29], en la que se empleó para identificar en los sistemas de rádar los aviones propios de los del enemigo, evolucionó durante toda la segunda mitad del siglo XX, hasta la aparición de un patente propiedad de IBM a principios de los años 90 que presentaba el primer sistema RFID en alta frecuencia (UHF-RFID). La aparición del Auto-ID Center del MIT en 1999, financiado por el UCC (*Uniform Code Council*), el EAN (*European Article Numbre*) , Gillete y Procter&Gamble, con el objetivo de crear un sistema RFID de bajo coste en alta frecuencia y enlazar la información de identificación con servicios de información en Internet, supuso el desencadenante del desarrollo actual de RFID. A la iniciativa inicial se unieron más de 100 compañias, incluido el USDoD (*U.S. Department of Defense*) y se crearon cinco centros más en las Universidades de Adelaida (Australia), Cambridge (Reino Unido), St. Gallen (Suiza), Fudan (China) y Keio (Japón) (actualmente se ha incluido la Universidad de KAIST en Corea del Sur). En la actualidad las áreas de aplicación de los sistemas RFID se han extendido en diversos ámbitos: transporte, agricultura, salud, etc., y es uno de los pilares fundamentales en el concepto del IoT.

2.2.1 Conceptos y terminología del RFID

El sistema RFID permite la identificación por señales de radio de cualquier objeto, animal o persona. La arquitectura básica está formada por dos elementos: las etiquetas o *transponders* y los equipos lectores. A esta arquitectura básica podemos incluir un tercer elemento que es el software de gestión de la identificación o *middleware* RFID.

Las etiquetas incorporan la información de identificación y están compuestas por la memoria de identificación y una antena que captura la señal emitida por los equipos lectores, y emplean su energía para activar y transmitir su información. Las etiquetas se pueden clasificar en pasivas, semi-pasivas y activas. Las etiquetas pasivas o de bajo coste usan la energía de la transmisión de los equipos lectores para activar y tranmitir la información. En este grupo se encuentras las etiquetas que

usan el EPC (*Electronic Product Code*) de clase 0 (sólo lectura), 1 (lectura/una escritura) y 2 (lectura/varias escrituras). Las etiquetas semi-pasivas incorporan algún mecanismo de alimentación propio que permite activar la memoria de la etiqueta pero no aumentar la potencia de transmisión de la antena. En este grupo encontramos las EPC de clase 3 que permiten incorporar más información en la memoria procedente de algún sensor o mayor información para seguridad. Las etiquetas activas incorporan fuentes de alimentación para generar la señal de radio y así disponer de mayor funcionalidad como proceso de los datos, encriptación y almacenamiento de información.

Los equipos lectores de RFID se encargan de la interrogación de las etiquetas para obtener el identificador del objeto que las lleva. Están formados básicamente por tres componentes: el módulo de control, el módulo de radio frecuencia y la o las antenas. El módulo de control incluye las funciones del protocolo de comunicaciones, tanto de radiofrecuencia como de la comunicación del lector con otros equipos. El módulo de radio frecuencia se encarga de la transmisión y recepción de las señales de radio y está unido a las antenas que se encargan de la propagación y recepción de las señales de radio. Podemos clasificar los equipos lectores en móviles o fijos. Los lectores móviles suelen integrarse en equipos propios de mano o integrados en otros equipos como teléfonos o tablets. Los equipos fijos se integran en otras infraestructuras como muros, portales o cajas de registro. Otra clasificación posible de los equipos lectores es según el interfaz de comunicación que ofrezcan. Pueden incorporar interfaces serie, por ejemplo USB, o interfaces de red, Ethernet, Bluetooth o ZigBee.

En cuanto a la frecuencia de radio que manejan, los equipos lectores pueden ser de baja frecuencia (125/134 KHz), alta frecuencia (13,56 MHz), ultra alta frecuencia (860-960 MHz) y super alta frecuencia (3 GHz) [30].

2.2.2 Redes de sistemas RFID

Los sistemas de lectura de RFID se organizan en redes de equipo lectores para dar soporte a una funcionalidad mayor. Podemos distinguir seis elementos en un sistema RFID tipo: etiquetas, antenas, lectores, *middleware*, servicios de información, e infraestructura de red (Figura 2.2).

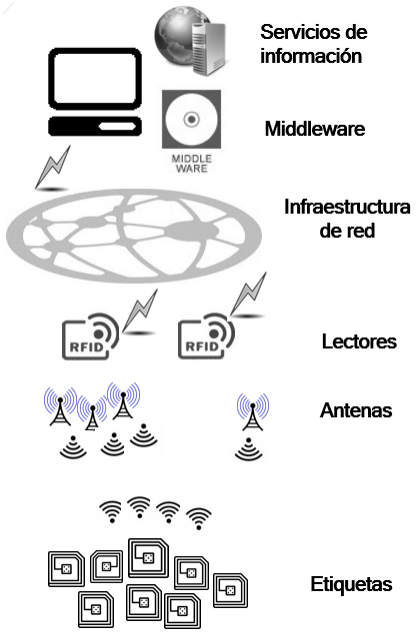

Figura 2.2: Arquitectura típica de un sistema RFID.

Componentes principales de un sistema RFID

El *middleware* que conecta los equipos lectores con los servicios de información tiene una función básica de filtrado, consolidación y distribución de los datos de identificación. Sirve de puente entre los lectores y las aplicaciones en donde se utilicen los datos y además puede utilizarse para controlar los equipos lectores y la red que los conecta. El término "servicios de información" es utilizado para explicar las conexiones de la información adquirida por radio frecuencia con el resto de aplicaciones de una organización. Estos servicios de *back-end* permiten integrar los datos del *middleware* en las aplicaciones que utilizan la información. La infraestructura de red permite conectar los equipos lectores con el *middleware* del sistema RFID. Puede ser una tecnología cableada basada en Ethernet o inalámbricas por WiFi o Zigbee.

Los sistemas RFID se escalan para permitir organizaciones de lectores y áreas de adquisición de información RFID que den soporte a las aplicaciones del IoT. Estas estructuras dan lugar a la distribución de la arquitectura RFID, ya sean en instalación de una organización o distribuciones más complejas en distintas áreas geográficas. La figura 2.3 muestra la distribución de un sistema RFID entre diver-

sas localizaciones de lectura. Los lectores del primer área recogen la información que envían al RNC (*Reader Network Controller)* vía protocolo TCP/IP. Los RNCs en los que reside el middleware de adquisición se encargan de envíar la información a los servidores de las aplicaciones coorporativas para que hagan el uso que corresponda según su función.

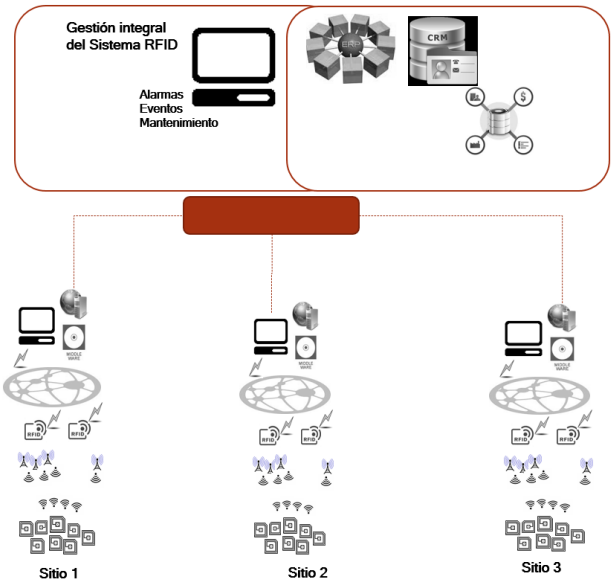

Figura 2.3: Sistema RFID distribuido.
Ejemplo de un sistema de redes de adquisición RFID

2.2.3 Evolución de los sistemas RFID

Tal y como hemos comentado en los apartados anteriores, la evolución de los sistemas RFID ha tenido tal auge que cualquier perspectiva previa ha sido superada. Aunque todavía hoy queda algo lejos la sentencia del "*one cent per tag*", otras barreras intermedias están superadas o a punto de superarse, como el auge de los dispositivos RFID de corto alcance, el uso masivo de contenedores basados en RFID, o las etiquetas a cinco centavos de dolar. Sin embargo, algunas de las cuestiones críticas en los sistemas RFID todavía quedan por resolverse de una forma definitiva:

- Anchos de banda. Los sistemas RFID pueden funcionar en diversas frecuencias, desde la baja frecuencia 125 KHz de los estándares para seguimiento animal ISO 14223 e ISO 18000-2, hasta las frecuencias ultra altas que suelen manejar la banda de los 900 a 915 MHz, para aplicaciones con alta tasa de lectura de etiquetas y largo alcance. Algunas de estas frecuencias se utilizan compartidas con otros dispositivos como la telefonía móvil, sistemas de alarma inalámbricos o sistemas SRD (*Short Range Devices*), lo que provoca conflictos y pérdidas de rendimiento en las identificación. Además, los rangos de frecuencia de los dispositivos lectores deben adaptarse a normativas de tipo regional con las dificultades y costes adicionales de integración. En un futuro próximo, las tendencias obligarán a flexibilizar el uso de las frecuencias con un mayor rango de espectro y a una unificación de los diferentes bloques regionales.

- Protección de la información. Las cuestiones relacionadas con la seguridad de la información en los sistemas RFID han sido uno de los aspectos más relevantes desde el punto de vista de la investigación. Los problemas derivados de la propagación de la información en entornos públicos y la obtención de relaciones entre todo tipo de actores que se identifican en los sistemas RFID plantean cuestiones tanto en cuanto a la privacidad como a la integridad. Los esfuerzos planteados han llevado diferentes propuestas de todo tipo: etiquetas más inteligentes que permita integrar mecanismos de protección como pueden ser claves *hash*, re-encriptación, o bloqueos selectivos; uso del comando especial "kill" para apagar etiquetas permanentemente; evitar la activación de etiquetas RFID en entornos no deseados con el uso de "jaulas de Faraday"; o provocar interferencias activas desde las propias etiquetas para que no sean leídas por quien no debe.

- Falta de estandarización. La falta de estandarización global a nivel mundial ha provocado que los fabricantes tengan que presentar diferentes opciones según el mercado en el que quieran distribuir sus productos RFID. En cuanto a las etiquetas de bajo coste en UHF, el estándar EPC Global Gen 2 ha obtenido una mayor aceptación, es necesario recordar que, dado que procede de una organización no gubernamental, la *EPCglobal*, introduce una incertidumbre

en cuanto a su desarrollo futuro. Frente al UHF, los sistemas en HF de casi todos los operadores se han centrado en el estándar ISO 15693 que define la forma de transmisión pero no la estructura de las etiquetas; y así aparecen incompatibilidades de formato de la información de lectores que conocen la forma de transmisión de las etiquetas pero no su estructura de información.

2.2.4 Aplicaciones de la RFID

La aplicación de tecnología RFID surge como uno de los elementos fundamentales del IoT para permitir la identificación automática de cualquier cosa sin necesidad de la intervención de otros recursos, y así dirigirnos hacia la virtualización completa del mundo real. El análisis del atlas del IoT [31] proporciona los siguientes sectores en los que se aplica esta tecnología:

- Edificios. La utilidad aplica tanto a entornos comerciales, institucionales como industriales, permitiendo identificar de forma automática sistemas de calefacción, aire acondicionado, sistemas anti-incendios, accesos, puntos de luz y suministro eléctrico, etc.

- Energía. El uso de sistemas RFID se ha extendido en los distintas líneas del sector energético: eléctrico, petróleo y gas, o en las energías renovables, como recurso, tanto para la gestión de las instalaciones como para todo tipo de procesos tanto de producción como de gestión de clientes.

- Consumo y hogar. La domótica y la electrónica de consumo son dos de las áreas en las que más se han integrado las tecnologías de radiofrecuencia. La integración de lectores de NFC (*Near Field Communication*) en dispositivos como cámaras o teléfonos móviles ha abierto el ámbito de aplicaciones de uso masivo que utilizan el sistema de radiofrecuencia de corto alcance: acciones automáticas como silenciar o activar las señales del móvil, transferir ficheros, etc.

- Salud e investigación. La integración de los sistemas RFID en los procesos sanitarios se ha producido en todo tipo de procesos básicos, por ejemplo en

hospitales para la identificación y seguimiento de pacientes o para el seguimiento de recursos médicos, gestión de implantes, la dispensación de fármacos; y en laboratorios para la identificación de muestras y seguimiento de procesos de análisis.

- Industria. La industria lo ha aplicado tanto en los procesos logísticos, como en los procesos productivos y de distribución la autoidentificación. Las líneas de producción incorporan sistemas RFID para automatizar las acciones robóticas de manipulación y simplificar las tareas que realizan.

- Transporte. El sector de los transportes ha integrado la identificación básicamente en dos aspectos: en los propios vehículos y en la gestión de los mismos. Los sistemas de peaje fueron uno de los primeros sectores en los que la tecnología RFID se aplicó de forma extendida [32] . Además, en la actualidad se utiliza para operaciones en el propio vehículo como apertura y cierre de puertas, manejos de los sistemas de audio, o avisos automáticos según el entorno.

- Comercialización. La utilización de identificación automática para todas las operaciones relacionadas con las cadenas de distribución fue uno de los objetivos iniciales y específicos de los sistemas RFID. La sustitución de la identificación basada en códigos por etiquetas de tipo RFID y su uso dentro de las organizaciones, para hacer llegar cualquier bien de consumo desde su origen hasta su destin,o ha sido una de las líneas estratégicas de cualquier compañía, tanto en sus instalaciones de origen como en los puntos de venta finales.

- Servicios públicos y seguridad. Los servicios públicos han empleado los sistemas RFID tanto en la gestión de las infraestructuras públicas como para ayudar a los sistemas que ponen a disposición de los ciudadanos. En este apartado también podemos incluir la referencia a la identificación personal, el e-passport [33], basado en RFID es una realidad en los países más avanzados y permite realizar la identificación y todos los procesos asociados de cada persona.

Como última aplicación de los sistemas RFID y, como se comenta en el siguiente apartado, es necesario referir el concepto de redes de sensores dinámicas, como elemento fundamental en el desarrollo del IoT. Estas redes de sensores dinámicas aplican la tecnología RFID para dar soporte al desarrollo de redes con capacidad de sustituir los mecanismos de identificación de las redes tradicionales por comunicaciones que se basan en la identificación de las cosas.

2.3 Redes de sensores

El segundo elemento relevante del IoT son los sistemas de comunicación que deben dar soporte a las cosas y sus relaciones. Además de la capacidad de identificar los objetos, el IoT debe integrar todo tipo de dispositivos de adquisición de información y actuación sobre el mundo real. Para resolver esta integración es necesario disponer de mecanismos que faciliten las relaciones entre los objetos y la distribución de la información. En la actualidad esta conexión se resuelve con el uso de Internet como plataforma que permite la distribución de la información de las cosas. Pero las exigencias tanto en cuanto a la naturaleza de las comunicaciones como en cuanto a la magnitud del tráfico que se puede desencadenar están dirigiendo las propuestas hacia un nuevo modelo de red que se integre en Internet, como las redes M2M [34] o las redes dinámicas de sensores [35].

2.3.1 Redes de sensores

Las redes de sensores son un concepto relativamente nuevo que nos sirve para permitirnos describir la infraestructura de conexión de elementos que contienen algún tipo de capacidad sensitiva automática, más algún sistema que permite la explotación de ese ese conocimiento sensorial.

Los sistemas más tradicionales consisten en redes propietarias cerradas y específicas [36] que permiten la conexión a través de alguna red de carácter genérica con un uso específico ad-hoc como por ejemplo Zigbee, Ethernet, Z-Wave, etc. Entre las características más comunes de las redes de sensores podemos comentar:

- Topología. Suele manejarse topologías dinámicas y adaptables al entorno donde adquieren información los sensores.

- Heterogeneidad. Los elementos que forman parte de la red pueden realizar funciones variadas. Por ejemplo, puede ser necesario que un nodo que adquiera información no sólo la transmita al sistema de proceso sensorial sino que ademas tenga que retransmitir la de otros nodos a modo de *gateway*.

- Tolerancia a errores. Deben soportar mecanismos para asegurar que ante fallos de transmisión la información adquirida de los sensores no se pierda, ya sea por mecanismos de almacenamiento local o retransmisión.

- Bajo coste. Tanto referido al consumo del energía como al mantenimiento que sea necesario.

A medida que se ha extendido el concepto de las redes de sensores, los campos de aplicación han aumentado de igual forma, sobre todo dirigiéndose a su uso en la fusión sensorial de información [37]. La fusión de sensorial se define como un proceso de múltiples etapas automáticas para asociar, correlar, estimar y combinar datos de múltiples fuentes simples de datos para obtener una información de carácter complejo [38].

Las redes de sensores están integradas por múltiples nodos que pueden organizarse de diversas formas. El estudio de los trabajos relacionados muestra tres tendencias principales: distribuida, descentralizada y centralizada (2.4). Las arquitecturas centralizadas proponen transmitir a un único nodo (central) la información de los sensores que realizan las operaciones de recopilación de la información desde los sensores. Existen múltiples soluciones de este tipo de redes con diferentes aproximaciones como por ejemplo, orientadas a la eficiencia y robusted [39], orientadas al procesamiento paralelo [40], o los sistemas independientes del hardware [41]. Las arquitecturas descentralizadas incluyen varios nodos que pueden hacer el proceso de adquisición de la información de los sensores. Estas arquitecturas permiten reducir las debilidades de las aproximaciones centralizadas mejorando tanto la robustez como la calidad de la información [42], pero incluyen complejidad a la hora de evitar doble proceso o consolidar la información. Por último, las arquitecturas de redes distribuidas reparten el proceso en varios nodos de la red, de esta forma se conforman sistemas en los que los nodos realizan partes específicas del proceso de adquisición distribuyendo las operaciones necesarias. Según la forma de distribuir la información de los sensores aparecen sistemas basados en mensajes [43], basados en difusión [44], etc.

En la actualidad y vinculado directamente con el IoT, las redes de sensores se refieren de forma más específica a las redes de sensores inalámbricas (WSN: *Wireless Sensor Networks*) para referir a aquellas redes de sensores en las que la conexión

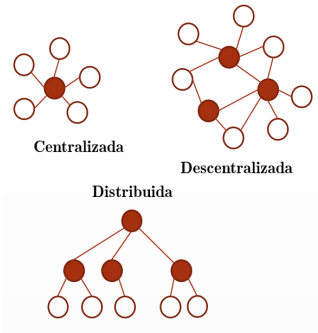

Figura 2.4: Tipos de redes de sensores.
Tipologías de red más habituales

entre los nodos se realiza por una comunicación inalámbrica. El uso de las WSN se ha extendido básicamente por el bajo coste y bajo consumo, y que pueden integrarse de una forma sencilla junto a los sensores, para formar redes inalámbricas.

2.3.2 EPCglobal Architecture Framework

El marco EPCglobal es una colección de estándares de hardware, software e interfaces. El marco EPCglobal [45] ha sido una de las propuestas más relevantes de los AutoID. La red consiste en varios elementos que conforman una red distribuida basada en la identificación de tipo EPC: las etiquetas, los lectores, una capa de *middleware* que abstrae el hardware de los lectores y maneja información, y un sistema de distribución de la información basada en servicio denominada EPCIS [46] (*EPC Information System*). El servicio EPCIS proporciona a los usuarios una forma de acceder a la información EPC a través de un interface único. El enlace entre la identificación EPC y la información que se distribuye se resuelve con dos servicios adicionales: el ONS (*Object Name Service*) y el EPCIS (*EPC Information Systems*) Discovery. El primer servicio se utiliza para identificar los nombres de los códigos de auto identificación y el segundo para determinar e identificar las conexiones entre dos servicios EPCIS de diferentes organizaciones que comparten información.

La relevancia de esta propuesta reside en haber sido la primera solución de arquitectura completa para el IoT. La propuesta original planteaba el funcionamiento de una cadena de distribución desde la fábrica hasta el consumidor final utilizando

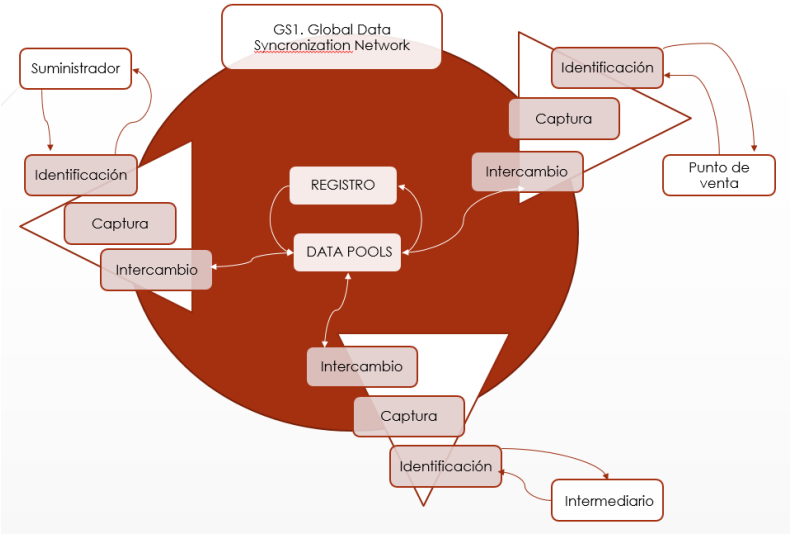

Figura 2.5: EPCGlobal.
Intercambios del EPCglobal

las etiquetas EPC como identificador para intercambiar la información entre los actores de la distribución y la comercialización, con los intermediarios que fuera necesario. La versión actual, distingue tres niveles de organización de los estándares: la identificación, la captura de información y la capa de intercambio; y tres tipos de servicios: de asignación, de registro y de recuperación. La capa de identificación incluye los estándares relacionados con las etiquetas tanto de tipo RFID EPC como del resto de mecanismos de identificación soportados en el GS1 como por ejemplo códigos de barras (EAN/UPC, ITF14, GS1-128, ..) y EBT (*Electronic Business Transactions*) (EANCM, eCom, ..). La capa de captura incluye los estándares relacionados con la recogida de información como son todos los relacionados con la captura por RFID: LLRP (*Low Level Reader Protocolo*, RP (*Reader Protocol*), el RM (*Reader Management*), el ALE (*Application Level Events*) y el DCI (*Discovery Configuration Initialization Interface*). El nivel de intercambio recoge las propuestas de intercambio a través de la denominada GDSN (*Global Data Syncronization Network*) que es la red en la que las organizaciones comparten la información que intercambian. Incluye dos elementos fundamentales: el registro de datos de la red en donde se almacenan la información de las organizaciones y los *pools* de datos de registros de entrada y salida que generar las fuentes de productos

etiquetados y que consumen las organizaciones de comercialización.

El gran problema que ha limitado la evolución de la EPCglobal ha sido la complejidad de elementos que conforman la red y que hace que pequeños desarrollos que se quieran integrar en la red exijan una gran cantidad de esfuerzo de desarrollo. Por esta limitación, la propuesta ha quedado restringida a organizaciones del grupo GS1 y grandes compañías de desarrollo de software, que han podido construir en parte las costosas exigencias de la arquitectura.

2.3.3 Visión vertical del IoT

Las redes necesarias para dar soporte al IoT pueden modelarse desde dos puntos de vista: vertical y horizontal. La visión horizontal se centra en el despliegue y gestión de la información, mientras que la aproximación vertical se centra en la resolución de las capas necesarias para la implementación. Como se comenta en [34], una visión vertical del IoT se puede plantear en tres niveles: el nivel sensorial, el nivel de red, y el nivel de aplicación.

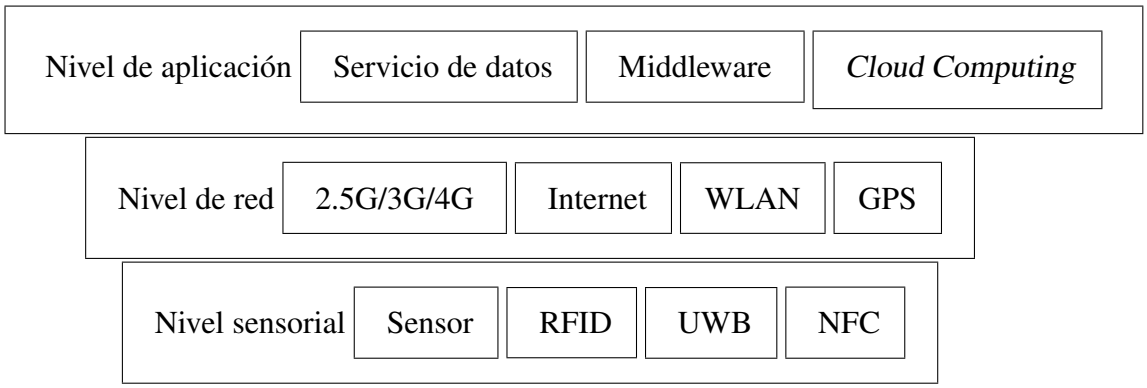

Figura 2.6: Visión vertical del IoT.

Ejemplo de elementos para una red del IoT

El nivel sensorial incluye básicamente dos funciones: primero, adquirir información o actuar con la información, y segundo compartir esta información. El estado de las cosas o los eventos asociados , como por ejemplo la temperatura, el color o sonidos del mundo real son adquiridos por dispositivos sensores. Los avances en estos sensores se centran en disminuir tanto el tamaño, como su precio, como su consumo, a la vez que se incrementa su capacidad sensorial y de proceso. Y estos

avances han dado lugar a la realización de las redes de sensores comentadas en el apartado anterior. Sin embargo otros temas se encuentran todavía en desarrollo como el uso de conexiones de corto alcance como NFC (*Near Field Communication* y UWB (*Ultra-Wide Band*), la automatización de la instalación de los sensores, conseguir la integración automática entre sensores y el nivel de red, o aumentar la capacidad de expansión de los sensores.

El nivel de red se encarga del transporte de los datos sensoriales hasta las aplicaciones. Es la capa más conocida puesto que en la actualidad se está empleando toda la infraestructura existente de Internet sin diferenciar su aplicación. Sin embargo las necesidades del IoT exigen resolver cuestiones adicionales como son el direccionamiento de las cosas, la integración de los dispositivos y las redes, y la gestión de los recursos de las redes. El direccionamiento tradicional basado en el IPv4 se está migrando al IPv6, pero no parece que sea el direccionamiento completo que puede resolver el direccionamiento del IoT, por lo que están surgiendo nuevas ideas como la comentada de los sensores con direccionamiento. La integración de dispositivos debe resolver sobre todo el funcionamiento de una red en tiempo real tan extensa como la necesaria para integrar los sensores y los actuadores del IoT y con la mezcla de tipos de red. Y por último, la gestión de los recursos de la red con las características de datos masivos y dispersos que deben procesarse, transmitirse o almacenarse.

La capa de aplicación incluye una capa de soporte al nivel de red y una capa de servicio. La capa de soporte incluye tecnologías distribuidas como P2P (*Peer To Peer*) o *cloud computing* y resuelve funciones de ocultación de la capa de red, búsqueda de dispositivos, direccionamiento, enrutamiento, control de la calidad de la información, etc. A partir de esta capa, la subcapa de servicio proporciona interfaces de acceso para extender la información y para poder integrarse en otros sistemas.

2.3.4 Redes híbridas: RFID y sensores

Las redes de RFID permiten adquirir información sobre la identificación de las cosas, mientras que las redes de sensores nos permiten adquirir información sobre las propiedades de las cosas. Las dos funciones tienen un valor importante en el

IoT, pero la unión de las dos adquisiciones es la verdadera representación virtual del mundo físico.

Un primer intento de sacar provecho de las redes RFID ha sido incorporar a la identificación el dónde y el cuándo, obtenidos de la propia semántica de la acción de lectura de una etiqueta en un dispositivo lector. Esto ha llevado a un error típico en el que parecía que los sistemas de RFID podían saber dónde estaban los objetos etiquetados en cualquier momento, cuando lo correcto ha sido que podíamos saber dónde y cuándo estaba un objeto siempre que fuese leído. Para algunos sistemas esta capacidad de incluir la tripleta quién, dónde y cuándo era más que suficiente. Sin embargo para otros sistemas con esta información no es suficiente, y así surge el concepto de "tag como sensor", que no es otra cosa que un sensor en el que se ha integrado una etiqueta RFID. A partir de este punto se han adoptados dos soluciones. La primera es que el sistema de adquisición del sensor ha integrado la información de autoidentificación. Y la segunda es que el sistema de adquisición RFID ha integrado la información del sensor. En este segundo caso, nos referimos a lectores RFID con capacidad sensorial, como por ejemplo lector Mini M1 [47] de SkyeTek. Al extender este concepto de "lector como sensor", hablamos de redes híbridas de sensores con RFID [48].

Las posibilidades al unir el mundo sensorial con la identificación automática generan el núcleo fundamental del IoT pero a su vez plantean nuevas cuestiones sobre la forma en la que podemos unir la identificación con los datos percibidos, las limitaciones de las lecturas por RFID, tanto en consumo como en tiempo de exposición de las etiquetas sensores. Un ejemplo de aplicación puede verse en [49] o en [50]. En ambos casos, la aproximación para resolver la unión del RFID y la información sensorial pasa por la definición de una capa de coordinación que se encarga de sincronizar las dos informaciones.

Otra aplicación alternativa de la identificación integrada con los sensores son las redes móviles dinámicas de sensores. Estas redes están formadas por sensores móviles identificados, que transmiten su información sensorial a un *gateway* también móvil que se encarga de conectar con las infraestructuras fijas. El "role" de *gateway* móvil también resuelve en caso necesario la información que permite descubrir o aportar información sobre algún identificador de sensor que aparezca en la red móvil.

2.4 Heterogeneidad del IoT

Una de las primeras cuestiones que surge a la hora de resolver cualquier aplicación para el IoT es la gestión de la naturaleza heterogénea de los elementos que lo conforman. Esta heterogeneidad se encuentra tanto en el entorno físico que se intenta maneja, como en los recursos que utilizamos en el entorno virtual. La forma de representar la heterogeneidad y cómo resolver la interacción entre las "cosas"de diferente naturaleza son dos de los aspectos fundamentales de la modelización del IoT.

2.4.1 Heterogeneidad

Para que se extienda el uso del IoT, no sólo es necesario resolver los modelos de las redes que permitan la conexión de las cosas con el mundo virtual en donde se emplea su información, sino que también es necesario resolver la forma en la que las cosas pueden interactuar entre ellas mismas. En la actualidad los entornos propios y cerrados de fabricantes según los tipos de aplicaciones que desarrollan, con características y capacidades diferentes dificultan cualquier operación que permita relacionar cosas de un entorno con otro [51]. Esta multiplicidad de aspectos que caracteriza el IoT afecta a la hora de integrar y relacionar "cosas" que siendo iguales en el entorno real, sin embargo, se presentan de diferentes formas en los entornos virtuales. Y esta diversidad, no sólo afecta al tratamiento en el IoT, sino que además se traslada a la hora de integrar la información generada a otros sistemas o a la hora de integrar la información de otros sistemas en el IoT [52].

Además, la heterogeneidad no ha hecho sino comenzar, la variedad de opciones y la propia naturaleza de los elementos que conforman el IoT, hace que cada vez más la variabilidad y diversidad de elementos se incremente. Incluso con el aumento de las capacidades sensoriales, de actuación o de proceso de las "cosas", la variabilidad se ha trasladado a las propias cosas, que en unos casos tienen unas capacidades y en otros otras según el entorno en el que se encuentran. E incluso, en algunos casos, puede suceder que una capacidad o función se encuentre disponible en algún momento de tiempo y no en otros, por lo que cualquier aplicación que haga uso o espere hacer uso de ese recurso, puede encontrar que no está disponible. En este entorno tan dinámico es necesario proporcionar representaciones de alto

nivel que faciliten de alguna forma la construcción de los sistemas que gestionen y modelen esta heterogeneidad.

2.4.2 Representación de la heterogeneidad

Se han propuesto diversas formas de representar el entorno real en un entorno virtual y poder integrar las capacidades de las cosas en funciones virtuales y abstractas. Básicamente, estas propuestas pueden agruparse en tres aproximaciones genéricas y complementarias relacionadas con la forma en que se va a representar la heterogeneidad. La primera es utilizar una representación uniforme de las "cosas" y sus capacidades de forma que pueda extraerse información de esta representación. La segunda consiste en encapsular las capacidades de las "cosas" en una abstracción. Y la tercera sería ocultar las cosas a través de un metaprotocolo de comunicaciones que resolviera el contacto con/entre las "cosas" para uniformizar la gestión de sus capacidades.

En cualquiera de las tres opciones anteriores, la representación del IoT se ha realizado utilizando ontologías[1] como mecanismo para modelar sus elementos y las relaciones entre ellos. La mayor parte de las representaciones existentes del IoT han utilizado mecanismos que están basados y heredados de las ontologías asociadas a representaciones de la redes de sensores, tipo WSN, como por ejemplo las propuestas de [54] y [55]. Estas propuestas basadas en esquemas XML se han extendido para dar lugar a ontologías con semántica particular del IoT, en particular sobre las "cosas" y sus capacidades. Como ejemplo de este tipo de representación del IoT se pueden comentar *Sense2Web* [56] y *IoT-lite* [57].

Sense2Web es una plataforma que proporciona mecanismos de publicación de datos de sensores. Permite la publicación de los datos de descripción de sensores basada en tripletas RDF (*Resource Description Framework*) formadas por sujeto, predicado y objeto, usar la descripción de otros recursos RDF, enlazar con otros repositorios públicos disponibles y poner a disposición de aplicaciones cliente los datos de los sensores como accesibles desde SPARQL (*Simple Protocol and RFD Query Language*).

[1] representación formal y explícita de una abstracción [53]

IoTlite es una ontología reducida que intenta describir el IoT con tres clases de conceptos: los objetos, los recursos y los servicios. Permite establecer clasificaciones de los elementos que forman parte del IoT y describir tanto unidades, tipos y magnitudes de los objetos.

Sin embargo, estas representaciones no resuelven fácilmente la heterogeneidad del IoT al centrarse en un limitado conjunto de elementos y no proporcionar elementos de abstracción mayores que oculten las diferentes capacidades de las "cosas".

2.4.3 Abstracción de las cosas

Las abstracción de las cosas y sus capacidades, es uno de los elementos de investigación fundamentales en el IoT. Las propuestas existentes han utilizado el paradigma SOC (*Service-Oriented Computing*) que permite abstraer las cosas en términos de servicios que se mantengan desacoplados de las peculiaridades heterogéneas. Con esta aproximación podemos seguir una de las dos opciones siguientes: abstraer las "cosas" físicas como servicios o abstraer las "cosas" virtuales como servicios.

En el caso de la abstracción de las "cosas"físicas como servicios disponemos de los sensores/actuadores directamente. La información que se recibe o que se necesita enviar se maneja sin procesos ni de integración ni de agregación. Los sensores son tratados como servidores. Y los actuadores como consumidores. Las aplicaciones que se encargan de su gestión son las que realizan cualquier tipo de inteligencia. En este caso, el IoT se convierte en el WoT [58].

En el caso de la abstracción de las "cosas" virtuales como servicios, los datos quedan desacoplados de los recursos físicos que los generan. En las "cosas" virtuales los datos han sido procesados y agregados en varios pasos y pueden ser resultado de uno o varios orígenes. Y de esta forma se aporta el beneficio a los clientes de los servicios que pueden manejar muchas "cosas" virtuales al mismo tiempo y sin necesidad de preocuparse de la heterogeneidad que las caracterice. El término de "cosas" virtuales aparece como extensión del término "sensores" virtuales, referido ya en [59] .

2.4.4 Interacción de las cosas

Una vez que tenemos los datos representados y las capacidades uniformizadas con servicios todavía se nos plantea un elemento más de heterogeneidad en el IoT que es cómo descubrir y acceder a los servicios. En este caso se pueden seguir dos soluciones: una es considerar que los servicios son de acceso directo y otra es considerar que la interacción es remota a través de Internet.

Si el acceso es directo, pueden utilizarse los diferentes protocolos de red habituales (p.e. Bluetooth, uPnP, ZigBee) en la conexión de redes de sensores. Incluso si es necesario en redes híbridas se puede utilizar un *middleware* que permita ocultar la heterogeneidad de protocolos. Este *middleware* puede extraer la información necesaria de cada protocolo y ocultarla con su descripción propia.

Si el acceso se realiza a través de Internet, además de los servicios basados en SOAP o en REST, se pueden utilizar otros mecanismos más ligeros , como por ejemplo los que se han desarrollado en el marco de los estándares del OMA (*Open Mobile Alliance)*, el COAP (*Constrained Application Protocol* que permite sustituir el habitual HTTP/TCP de los servicios web de Internet por el COAP/UDP. Otra solución que también se está extendiendo en el ámbito del IPv6, es el I6LowPAN, una versión reducida del IPv6 para dispositivos con poca capacidad.

2.4.5 El IoT Industrial

Entre la heterogeneidad del IoT, en los últimos años ha surgido el concepto de I-IoT o el Internet de las Cosas en la Industria [60] para intentar establecer la arquitectura de un nuevo paradigma que sirva para unir el entorno de las TI (Tecnologías de la Información) y las TO (Tecnologías de la Operación) con el IoT. Se trata en definitiva, de aprovechar las ventajas que se pueden obtener del IoT para incluir información en los diferentes sistemas de información de una organización: planificación de recursos, gestión de clientes o sistemas de ayuda a la decisión; y también en los sistemas operativos como: sistemas de control, sistemas de control de la producción o líneas de producción.

Las bases del I-IoT se establecen sobre: computación dirigida por los sensores, los sistemas de análisis, y las máquinas inteligentes. La computación dirigida por sensores convierte los datos de los sensores incluidos en los objetos en información

relevante para los sistemas. Las características de los sensores, con la reducción de tamaño, la reducción de precio y el aumento de sus capacidades ha producido y continuará produciendo un cambio significativo en la industria. Por ejemplo, en el año 2007 el precio de un sensor acelerómetro era de 3$, en el año 2014 tiene un precio de 54 cent., y ya disponemos de sensores que no tienen baterías, y mecanismos de comunicación de bajo consumo que permiten la interconexión entre objetos sin necesidad de sistemas de mayor jerarquía. Todos estos recursos generan datos que deben procesarse para poder crear información que se puede manejar por otros sistemas o por usuarios. Por ejemplo, la compañía Caterpillar ha introducido en sus máquinas un sistemas para recopilar la información sobre su funcionamiento, y a través de un sistema de análisis ser capaces de anticipar problemas o programar sistemas de mantenimiento proactivos que permiten evitar problemas de mayor coste. El último punto del I-IoT son las máquinas inteligentes. Igual que se han incorporado sistemas de todo tipo en los vehículos, las máquinas de la siguiente generación incorporarán a su mecánica específica todo tipo de sensores y sistemas software para procesar los datos y poder modificar su funcionamiento. Incluso tendrán la capacidad de comunicarse con otros dispositivos para poder ofrecer información sobre su funcionamiento o recibir información para determinar como cambiar su comportamiento.

En estos momentos, las plataformas del I-IoT todavía están en sus fases iniciales de desarrollo. Y aunque existen propuestas genéricas como las del IIoT Consortium, el Open Interconnect Consortia, o la AllSeen Alliance, continua siendo necesario buscar propuestas que faciliten el desarrollo y la evolución de aplicaciones dentro de este área.

2.5 Desarrollo de aplicaciones para el IoT

El proceso de construcción de aplicaciones para el IoT tiene características particulares que plantean diferencias con las metodologías habituales de desarrollo.. La primera característica está relacionada con el apartado anterior, la heterogeneidad del entorno y del propio IoT. Como hemos comentado, la heterogeneidad aplica tanto a la escala del problema, aplicaciones "pequeñas" y aplicaciones de "gran" tamaño y funcionalidad, como a las características de los sistemas de conexión, o del propio hardware que debemos manejar. Y la segunda característica es la falta de abstracciones de alto nivel que faciliten la realización de aplicaciones, no existen aportaciones significativas que permitan modelar aspectos del IoT. Las aportaciones actuales presentan aplicaciones y objetivos particulares sobre partes del sistema o sobre situaciones muy específicas. En los siguientes apartados se presentan algunas conclusiones sobre los trabajos actuales y que justifican la propuesta realizada en el presente trabajo.

2.5.1 Ejemplo de aplicación

Para comentar las características particulares del desarrollo de aplicaciones en el IoT vamos a utilizar un ejemplo que ponga de manifiesto las cuestiones que surgen. El ejemplo que vamos a comentar pertenece al ámbito de la domótica pero vamos a plantearlo con un objetivo algo más específico.

Supongamos que una organización dispone mayoritariamente de tele-trabajadores y está pensando en habilitar en una nueva sede la infraestructura para atender a este tipo de trabajadores que sólo acuden de forma extraordinaria al centro de trabajo. Por este motivo, la organización ha reservado "n" edificios, con "m" plantas cada uno y "p" salas por planta, para que sean utilizadas por los teletrabajadores que quieren acudir al centro de trabajo (podría corresponderse con la representación de la figura 2.7).

Entre los elementos instalados se tienen que manejar distintos tipos de sensores, actuadores y gestionar la información de los distintos edificios a través de un sistema de control y supervisión. Para mejorar la eficiencia energética el sistema debe encargarse también cada día de asignar los espacios a los teletrabajadores que

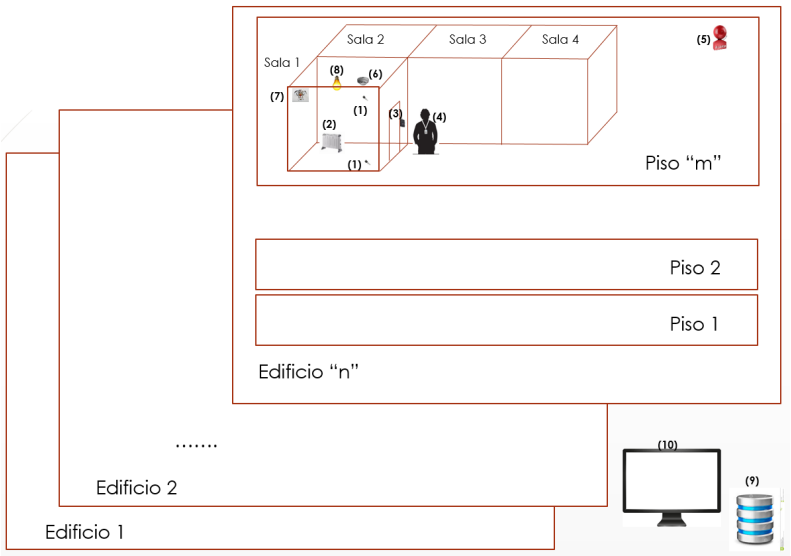

Figura 2.7: (1) sensores de temperatura, (2) calefacción, (3) lector RFID, (4) etiqueta RFID, (5) alarma, (6) sensor de humos, (7) aspersor antiincendios, (8) punto de luz, (9) discos de datos, (10) sistema monitor

Esquema del problema.

acuden a las oficinas, de acuerdo a un perfil personal de cada empleado y a las condiciones ambientales. Los lectores RFID de corto alcance deben indicar, el acceso a los edificios de un empleado y a partir de este acceso dirigirlo hacia la mejor posición de acuerdo a sus preferencias y al estado del edificio. Por supuesto, además el sistema debe permitir en todo momento que los trabajadores o los operarios del control modifiquen las condiciones a través de los actuadores existentes.

2.5.2 Características de los desarrollos para el IoT

El análisis del problema anterior muestra una serie de características propias del desarrollo de las aplicaciones del IoT como son:

- **Repetición de aspectos comunes en varios niveles.** Un primer análisis podría pensar en un sistema único que pudiese instalarse en diferentes organizaciones con el mismo tipo de problema para la gestión de sus oficinas. Tal vez, incluso podría utilizarse la aplicación de forma común, pero en un problema diferente, en lugar de la gestión de los sitios de los teletrabajadores, para

otra gestión. En cualquiera de los dos casos, aparece un dominio común (el de los edificios inteligentes) del problema que comparte buena parte de vocabulario de los objetos del problema (termostato, lector RFID, etc.), de los atributos que se manejan (identificación del trabajador, temperatura, luminosidad, humedad) e incluso en las entidades de interés del problema (sala, planta, edificio). De esta forma, podemos considerar una primera opción de abstraer las aplicaciones del IoT en esos dominios particulares formados por elementos comunes.

- **Múltiples aspectos.** Podemos detectar varios aspectos en las aplicaciones para el IoT. Primero, el dominio específico para el que se desarrolla la aplicación, en el ejemplo: el control de los edificios inteligentes. Segundo, operaciones específicas de las aplicaciones como por ejemplo en el problema comentado; la regulación de la temperatura y la detección de incendios. La tercera, las operaciones específicas del sistema, por ejemplo usar APIs del sistema Android para obtener datos de los sensores de temperatura del edificio. Y cuarta, operaciones específicas de la instalación, como se trata en este caso al tener que dirigir a los teletrabajadores hacia un puesto según las condiciones de un determinado edificio.

- **Dispositivos heterogéneos.** La aplicación debe tener que manejar distintos tipos de conexión y distintos tipos de dispositivos. En el problema planteado, aparecen distintos tipos de sensores y actuadores, distintos tipos de interfaces, el sistema de almacenamiento y todos ellos conectados de una u otra forma con el sistema de supervisión y control.

- **Plataformas heterogéneas.** Los entornos típicos del IoT incluyen casi inexcusablemente cuestiones multi-plataforma. Nuestro ejemplo, emplea algún tipo de plataforma para resolver la identificación del acceso, otro entorno para la supervisión y control e incluso puede emplearse alguna plataforma adicional para la gestión de los perfiles de los trabajadores, el mantenimiento de los dispositivos domóticos, etc. Las aplicaciones que se construyen debe plantearse en términos de independencia de la plataforma y con mecanismos

que permitan resolver las funciones necesarias sin tener que depender de los recursos particulares de cada una de ellas.

- **Formas de interacción heterogéneas.** Los diferentes tipos de dispositivos que aparecen en el entorno tienen distintas capacidades y distintas formas de permitir el acceso a la información que reciben o sobre la que actuan. Los modos de interacción con la información de los sensores pueden ser: suscripción/publicación, petición/respuesta, cliente/servidor, comando, excepciones, broadcast, etc. En el ejemplo, el lector de RFID puede notificar la detección de la identificación y la identificación a aquellos equipos que estén suscritos a sus servicios, y los calefactores de las salas pueden recibir los comandos de "poner temperatura" de forma directa.

- **Evolución.** Las aplicaciones que se desarrollan para el IoT deben contemplar la facilidad para incorporar cambios en los requisitos tanto funcionales como no funcionales. Por ejemplo, en nuestro caso puede surgir la necesidad de integrar otro tipo de trabajadores distintos a los teletrabajadores que pases a utilizar las instalaciones, o por cuestiones de mantenimiento puede ser necesario modificar los dispositivos relacionados con el sistema anti-incendios. En definitiva, los cambios tecnológicos y la extensión de los sistemas es algo intrínseco a cualquier aplicación del IoT.

- **Escala.** La última característica que queremos comentar de las aplicaciones del IoT es la naturaleza multiescala de los problemas que se plantean. Las aplicaciones multi-escala refieren a sistemas en los que se mezclan aplicaciones de distinto tamaño y diferente ámbito [61], [62]. En estas aplicaciones aparecen problemas relacionados con la distribución de la información, la seguridad de los datos en los diferentes entornos, la distribución de las capacidades de cómputo, gestión de la mobilidad de los equipos, evolución de los dispositivos, incorporación y mantenimiento de los equipos, etc.

2.5.3 Problemática del desarrollo de aplicaciones en el IoT

Una vez caracterizadas las aplicaciones en el IoT podemos describir las cuestiones que surgen para el desarrollo de este tipo de sistemas. Entre los principales pro-

blemas del proceso de desarrollo podemos comentar la falta de identificación de los roles necesarios para construir este tipo de sistemas, la heterogeneidad de los sistemas, los diferentes ciclos de vida que pueden producirse en la aplicaciones, la falta de recursos específicos para estos desarrollos y la heterogeneidad comentada en los apartados anteriores.

El desarrollo de las aplicaciones del IoT implica el conocimiento de múltiples aspectos que necesitan integrarse. Este proceso involucra a expertos que necesitan un perfil multi-disciplinar que permita resolver el conjunto de cuestiones que pueden aparecer. Esta falta de identificación de los roles necesarios plantea un conflicto a la hora de establecer quién y cómo deben resolverse determinadas funciones (en el ejemplo del apartado de quién es la responsabilidad de la parte de la aplicación del servidor que da soporte a la supervisión y control, o cae en el instalador de los sistemas anti-incendios). Y estos conflictos pueden extenderse a otras partes del proceso como definir los dominios de competencia de cada parte del desarrollo, las responsabilidades en las instalaciones o el mismo proceso de diseño e implementación [63].

Las fases tradicionales de los ciclo de vida de las aplicaciones incluyen tareas claramente definidas en una u otra organización. Sin embargo en el desarrollo de aplicaciones para el IoT, estas tareas se entremezclan en las diferentes fases: el desarrollo, la instalación y el mantenimiento, se encuentran a menudo superpuestas ante la imposibilidad de realizarse de forma separada. La característica evolutiva de las aplicaciones lleva a que la fase de desarrollo se realice de forma simultánea en muchos casos a la instalación por la imposibilidad de realizar cualquier tipo de desarrollo sin la necesaria infraestructura en funcionamiento, y de igual forma el mantenimiento en sí mismo exige requisitos de desarrollo que faciliten de alguna forma una automatización mínima de esta fase.

Por último, las aplicaciones en el IoT actual se plantean en términos de dominios de aplicación particulares, por ejemplo la domótica en el caso del ejemplo de este apartado, en los que aparecen funcionalidades específicas de ese dominio (p.ej. el sistema anti-incendios, el sistema de calefacción, etc.). Pese a esta especialización de los dominios, los lenguajes que se utilizan, como por ejemplo PervML [64] se proponen como extensiones de elementos de propósito general relacionado con

las notación UML, con las ventajas que existen en estos lenguajes, pero sin los elementos específicos para ayudar a resolver aplicaciones de este dominio.

2.5.4 Soluciones para el desarrollo del IoT

El desarrollo actual de aplicaciones para el IoT se realiza fundamentalmente utilizando la programación orientada al nodo, en la que los expertos en sistemas embebidos o en sistemas distribuidos resuelven las operaciones de cada dispositivo por separado. Estas operaciones se manejan de forma explícita y se realizan las interacciones entre dispositivos a través de mecanismos ad-hoc específicos en cada solución. En este caso, disponemos de los nodos como centro de las aplicaciones y se agregan las informaciones desde el resto de eventos disponibles, como por ejemplo la identificación o la posición.

Una alternativa que se usa para el desarrollo de las aplicaciones es la aproximación basada en macro-programación. Este tipo de resolución de las aplicaciones tiene su origen en las soluciones de existentes que implementan redes de sensores. En este tipo de sistemas, las aplicaciones se diseñan de forma global, centrando los objetivos del sistema completo en lugar de centrarse en los nodos que lo forman y para ello se utilizan mecanismos de abstracción que permiten indicar cómo relacionar los nodos.

La aproximación basada en los nodos tiene las ventajas de obtener sistemas muy eficientes y poder emplear técnicas de desarrollo habituales con diseños, lenguajes y plataformas de orientación genérica. Sin embargo tiene dificultades a la hora de resolver los aspectos heterogéneos del IoT y encontrar una solución para la integración de todos los dispositivos. En el caso de la resolución con macro-programación las ventajas que encontramos nos permiten describir el funcionamiento de la aplicación de forma global, pero necesitamos disponer de mecanismos de traducción desde las macros de alto nivel a las aplicaciones particulares de cada dispositivo o plataforma de red.

La aproximación seguida en este trabajo, modelo y lenguaje de dominio propone una solución para el desarrollo de aplicaciones del IoT en el marco de los desarrollos basados en macro-programación. La utilización del modelo facilita la

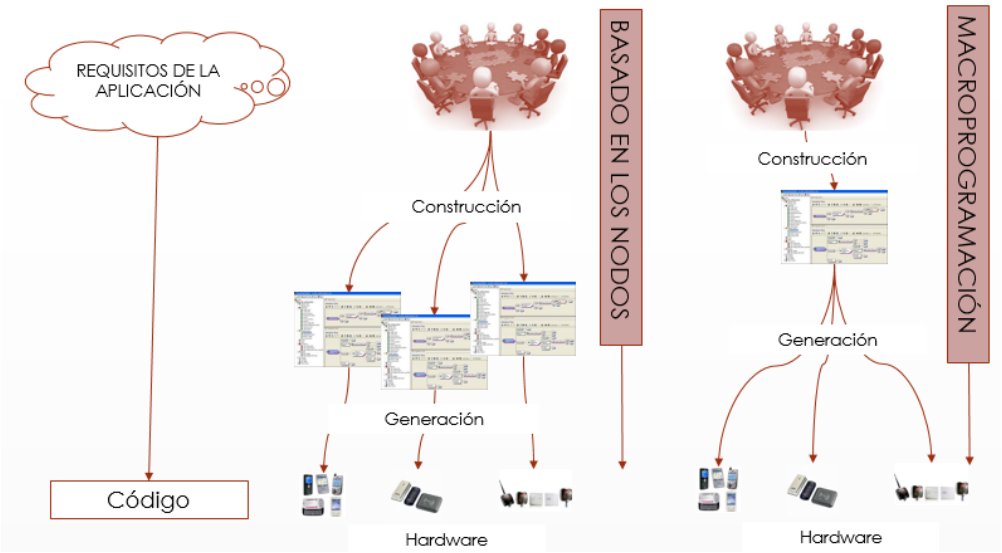

Figura 2.8: Procesos de desarrollo del IoT.

Aproximación basada en los nodos y macroprogramación

abstracción de los problemas planteados en el marco heterogéneo del IoT y el lenguaje de dominio sirve de herramienta para hacer posible la construcción de los sistemas.

Todo del mundo visible es una representación del mundo invisible

Dan Petersen

CAPÍTULO

3

metaDEPCAS

El modelado del IoT de los próximos años debe proporcionar herramientas que permitan la integración de las tecnologías existentes. La identificación automática basada en radiofrecuencia integrada en los sensores distribuidos por el mundo real parece que puede ser la base de cualquiera de las aplicaciones que se utilicen en el ámbito IoT. Sin embargo, el punto de partida actual, difiere bastante de esta integración puesto que los dos conceptos se han aplicado por separado en diferentes dominios. Mientras que las redes de sensores se utilizaban en la ámbitos domóticos y robóticos, la identificación automática quedaba relacionada con cadenas de suministro o herramientas específicas de identificación. Estas diferencias han llevado a líneas de investigación y propuestas diferentes, específicas de cada ámbito, sin que existan aportaciones relevantes que incorporen una integración real de las prestaciones y capacidades de las dos tecnologías.

El modelo MetaDEPCAS es una propuesta para la integración de datos de sensores con identificación por RFID, basado en un modelo de referencia que define un dominio común sobre el IoT; una arquitectura de referencia que define el marco común de desarrollo de aplicaciones y un sistema de acceso a la información basada en la arquitectura propuesta en el middleware DEPCAS. Estos tres elementos:

modelo de referencia, arquitectura de referencia y sistema de adquisición, proporcionan un mecanismo de modelado de sistemas que integran la información de los sistemas sensoriales junto a la identificación basada en RFID.

En los siguientes apartados se describen, primero: el modelo y la arquitectura de referencia de MetaDEPCAS, a continuación el sistema de adquisición DEPCAS y se finaliza la descripción con los mecanismos de integración de información RFID y sensores y la distribución de la información.

3.1 Modelado del IoT

3.1.1 metaDEPCAS

MetaDEPCAS es una propuesta para modelar los sistemas del IoT, basada en la integración de información sensorial de distintas fuentes, con un sistema de identificación por radiofrecuencia y con capacidad de adaptación distribuida a diferente escala. Esta definición extensa se concreta en tres elementos que permiten el uso de metaDEPCAS como herramienta de modelado: primero el modelo de referencia que define el nivel de abstracción de mayor nivel para la definición de las arquitecturas, segundo la arquitectura de referencia que permite disponer de los mecanismos abstractos que se utilizan en las aplicaciones concretas, y tercero el sistema de adquisición de la información basado en RFID que aporta la fuente de información de la arquitectura.

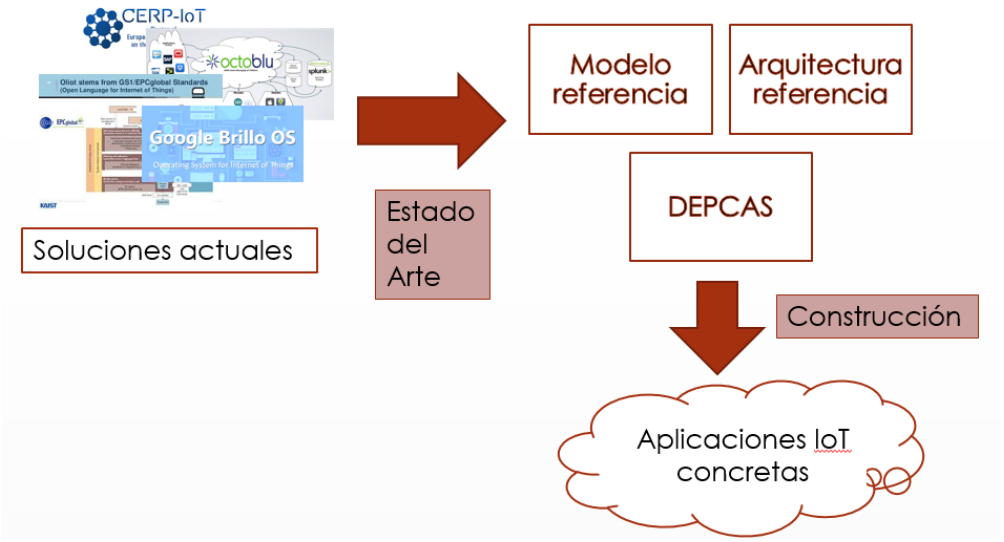

Figura 3.1: Elementos de metaDEPCAS.

Ámbito de aplicación de metaDEPCAS

El modelo de referencia junto con la arquitectura de referencia aportan los mecanismos de abstracción, que aplicados de forma guiada pueden utilizarse en la construcción de sistemas para el IoT. El resultado de esta aplicación deberá resolver las cuestiones de heterogeneidad y de interoperabilidad entre los sistemas

que estén basados en la misma arquitectura o derivada, aunque se construyan para dominios de aplicación diferentes.

3.1.1.1 Introducción

Como se ha comentado en el capítulo del estado del arte relacionado con diversos aspectos, el IoT actual incluye una amplia variedad de elementos y aplicaciones. La tendencia a agruparse en dominios específicos de aplicación ha aprovechado las tecnologías para resolver los aspectos específicos de cada uno de ellos, pero no ha facilitado que se resuelva la integración entre los sistemas o que se aprovechen los esfuerzos de diseño e implementación de unos sistemas en otros. Esto ha provocado, que el paradigma del IoT se encuentre limitado a las resoluciones particulares que se han implementado y en las que se ha podido obtener un éxito en la aplicación como sistemas de intra-net, pero sin llegar a formar parte de un inter-net de las cosas propiamente dicho, tal como se incluye en el acrónimo del IoT.

Para conseguir que las aplicaciones del IoT se relacionen, es necesario disponer de mecanismos que faciliten la integración de alto nivel entre los sistemas de diferentes dominios a partir de recursos comunes. Y esta es la idea fundamental de metaDEPCAS: proporcionar mecanismos de abstracción de sistemas del IoT que faciliten la integración de servicios e información, a partir de un nivel común que se pueda compartir entre los diferentes dominios. La aproximación comentada en [65] es la seguida para esta propuesta y está basada en los conceptos fundamentales de las arquitecturas software aplicando los conceptos de vistas y perspectivas. Según esta propuesta, podemos definir la arquitectura de un sistema software a partir de un modelo de referencia de la arquitectura (ARM: *Architecture Reference Model*), que está formado por el modelo de referencia y la arquitectura de referencia. A partir del ARM, y con la aplicación de las guías y las decisiones de diseño, podemos construir los sistemas concretos específicos de cada dominio. El contenido del ARM debe incluir aquellos elementos mínimos, que permiten dar soporte a las necesidades específicas de cada dominio y además, asegurar que existen los elementos necesarios que faciliten la creación de nuevos sistemas de diferentes características que sean interconectables.

3.1.1.2 Ventajas del uso del modelo de arquitectura de referencia

Las ventajas de usar una ARM para el modelado de las aplicaciones del IoT pueden concretarse en:

- *Asegurar la operatividad entre los sistemas del IoT.* El uso de un modelo de partida para construir cualquier sistema del IoT no asegura la interoperabilidad en cualquier caso, pero si que asegura disponer de una base que permita tener el conocimiento para distinguir qué decisiones de diseño o instanciación del modelo pueden provocar las diferencias. La comparación de estas decisiones de diseño que derivan dos sistemas desde el mismo modelo, pueden utilizarse para poder resolver las operaciones que faciliten el funcionamiento de los sistemas o incluso su integración.

- *Favorecer la evolución de los sistemas del IoT.* El entorno de IoT se caracteriza por su rápida evolución tecnológica y la heterogeneidad de los componentes. Al disponer del marco común podemos establecer un camino de partida y evolución que nos facilite incorporar las diferentes funciones de manera consistente y ordenada.

- *Disponer de una base.* El modelo de la arquitectura de referencia debe servir para tener una serie de puntos de conocimiento comunes que puedan comprender todas las personas que se involucran en la construcción del sistema para el IoT. Esta base común, difícil de conseguir, debe caracterizarse por tener el mínimo número de elementos comunes a los diferentes dominios de aplicación del IoT, a la vez que proporcione los mecanismos de flexibilidad necesarios para adaptarse a las particularidades de cada uno de ellos.

- *Proporcionar una herramienta para iniciar la construcción del sistema en el IoT.* El modelo de arquitectura de referencia aporta cuatro puntos de apoyo para iniciar el proceso de desarrollo. Primero: proporciona un vocabulario común entre todos los actores que están involucrados. Segundo: permite disponer de una visión abstracta del sistema que se quiere construir. Tercero: facilita la organización y planificación de lo que se quiere realizar. Y cuarto: permite disponer una organización inicial de los bloques o estructuras que se necesitan construir.

- *Generar sistemas para el IoT.* A partir del modelo de referencia y con las instrucciones en forma de guías o procedimientos necesarios, podemos plantear la construcción de los sistemas del IoT como un proceso de generación o instanciación del sistema, de acuerdo a los parámetros particulares que quieran conseguir. Esta aproximación, con mayor o menor grado de automatización, facilita tanto la reutilización inicial de conocimiento previo que hayamos realizado como la posterior integración entre los sistemas que se generan.

3.1.1.3 Conceptos generales del ARM

En esta sección se comentan brevemente los conceptos que se han utilizado para la elaboración del modelo de la arquitectura de referencia. El modelo de la arquitectura de referencia se ha organizado en dos elementos: el modelo de referencia y la arquitectura de referencia. Esta descomposición asegura una división coherente del modelo, al proponer por una parte los modelos asociados a las vistas de la arquitectura en el modelo de referencia y por otra, las características asociadas en la arquitectura de referencia.

El concepto de *vista* de la arquitectura, con carácter general, se define como una descripción estructurada de la misma. El proceso de selección de las vistas de una arquitectura en un modelo plantea diversas cuestiones fundamentales a la hora de resolver el modelo que se quiere proporcionar. En el caso de metaDEPCAS se han elegido las siguientes vistas:

- Vista funcional.
- Vista de la información.
- Vista de la instalación.

La selección de estas tres vistas no excluye otras vistas que para determinados sistemas pueden ser de interés, como por ejemplo, la vista de operación, la vista del contexto o la vista física. Los motivos para plantear únicamente las tres vistas anteriores se centran básicamente en las características propias del IoT que determinan la conveniencia de tener una aproximación más simple del modelo. Por ejemplo, tratar de plantear una vista física de todos los elementos que podemos encontrar en

Modelo de referencia de metaDEPCAS	ARM de metaDEPCAS
Modelo del dominio	–
Modelo de información	Vista de información
Modelo funcional	Vista funcional
Modelo comunicaciones	Grupo funcional de comunicaciones
Modelo de seguridad	Grupo funcional de seguridad

Tabla 3.1: Relaciones entre los modelos de referencia y los elementos de las vistas

el IoT plantearía una cuestión difícilmente abordable dada la heterogeneidad que podemos encontrar y la propia evolución tecnológica.

Además del concepto de vista, el otro recurso de las propuestas para la definición de arquitecturas de referencia que vamos a utilizar, es el concepto de perspectiva de la vista. Tal y como se comenta en detalle en [65], las vistas resuelven cuestiones técnicas de la arquitectura, pero no resuelven otros aspectos cualitativos que se pueden encontrar en los requisitos. Para resolver estos aspectos que pueden influir a varias vistas, se introduce el concepto de perspectiva. Para la definición del modelo y de la arquitectura de referencia utilizaremos las siguientes perspectivas:

- Seguridad.
- Rendimiento y escala.
- Evolución e interoperabilidad

La tabla 3.1 presenta el resumen de las relaciones entre los modelos del modelo de referencia y las vistas y perspectivas de la arquitectura de referencia que se desarrollan en los siguientes apartados.

3.1.2 Modelo de referencia

El modelo de referencia proporciona los conceptos y las definiciones básicas sobre las que definir la arquitectura de referencia. En metaDEPCAS este modelo se fundamenta en cinco conceptos que son clave para la abstracción de cualquier propuesta para el IoT: los dispositivos (actuadores y sensores), las cosas (con carácter

virtual) que agrupan a los dispositivos, los recursos (red, identificación o dispositivos) que dan soporte pero no atributos a las cosas, los eventos identificados que aportan los sucesos a los recursos y a las cosas, y los conectores que dan servicio a la información.

En los siguientes apartados se detallan las estructuras del modelo y cómo se organizan y relacionan los distintos elementos que lo conforman.

3.1.2.1 **Introducción**

El modelo de referencia se compone de tres submodelos principales que son: el modelo del dominio, el modelo de la información y el modelo funcional. El modelo del dominio aporta los conceptos fundamentales de cualquier abstracción del IoT y es la base conceptual de la arquitectura de referencia. A partir de estos conceptos, el modelo de información define la estructura (relaciones, atributos,) de todos los datos que se manejan en un sistema del IoT en un nivel conceptual. El modelo funcional identifica los grupos de funciones que manejan los conceptos del modelo del dominio. Estos grupos funcionales proporcionan las operaciones para interactuar con las instancias de los conceptos del modelo del dominio o para gestionar los datos relacionados con ellos. Esta interacción y gestión se realiza utilizando la información procedente del modelo de información. Las relaciones fundamentales entre los submodelos se representan con la figura 3.2

El modelo funcional incluye dos grupos específicos de funcionalidad: el grupo de funciones de comunicaciones y el grupo de funciones de seguridad. El grupo de funciones de comunicaciones incluye conceptos específicos para manejar la complejidad de las comunicaciones en entornos heterogéneos del IoT. El grupo de funciones de seguridad incluye operaciones para establecer las relaciones y dependencias entre sistemas que comparten información.

3.1.2.2 **Modelo del dominio**

El núcleo del modelo del dominio se centra en los cinco conceptos comentados en la introducción: eventos, dispositivos, recursos, cosas y conectores, relacionados tal y como se muestra en la figura 3.3:

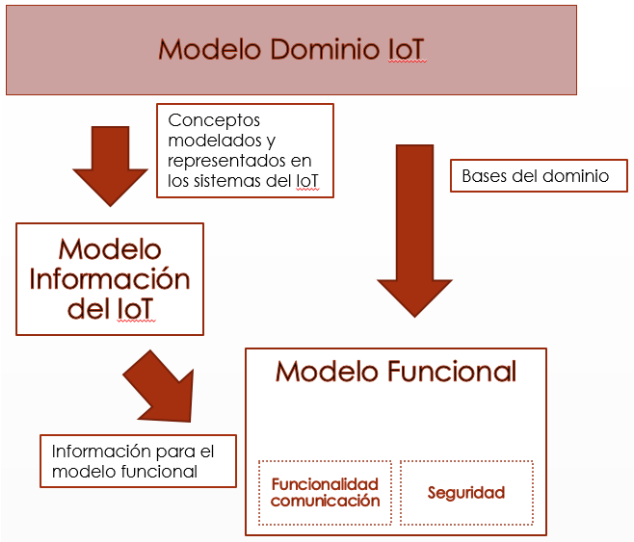

Figura 3.2: Submodelos de metaDEPCAS.

Principales relaciones entre los submodelos de metaDEPCAS

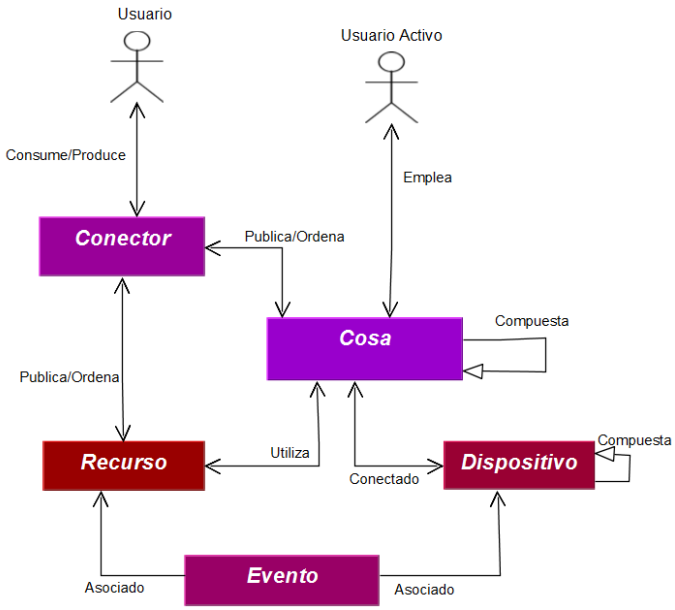

Figura 3.3: Conceptos básicos.

Conceptos fundamentales de metaDEPCAS para el IoT

Estos conceptos son una refinación y extensión de las propuestas disponibles en [66] y [67] en las que se fusionan varios de los conceptos comentados, y se incorpora el concepto de eventos identificados para incluir el concepto de identificación a cualquier elemento recibido desde los dispositivos.

El escenario más genérico del IoT puede modelarse con un usuario que necesita interactuar con una cosa virtual (como se comenta más adelante puede ser una cosa física, una cosa lógica o una cosa aumentada) que abstrae una entidad del mundo real. El usuario, persona física o cualquier otro tipo de sistema, por ejemplo a través de un conector que le da servicio, emplea la cosa para recibir o para enviar información.

El concepto "cosa" que empleamos en metaDEPCAS está asociado al término "elemento virtual" clásico de la bibliografía [68] para representar entidades en el mundo virtual. Existen muchos tipos de representaciones como por ejemplo modelos 3D, avatares, u objetos (como instancias de clases según la programación orientada a objetos), o incluso las cuentas en la redes sociales; todos ellos caracterizan ciertos aspectos del mundo físico en el mundo virtual. Sin embargo, en el IoT, las cosas queremos caracterizarlas fundamentalmente por dos propiedades:

- Son entidades identificadas. La cosa virtual puede representar una entidad física, una agrupación de entidades, o una entidad lógica resultado de algún tipo de composición, pero siempre tiene que estar identificada por un identificador único en el IoT. De esta forma las cosas pueden clasificarse en activas o pasivas. Las cosas activas incluyen aplicaciones, agentes o servicios software que pueden utilizarse por parte de los recursos o de los conectores. Las cosas pasivas son elementos que funcionan como fuentes de información de la representación de la entidad de la cosa.

- Tienen características relevantes. Las cosas son representaciones de un conjunto de propiedades de una entidad. Esta representación virtual de las características físicas nos permite actualizar el estado de la entidad a través de los eventos identificados asociados a ella y de la información obtenida de los dispositivos a los que está conectado. De hecho, no sólo permite conocer el estado, sino que también puede actuar sobre el entorno que rodera a la cosas a través de los correspondientes dispositivos (actuadores).

En estos párrafos relacionados con el concepto de cosa, es necesario comentar brevemente el término empleado para referirnos al usuario. En el escenario simplificado anterior podíamos pensar en un usuario humano interactuando con una cosas física. Sin embargo, el término usuario en el IoT debe ser más amplio, por ejemplo para referirnos a un sistema que da órdenes a otro, o a dos máquinas compartiendo recursos. De igual forma, podemos pensar no sólo en términos de uso del sistema, sino que también podemos incorporar otros roles como el mantenimiento del sistema, los supervisores, etc. Estas clasificaciones de los usuarios se han dejado fuera del modelo de referencia porque se considera que deben ser aspectos de la instaciación de la arquitectura los que se encarguen de plantear y resolver en cada caso estas cuestiones. No hay que dejar pasar que estos aspectos pueden estar relacionados con la organización, las normas, o los procesos de los entornos en donde va a utilizarse la arquitectura de referencia.

El concepto de "dispositivo" incluido en una cosa está asociado a cualquier elemento que disponga de una interfaz con capacidad para obtener información del mundo real o actuar sobre ella. Los dispositivos son los elementos que conectan el IoT con el mundo real. Cada cosa puede incluir uno o varios dispositivos. Los dispositivos conectados a una cosa pueden ser del mismo tipo o pertenecer a varios tipos. Por ejemplo, una cosa asociada a una sala de reuniones puede tener varios sensores iguales de temperatura o puede incluir un sensor de temperatura, un sensor de humos, un sensor de presencia, etc. Desde el punto de vista del IoT podemos considerar dos tipos de dispositivos: sensores y actuadores. Los sensores proporcionan información sensorial sobre el entorno físico en el que se encuentran. En este contexto, de los sensores podemos obtener informaciones atómicas de estados, rangos de valores o informaciones complejas como mallas de puntos 3D obtenidos en un cámara de visión estereoscópica. Los sensores pueden estar embebidos en la cosa física en la que se conectan o en el entorno a la cosa. Los actuadores pueden modificar el estado de la cosa o de otras cosas que se encuentran en el entorno de la cosa a la que se conectan. Se puede modificar el estado simple (encender/apagar o arrancar/detener) o modificar las funciones más o menos complejas en el entorno. También es necesario tener en cuenta que los dispositivos pueden incluir otros dispositivos. Por ejemplo, un dispositivos de "puerta automática", incluye sensores de

movimiento, a la vez que dispone de actuadores en forma de motores que abren y cierran los mecanismos de la puerta.

El concepto de "recurso" utilizado en el modelo representa todo aquel medio de información que utilizan las cosas para interactuar con los dispositivos. Desde este punto de vista, los recursos aportan la infraestructura para las cosas en el sentido más amplio del término. Los recursos pueden ser externos o internos a la cosas. Los recursos externos a las cosas más típicos son los interfaces de red o los sistemas de almacenamiento de información. Los recursos internos son aquellos asociados a los propios dispositivos conectados a las cosas, como memorias de los sensores o interfaces de actuadores en red. Se ha diferenciado el concepto de recurso específicamente de los dispositivos para hacer énfasis en el sentido de soporte a las cosas que aportan los recursos frente a propiedades de las cosas que aportan los dispositivos.

El concepto de "evento identificado" está asociado a una de las propiedades fundamentales de las cosas: la identificación. Dado el carácter heterogéneo del IoT, el evento identificado se asocia a la capacidad de recibir una identificación (ID) de un recurso o dispositivo, en un lugar y en un momento de tiempo. Estos eventos asociados a los dispositivos nos permiten representar la fusión de la información de identificación con la información sensorial o de actuación de los dispositivos. Los eventos también pueden asociarse a recursos para poder identificar determinada información de soporte de las cosas, que puede ser necesaria manejar para diferentes roles en las aplicaciones del IoT. Por último, y dado el nivel de abstracción que usamos en este modelo, los eventos identificados procederán en un futuro próximo de sistemas automáticos de identificación, como por ejemplo los comentados en anteriores apartados con RFID, pero también pueden asociarse a otros recursos de lectura de identificación como por ejemplo, reconocimiento de matrículas, lectores de códigos de barras, etc.

El concepto de "conectores" representa una interfaz única para proporcionar todas las funciones necesarias para interactuar con las cosas desde los usuarios externos. Los conectores tienen la capacidad de publicar la información de las cosas o de los recursos, para que pueda ser consumida por los usuarios, o también pueden producir y ordenar a las cosas o a los recursos, instrucciones. Los conectores en el IoT pueden clasificarse según el nivel de abstracción en: conectores de recursos,

conectores de cosas o conectores integrados. Los conectores de recursos publican información sobre aspectos de calidad de los recursos como por ejemplo, control de acceso, disponibilidad, rendimiento, tiempo de respuesta, etc. Los conectores de cosas publican información sobre las cosas. Este servicio puede funcionar para un única cosa o para bloques de cosas y utilizar diferentes mecanismos de intercambio de información. Los conectores integrados pueden combinar los servicios de conexión de recursos y cosas para obtener información integrada de los dos elementos.

A partir de estas definiciones anteriores podemos ampliar la representación del modelo y reflejar la naturaleza de los distintos conceptos (software, hardware, usuario y combinaciones) en la figura 3.4.

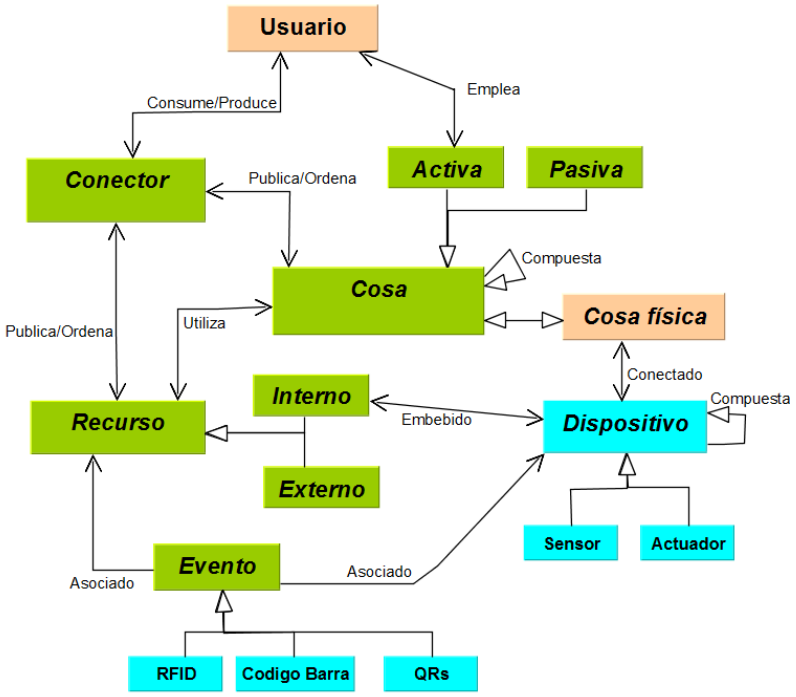

Figura 3.4: Modelo de dominio.

Conceptos de dominio utilizados en metaDEPCAS para el IoT

3.1.2.3 Modelo de información

El modelo de información define la estructura de todas las informaciones en un nivel conceptual. El término "información" se utiliza en este caso como un valor sin

información relevante del contexto. La información incluye únicamente los datos relevantes para contestar a las cuestiones de ¿quién?, ¿qué?, ¿dónde? y ¿cuándo?

La representación de la figura 3.5 muestra la estructura de la información que se maneja y procesa en un sistema en el IoT. Los tres principales elementos son las cosas, las descripciones y las asociaciones. Una cosa en este caso modela la estructura de la información sobre las cosas del modelo del dominio. Incluye al menos dos atributos que son el nombre y el tipo, y ”n” valores a los que se puede asociar metainformación procedente fundamentalmente de los eventos identificados. Por ejemplo, la información de temperatura de un sensor tiene que etiquetarse desde un evento con la identificación de la cosa en la que se conecta y con la referencia temporal en la que se ha medido el valor. Los metadatos pueden incluir metadatos como por ejemplo: las unidades en las que se miden los metadatos. La asociación entre una cosa y una descripción se detalla en relación a alguno o algunos de los atributos de la cosa. Esta asociación incluye un tipo de servicio entre la cosa y la descripción que puede ser de información si el servicio utiliza un valor de atributo o de actuación si se trata de un valor que se escribe en el atributo.

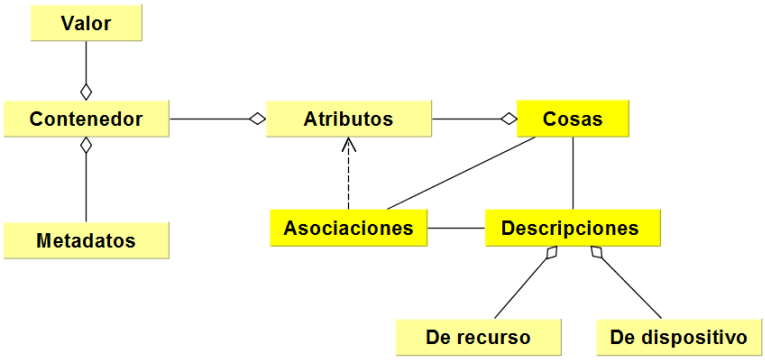

Figura 3.5: Representación del modelo de información.

Conceptos del modelo de información de metaDEPCAS para el IoT

La figura 3.5 muestra la estructura del modelo que se maneja para un sistema en el IoT. Cada cosa necesita incluir un identificador único o un tipo de entidad. El identificador único se asocia a los eventos identificados, mientras que el tipo de entidad representa una clase, por ejemplo: un coche o un sensor de temperatura. Cada atributo se identifica por un nombre, un tipo de atributo y uno o muchos

valores (contenedor). El tipo de atributo identifica el tipo de semántica del atributo, por ejemplo que un valor representa la temperatura. La descripción del servicio representa los elementos más relevantes de un conector, en particular la interfaz que ofrece. Puede incluir descripciones de servicios sobre recursos o sobre dispositivos, si fuera necesario en el sistema del IoT que se modela.

La relación entre el modelo de información y los conceptos del modelo del dominio, se define como que con el primero vamos a representar los elementos del segundo que deben manejarse en un entorno virtual. Además el modelo de información representa las relaciones explícitas entre estos conceptos del dominio. Y en este sentido, podemos considerar que el modelo de información hace de metamodelo que proporciona una estructura de información que se puede manejar en los sistemas del IoT. Esta estructura proporciona todos los elementos que un sistema puede necesitar para representar, manejar, almacenar y recuperar información, y poder utilizarla en la definición del modelo funcional que se comenta a continuación.

3.1.2.4 **Modelos funcionales**

El modelo funcional de metaDEPCAS incluye siete grupos funcionales obtenidos por la aplicación de la propuesta de [69]. Esta descomposición funcional se basa en la definición de descomposición funcional que encontramos en la referencia anterior y que plantea el modelo funcional como un marco abstracto para comprender los grupos funcionales principales y sus relaciones. Este marco define la semántica común de las principales funcionalidades y se utiliza para desarrollar las vistas funcionales. Sobre esta definición es necesario incluir algún comentario particular a la aplicación hecha para el IoT. El marco abstracto refiere a que en el modelo funcional de metaDEPCAS no se incluyen vínculos o conceptos relacionados con tecnologías, dominios de aplicación o aspectos de implementación. En la definición funcional del modelo no es suficiente con enumerar estas funciones, es necesario relacionarlas y establecer cómo se vinculan entre ellas. Las vistas funcionales hacen referencia a los componentes funcionales del sistema en funcionamiento. En este caso, estas vistas se comentan en el apartado de Vistas de la arquitectura de referencia (3.1.3.2).

A partir de los conceptos de modelo de dominio, podemos organizar el modelo funcional en siete grupos funcionales: dispositivos, comunicaciones, DEPCAS, cosas, procesos, organización de los conectores, y aplicación. Además dadas las características de los sistemas en el IoT se ha incorporado de forma trasversal al resto de grupos las funciones relacionadas con seguridad.

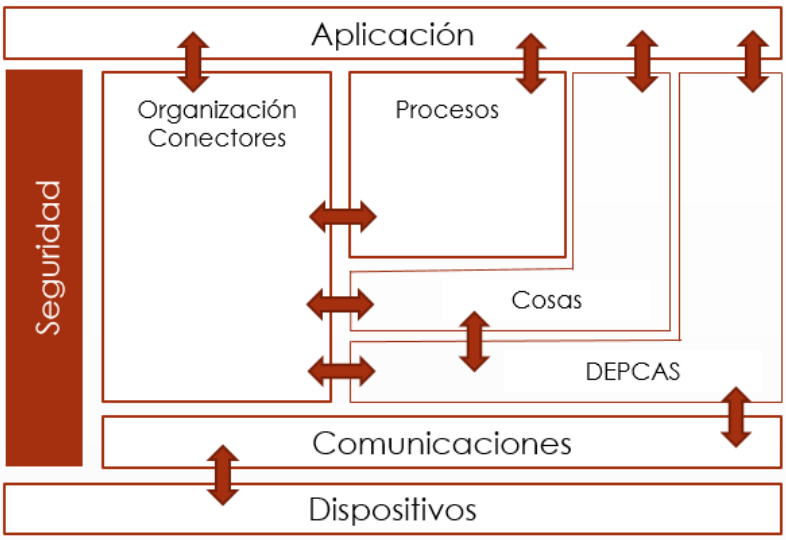

Figura 3.6: Grupos funcionales de metaDEPCAS.
Grupos de funciones y relaciones entre ellos en metaDEPCAS para el IoT

Como se observa en la figura 3.6, las relaciones entre los grupos funcionales tienen un modelo jerárquico desde el grupo funcional de los dispositivos hasta el grupo funcional de las aplicaciones. El grupo trasversal de la seguridad se relaciona con todos los demás grupos aunque no se hayan representado las fechas de relación de forma explícita.

El grupo funcional de la organización de los servicios, se propone como centro del modelo, al servir para componer y estructurar el concepto básico de conector incluido en el modelo de dominio. De esta forma, este grupo funcional enlaza las peticiones de alto nivel del grupo funcional de la aplicación, con los niveles inferiores como las comunicaciones o DEPCAS.

El grupo funcional de los procesos está relacionado con los conceptos más cercanos al resto de sistemas de las organizaciones. El planteamiento de este grupo

funcional es dar soporte a conceptos procedentes del IoT y poder combinarlos con conceptos de los dominios de negocio.

Los grupos funcionales de las cosas y de DEPCAS dan soporte a las funciones de interacción con el sistema del IoT. Se incluye en este grupo de funciones, todo lo relacionado con las informaciones sobre las cosas y los eventos identificados, que se comentaban en el modelo del dominio.

El grupo funcional de comunicación incorpora los diferentes modelos de comunicaciones que se pueden dar a partir del esquema de relaciones del modelo de dominio: comunicación usuario con conector, conector con conector, conector con recurso, conector con cosa, etc. Además incluye los diferentes aspectos relacionados con las comunicaciones físicas, los enlaces, o la red que se utilice.

Los grupos funcionales de aplicación y dispositivos quedan fuera de este modelo, puesto que sus propiedades son, por la naturaleza de las aplicaciones sobre el IoT, tan genéricas que no es posible capturarlas.

Respecto al grupo funcional de la seguridad se encarga de todas las operaciones relacionadas con mantener y garantizar la información. Entre estas funciones se incluyen: asegurar la legitimidad de los accesos, proteger la privacidad de los usuarios, y asegurar la protección de los usuarios del sistema frente a ataques. También se incluyen en estas funciones todas las operaciones necesarias para establecer los vínculos entre extremos que comparten información, ya sea en un mismo sistema o en la vista distribuida del modelo.

3.1.3 Arquitectura de referencia

Como se ha comentado en la introducción, la arquitectura de referencia de meta-DEPCAS pretende proporcionar una base abstracta que permita la construcción de sistemas del IoT que utilizan la identificación por radiofrecuencia en redes de sensores heterogéneos. Para la definición de la arquitectura se utilizan los conceptos de vista y perspectiva, usados en otros modelos de arquitecturas software y descritos ampliamente en la bibliografía sobre este tema. El uso de estos conceptos bien conocidos debe facilitar el proceso de construcción de sistemas, tanto en su diseño como en la implementación. En concreto, se ha utilizado como base la definición de vista y el catálogo de perspectivas propuesto por [70].

3.1.3.1 Introducción

La arquitectura de referencia debe responder a las cuestiones relacionadas con los elementos funcionales y sus relaciones, con la gestión de la información, con las capacidades del sistema y los mecanismos de uso. La arquitectura de referencia debe proporcionar una colección de recursos que puedan utilizarse como base para la construcción de un sistema. Cómo proporcionar estos recursos puede diferir según la propuesta metodológica que se elija. Podemos disponer de una serie de documentos a modo de guía monolítica, que indiquen cómo resolver los puntos antes comentados, o siguiendo la propuesta de [65] estructurar la arquitectura en torno a vistas de la misma que respondan a las cuestiones que se planteaban y perspectivas que incluyan aspectos cualitativos necesarios en el sistema.

Las vistas de la arquitectura se utilizan en la fase de diseño e implementación de un sistema concreto que quiera seguir la arquitectura de referencia. Una vista de la arquitectura es una representación que muestra cómo puede solucionarse uno o más aspectos del sistema que se está construyendo. Cada vista está formada por "puntos de vista" que facilitan el uso de la misma. Cada punto de vista incluye una colección de patrones, plantillas y recomendaciones para realizar un tipo de vista. La lista de vistas no es cerrada. Cada arquitectura de referencia incluye, según sus características, la lista de vistas que mejor resuelve su representación.

Las perspectivas de una arquitectura se utilizan para representar aquellas características que afectan a más de una vista. Estas características suelen estar relacionadas con aspectos no funcionales, con requisitos cualitativos o con requisitos específicos según los actores involucrados. Las perspectivas se definen como un conjunto de actividades, tácticas y guías que se usan para poder conseguir que un sistema tenga en todas sus vistas las propiedades cualitativas definidas en los requisitos. Las propiedades cualitativas son propiedades no funcionales visibles de un sistema como por ejemplo: el rendimiento, la seguridad o la escalabilidad.

Como se ha comentado en los apartados anteriores metaDEPCAS, incluye las vistas funcional, de información, y de uso. Y quedan fuera las vista de los dispositivos y del contexto. En cuanto a las perspectivas, se emplean tres relacionadas con la evolución y la interoperabilidad, la seguridad y el rendimiento.

3.1.3.2 **Vistas de la arquitectura**

La arquitectura incluye tres vistas: vista funcional, vista de la información y vista del uso.

La vista funcional se basa en el modelo funcional comentado y en los requisitos. Cada uno de los siete grupos funcionales del modelo se descompone en diferentes componentes funcionales como resolución a los requisitos que se utilicen. Una vez que se dispone de estos componentes funcionales se pueden definir los casos de uso, los diagramas de secuencia y las interfaces. Los componentes funcionales de metaDEPCAS son:

- Grupo funcional de proceso. Se incluyen dos componente funcionales: modelado y ejecución. El componente de modelado proporciona un entorno para definir procesos basados en notaciones estandarizadas, y para vincular la información del IoT a los procesos de negocio de las organizaciones. El componente de ejecución proporciona la base sobre el IoT para ejecutar los modelos desarrollados con el componente de modelado.

- Grupo funcional de organización de conectores. En este grupo funcional se incluyen tres componentes: suministro de conectores, composición de conectores, y sincronización de conectores. El componente de suministro de conectores proporciona los conectores necesarios para dar servicio a los procesos modelados, a los usuarios y al resto de grupos funcionales: cosas, DEPCAS, y comunicaciones. El componente de composición, facilita la integración de conectores para dar lugar a servicios con funciones más amplias. Y el último componente, el de sincronización de los conectores, se encarga de proporcionar los mecanismos de publicación y suscripción de los servicios ofrecidos.

- Cosas.El grupo funcional de las cosas incluye tres componentes: la resolución, el servicio y la monitorización. El componente de resolución incluye las funciones para vincular una cosa con sus conectores. El componente de servicio maneja las operaciones que una cosa ofrece en sus conectores. El componente de monitorización es responsable de buscar nuevas asociaciones que se incluyen con el componente de resolución.

- DEPCAS. El grupo funcional de DEPCAS incluye dos componentes: el servicio y la resolución. El componente de servicio proporciona las funciones para dar la información sobre la identificación de la cosa obtenida desde los eventos identificados unida a los datos sensoriales procedentes de los dispositivos. El componente de resolución aporta las funciones necesarias para buscar las cosas y sus capacidades.

- Comunicaciones. Este grupo se estructura en tres componentes: conexión lado a lado, red, conexiones extremos a extremo. La conexión lado a lado proporciona el primer nivel de abstracción para resolver las conexiones entre recursos o dispositivos con acceso directo. El componente de red se encarga de las comunicaciones entre redes con mecanismos de direccionamiento y resolución independientes de los tipos y tecnologías empleados. El componente de conexión extremo a extremo, proporciona las funciones para resolver las comunicaciones en los extremos que se conectan.

La vista de información debe ayudar a resolver la estructura dinámica y el flujo de información de los objetos incluidos en el sistema para el IoT que se quiere resolver. A partir del modelo de información, la vista de información debe ofrecer más detalle sobre la información que se representa en el sistema, y dado que se trata de una arquitectura de referencia, esta estructura debe ser independiente de datos específicos de dominio o aplicación. La vista de información se centra, por lo tanto, en el ciclo de vida de la información en el sistema y sus usos por parte de los componentes funcionales comentados en la vista funcional.

La vista de uso proporciona a los usuarios del modelo de referencia, un conjunto de guías para dirigir las decisiones de diseño y construcción de los sistemas. A partir del modelo del dominio básico se distinguen tres grupos de elementos principales que podemos categorizar: los dispositivos, los recursos y las cosas con sus conectores. Cada una de estas categorías plantea diferentes cuestiones de uso según las capacidades que aportan al sistema. En particular los puntos de vista utilizados en metaDEPCAS son:

- Modelo de dominio. Debe ser la base para describir el dominio de la aplicación que se quiere diseñar extendiendo los elementos que lo componen.

- Grupo Funcional de comunicaciones. Debe utilizarse para resolver las capacidades de red que se quiere realizar. En la vista de uso debe utilizarse el diagrama de red con la jerarquía de red y las posibles composiciones de subredes que sea necesario.
- Dispositivos. Es necesario enlazar los dispositivos con los recursos y las cosas que referimos en el modelo, para resolver las capacidades que se pueden ofrecer en los conectores.

3.1.3.3 **Perspectivas**

Las perspectivas que se proponen en metaDEPCAS son:

- Evolución e interoperabilidad.
- Seguridad.
- Rendimiento y escala.

Las perspectiva del modelo se plantea en los siguientes términos: cualidad deseada, requisitos que están afectados, aplicación, actividades que se involucran y tácticas a seguir.

La perspectiva de evolución e interoperabilidad se aplica a las exigencias de cambio continuo en el entorno del IoT. Los requisitos y la tecnología software y hardware cambian tan rápido que es necesario disponer de mecanismos que faciliten la integración tanto de los recursos existentes en un determinado momento como los que puedan surgir en un futuro cercano. La perspectiva puede resolverse con los términos de la tabla 3.2.

La perspectiva de seguridad es fundamental en los sistemas del IoT. Los términos en los que podemos resolver esta perspetiva se recogen en la tabla 3.4.

La perspectiva de rendimiento y escala incluye refiere a dos cualidades que se encuentran muy relacionadas en los sistemas software tradicionales. Y que en el caso de los sistemas para el IoT además todavía se encuentran más vinculadas por el agravante de las características distribuidas y de heterogeneidad que los caracteriza. Los términos en los que podemos resolver esta perspectiva se recogen en la tabla 3.5.

Criterio	Descripción
Cualidad deseada	Disponer de opciones para tener flexibilidad en el sistema para introducir cambios una vez que se encuentra en funcionamiento.
Requisitos afectados	El sistema debe ser capaz de crear cosas nuevas y su semántica asociada, y también capacidades de cambios de las cosas ya existentes. El sistema debe incorporar capacidad para que los objetos compartan recurso con un objetivo común global. El sistema debe manejar las interfaces de los dispositivos. El sistema debe asegurar la interoperabilidad entre objetos de forma identificada.
Aplicación	Las opciones de extensión son una de las características que más se exige a cualquier aplicación para el IoT, sobre todo en el punto de partida actual en el que se trata de una tecnología emergente.
Actividades	Caracterizar las necesidades de evolución. Considerar la evolución de los elementos (dispositivos, recursos y cosas) que se utilizan. Asegurar la facilidad de interoperabilidad de todos los elementos.
Tácticas	Crear interfaces extendibles. Aplicar técnicas de diseño que faciliten cambios. Aplicar técnicas de patrones extendibles. Introducir puntos de cambio en la construcción del sistema que permita usar técnicas de vuelta atrás o realizar derivaciones alternativas. Conservar los pasos intermedios.

Tabla 3.2: Elementos de la perspectiva de evolución e interoperabilidad

Criterio	Descripción
Cualidad deseada	Proporcionar al sistema el nivel de confidencialidad, integridad y control de accesos que exijan los requisitos y detectar y recuperar en caso de fallo de estos mecanismos.
Requisitos afectados	El sistema debe incluir comunicaciones y gestión de la información segura. El sistema debe incluir diferentes niveles de acceso a la información. El sistema debe incluir la información accesible y con qué nivel de acceso está asociada a un conector. Los propietarios de la información deben tener capacidad para establecer los derechos de acceso a su información en los recursos.
Aplicación	A cualquier sistema para el IoT
Actividades	Capturar los requisitos de seguridad. Comprobar el impacto de los requisitos de seguridad en la interoperatibilidad. Análisis de los riesgos con la información. Considerar los aspectos de seguridad en el rendimiento y la escala del sistema.
Tácticas	Utilizar componentes seguros. Asegurar dispositivos y redes que sean de acceso público. Trabajar con infraestructuras seguras.

Tabla 3.4: Elementos de la perspectiva de seguridad

Criterio	Descripción
Cualidad deseada	Que el sistema funcione en un perfil de rendimiento de acuerdo a los requisitos y que exista capacidad para realizar la extensión del mismo según los requisitos de escalabilidad que se exija.
Requisitos afectados	El sistema debe establecer prioridades de funcionamiento de los componentes funcionales. El sistema debe incorporar prioridades de uso de los conectores que proporcionan información. El sistema debe proporcionar integridad a todos los elementos: cosas, recursos, eventos, dispositivos y conectores. El sistema debe ser escalable, utilizable con elementos de diferentes capacidades (simples o complejos).
Aplicación	Según los objetivos del sistema pueden aparecer exigencias de rendimiento específicos en determinadas operaciones o funciones. También es aplicable en aquellos sistemas que quieren tener capacidad de extensión o que no conocen de forma definitiva las funciones que van a realizar pero que quieren tener la capacidad de incluir recursos heterogéneos del IoT.
Actividades	Capturar los requisitos de rendimiento. Crear modelos de rendimiento. Analizar las exigencias de escalado del sistema.
Tácticas	Optimizar y priorizar las operaciones. Distribuir procesos en la medida de los posible y minimizar el uso de recursos más compartidos. Determinar decisiones de diseño que comprometan el rendimiento y el escalado.

Tabla 3.5: Elementos de la perspectiva de rendimiento y escalabilidad

3.2 DEPCAS

Entre los elementos que se han comentado que forman parte del IoT, los sistemas RFID son el futuro de los sistemas de identificación de las cosas que se gestionan. El *middleware* que adquiere la información se ha convertido en uno de los elementos fundamentales y más desarrollados entre las propuestas de investigación.

3.2.1 Introducción a DEPCAS

Data EPC Acquisition System es una arquitectura software para la realización de *middleware* aplicado a la adquisición de información de dispositivos de radiofrecuencia. DEPCAS es una propuesta basada en las arquitecturas software de los sistemas SCADA (*Supervisory Control And Data Acquisition*) creadas para la supervisión y control de instalaciones de cualquier tipo (edificios, subestaciones eléctricas, redes de distribución de gas, depuradoras de agua, etc.). Las características fundamentales de DEPCAS son las de proporcionar un mecanismo abierto y flexible para poder disponer de un sistema de adquisición de información de identificación por radiofrecuencia. El uso del término *EPC* hace referencia a una vocación inicial del sistema para creer en que en el futuro próximo la RFID de bajo coste y medio alcance, será el mecanismo extendido de identificación automática, más allá de si esta situación se produce bajo el estándar *EPC* o con otro mecanismo de codificación.

Las similitudes de los SCADAs y de los sistemas RFID justifica la caracterización del sistema DEPCAS. Tanto los SCADAs como los sistemas RFID comparten una serie de propiedades como son:

- Disponen de fuentes de información. En el caso de los SCADA los equipos remotos o RTUs (*Remote Terminal Units* recogen información sobre los valores que se quiere supervisar y controlar. En el caso del RFID, los equipos lectores reciben información de autoidentificación al detectarse las etiquetas RFID en los campos de lectura de las antenas conectadas.

- Heterogeneidad. En los dos casos el entorno tecnológico tanto software como hardeware es muy heterogéneo y rápidamente evolutivo. Los entornos SCADAs disponen de una gran variedad de dispositivos, tecnologías de red,

aplicaciones software, y entornos debido principalmente a su madurez de uso. En el caso, de los sistemas de adquisición de RFID nos encontramos en un punto emergente de este tipo de entornos y las variaciones y alternativas son más restringidas.

- Aplicaciones por capas. Las dos arquitecturas se plantean en términos similares de una organización del software de abajo a arriba: capa de adquisición y gestión de dispositivos, capa de proceso que filtra, calcula y consolida información, y una capa superior de conectores que ofrece de una forma u otra información a usuarios (como se ha comentado en el modelo del IoT, en el sentido más amplio del término).

- Aplicaciones para el proceso. Tanto en los SCADAs como en la adquisición por RFID, el propósito final de las aplicaciones consiste, en realizar algún tipo de proceso que genere información con valor relevante para la aplicación. En el caso del RFID, este proceso se extiende cuando incluimos los conceptos comentados en metaDEPCAS sobre la integración sensorial con identificación por radiofrecuencia.

- Distribución de la infraestructura. Las funciones de los sistemas de adquisición tienen un carácter nativo de distribución, que puede extenderse desde un pequeño núcleo accesible desde una red local tradicional a los más complejos sistemas de red basados en transmisión por redes WAN (*Wide Area Network*) por los medios de transporte más variado (radio, satélite, fibra dedicada, etc.).

- Entornos multiescalado. El concepto de multiescalado es también una característica intrínseca a los dos tipos de entorno. Los sistemas SCADA se han aplicado a todo tipo de instalaciones en las que se quería tener una supervisión y control del funcionamiento. Por ejemplo, se utilizan para pequeños sistemas en centros de depuración de agua, y también en sistemas de control de oleoductos transnacionales con largas distancias de recorrido controlado. Y este mismo concepto del multiescalado lo podemos detectar en los sistemas de adquisición de RFID que se aplican en pequeñas aplicaciones móviles

para realizar acciones básicas y también en complejos sistemas coorporativos de distribución de productos en toda la cadena de distribución.

Los bloques en los que se organiza DEPCAS son tres: el MDM (*Middleware Device Manager*, el MLM (*Middleware Logic Manager*), y el EPCIS (*EPC Information System*) 3.7.

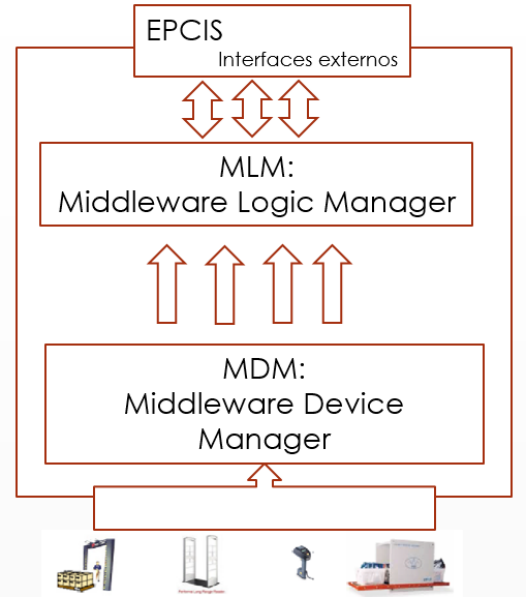

Figura 3.7: Bloques de DEPCAS.

Capas y subsistemas de DEPCAS

El MDM es la capa de abstracción de la red de adquisición y de los dispositivos de lectura RFID y sus antenas. Recibe información de autoidentificación y genera información consolidada en forma de tuplas de quién-dónde-cuándo. Estas información relevante para cualquier sistema de adquisición RFID es independiente de los equipos y de la topología que se utilice para la lectura.

El MLM es la capa de proceso de la información. Los procesos de información a partir de la tupla quién-dónde-cuándo se configuran en función de escenarios (siguiendo la similitud con los sistemas SCADA en los que se procesa las señales de campo para generar información de mayor valor). Cada escenario genera una información que se transfiere a los conectores disponibles en el nivel superior de DEPCAS. En esta capa también es posible recibir información externa que permita

establecer el funcionamiento de escenarios o la obtención de información específica.

El EPCIS es el servicio de información de DEPCAS. Su función básica es realizar de conector software entre los datos consolidados de la adquisición y su proceso, y las aplicaciones externas que utilizan DEPCAS como fuente de información.

En los siguientes apartados se desarrollan las funciones y características más relevantes de cada una de estas tres capas.

3.2.2 MDM: Middleware Device Manager

La capa MDM en DEPCAS tiene las siguientes funciones principales:

- HAL (*Hardware Abstraction Layer*). Dar soporte a la heterogeneidad de lectores y etiquetas que se pueden utilizar.
- RRTL (*RFID Reader Tolopology Language*). Un lenguaje de dominio para modelar los mecanismos de conexión del sistema de adquisición con los lectores.
- MARC (*Minimun Abstract Reader Command*). Un grupo de operaciones mínimas de carácter general que se adaptan a las peculiaridades de los equipo lectores y que facilitan las operaciones básicas entre el sistema de adquisición y los equipos de lectura RFID.

DEPCAS proporciona la lectura de las etiquetas como una cadena de *bytes* y es en esta capa de abstracción, en la que se independiza de las características de los formatos de codificación más comunes, como el ISO 14443 [71] , el EPC Gen2 [72], el ISO 15693 [73] y de los protocolos de de acceso y tamaños [74].

MARC es un conjunto de operaciones que nos permiten operar con un equipo lector alcanzable en nuestra red de adquisición. Los equipos de diferentes suministradores incorporan y resuelven las operaciones de diversas formas, por lo que es necesario disponer de algún mecanismo que realice la abstracción de todas estas particularidades y ofrezca una interfaz común. Estas operaciones se agrupan en las funciones principales que podemos realizar: inicialización, arranque, lecturas, escrituras, y órdenes; y algunos de los comandos disponibles se resumen en la tabla 3.6.

Comando	Tipo	Descripción
setReaderId	Inicialización	Establece el identificador del equipo lector
setNotifyTime	Inicialización	Establece el periodo de notificación
setPollConfiguration	Inicialización	Establece la configuración de interrogación
setStart	Arranque	Inicia el ciclo de interrogación
setStartTimer	Arranque	Inicia el temporizador del ciclo de interrogación
setRun	Arranque	Inicia el ciclo de lectura
readTag	Lectura	Lee un determinado código de etiqueta
readBlock	Lectura	Lee un bloque de etiquetas
writeConf	Escritura	Escribe un bloque de configuración
setAutoStartTimer	Comando	Inicia un temporizador de funcionamiento automático
setAutoMode:	Comando	Inicia el modo de lectura automático

Tabla 3.6: Ejemplos de comandos MARC de DEPCAS-MDM

El tercer elemento relevante en la capa de MDM es el lenguaje de representación de una red de lectores RFID, el RRTL. El RRTL es un lenguaje que permite la definición de redes de lectores adquiridas por el midleware DEPCAS.La estructura básica de esta propuesta incorpora sentencias para gestionar la interacción entre la red que se establece de lectores RFID y el *middleware* que debe recibir la información. El enfoque seguido es el mismo que se utiliza para los sistemas dinámicos de servicios, DSE (*Dynamic Service Evolution*) [75], en donde se puede adaptar el sentido de los elementos de un sistema, al cambiar o evolucionar respecto al esquema original. Cada esquema del RRTL incluye al menos una o varias RRNs (*RFID Reader Networks*), y se vincula a un servicio que de forma habitual se corresponde con un equipo lector (o con varios que transmiten al *middleware* por el mismo interfaz). Las RRNs se pueden relacionar en el esquema del RRTL para realizar distintas operaciones como por ejemplo: función principal, función respaldo, segmento de entrada, segmento de salida, etc. Estas operaciones no son fijas, pueden cambiar en el sistema completo para pasar de una función a otra, según se establece desde el *middleware* que lo utiliza.

3.2.3 MLM: Middleware Logic Manager

La capa de proceso de DEPCAS se denomina MLM y tiene como funciones principales dar soporte al proceso de la lecturas relevantes o RR (*Relevant Readers*) que son las tuplas quién-dónde-cuándo que se generan desde el MDM y consolidar los resultados del soporte. El esquema genérico de funcionamiento puede representarse con la figura 3.8.

El MLM emplea dos grupos principales de configuraciones: las tablas de configuración del escenario y las tablas de información estática. Las tablas de configuración del escenario incluyen datos sobre la estructura del escenario que se que se procesa, por ejemplo, un escenario de seguimiento de objetos establecería rutas entre los “dónde” de las lecturas relevantes del sistema DEPCAS. Y las tablas de información estática incluyen información general de los elementos que se utilizan en el proceso, por ejemplo, el identificador de un “donde” asociado a un equipo lector o el catálogo de códigos de etiquetas que se están supervisando.

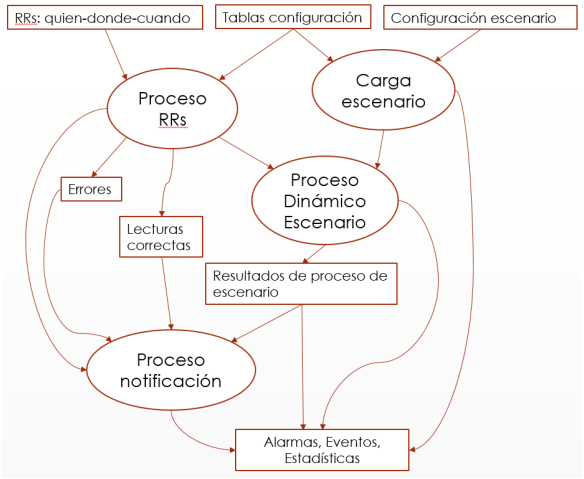

Figura 3.8: MLM.

Procesos y datos relevantes en el MLM de DEPCAS

El proceso del MLM se inicia siempre desde la llegada de una RR que incluye tanto el código leído, el punto de lectura en donde se adquiere, y la etiqueta de tiempo de la lectura asignada. Según el proceso que se realice, las operaciones pueden ejecutarse a partir de una lectura relevante única o para un bloque completo que se reciba. Este proceso genera resultados correctos, si la lectura relevante es significativa en las tablas de configuración estática, o errores si la lectura relevante no es significativa según las tablas de configuración, como por ejemplo recibir un código de etiqueta desconocido.

El proceso dinámico del escenario recibe la configuración del escenario y las lecturas relevantes y aplica los estados de proceso que correspondan en cada caso, utilizando como soporte para realizar las operaciones dinámicas, la estructura de información dinámica del escenario. Según se vayan realizando las operaciones asociadas a cada caso, este proceso consolida los resultados del proceso del escenario y pone la información a disposición del proceso de notificación.

El proceso de notificación realiza dos funciones principales. La primera es consolidar en el subsistema de alarmas, eventos y estadísticas, la información necesaria a partir de los resultados del procesador de lecturas, del proceso de carga del escenario y del procesador dinámico. Y la segunda es notificar los resultados generados a los procesos superiores disponibles en el sistema que residen en la capa EPCIS.

3.2.4 EPCIS: EPC Information System

Las funciones principales del EPCIS en DEPCAS son: gestionar los servicios de publicación y suscripción de las fuentes de información del MLM, gestionar la información histórica consolidada como sistema de información, transferir la información a las aplicaciones de negocio que necesiten los datos del sistema de adquisición, y asegurar los aspectos de integridad y seguridad de la información adquirida.

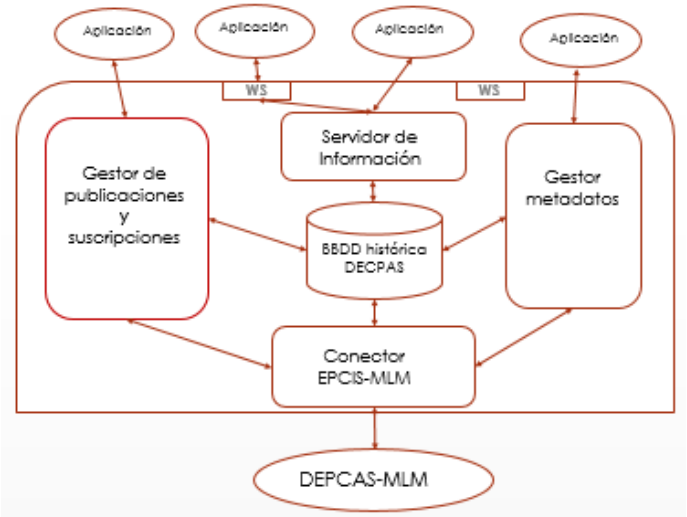

Figura 3.9: EPCIS.

Bloques principales del EPCIS en DEPCAS

El EPCIS se organiza en cuatro bloques principales: el gestor de las suscripciones, el gestor de metadatos, el servidor de información, y el conector con el sistema MLM en tiempo real (Ver la Fig. 3.9). El objetivo fundamental de los tres primeros bloques consiste en proporcionar mecanismos de acceso a la información que se adquiere desde DEPCAS. Cada uno de ellos proporciona la información con un objetivo diferente. El cuarto bloque enlaza en caso necesario, las peticiones de información con el sistema en tiempo real de proceso que representa el MLM.

El gestor de suscripciones y publicaciones aporta la información a modo de proveedor de servicio a las aplicaciones que necesiten mantener una conexión con el sistema de información. Dispone de mecanismos tanto para recibir, como para

entregar información al sistema. Las suscripciones y publicaciones se pueden realizar a nivel de campo, tabla o bloques almacenados de información, obtenidas de consultas predefinidas. Y dispone de un mecanismo de acceso directo que permite transferir información, en caso de que la aplicación de negocio lo necesitase, desde el proceso del MLM a la aplicación externa a DEPCAS.

El servidor de información proporciona acceso a la información consolidada en EPCIS con un mecanismo de cliente/servidor. Resuelve el intercambio de información entre los dos sistemas, el de adquisición y el de negocio, por medio de servicios web que permiten la realización de consultas y actualizaciones de una forma uniforme desde las aplicaciones externas a la información de DEPCAS.

El gestor de metadatos es un convertidor de información que permite establecer relaciones entre estructuras de datos de negocio y las estructuras de datos DEPCAS. Ofrece operaciones para agregar y filtrar la información, con el objetivo de conseguir datos más relevantes para las aplicaciones externas.

3.3 Extensiones de metaDEPCAS

La amplitud de cuestiones que plantean los sistemas del IoT ha causado que las propuestas de arquitectura, consideren extensiones particulares para resolver determinados aspectos particulares. Por ejemplo, es habitual la consideración de los aspectos relacionados con el M2M, extensiones sobre el H2M , o las que consideramos en metaDEPCAS: la extensión relacionada con la fusión sensorial de cosas, sensores y actuadores, y la relacionada con el multiescalado de modelos para el IoT.

3.3.1 Integración sensorial con identificación por RFID

Como se ha comentado en la arquitectura de referencia, la utilización de la información a partir de la adquisición de información de RFID, exige resolver de alguna forma la integración de los datos sensoriales junto a los datos de identificación de radiofrecuencia. Esta integración se puede resolver de tres formas en metaDEPCAS: por lectura directa, por integración de identificación y lectores, y por escenario. Cada una de estas tres formas de integración está relacionada con el momento en el que realiza el vínculo de la información sensorial con la información de identificación.

3.3.1.1 Tags con información sensorial

En el caso de la lectura directa, se utilizan las denominadas “etiquetas sensoras”. Este tipo de etiquetas RFID consisten en integrar en una etiqueta, además del espacio para el código de identificación, microsensores y espacio de almacenamiento adicional de forma que se pueda guardar información sensorial del entorno en donde se encuentran. En este momento, ya se puede acceder a este tipo de etiquetas con sensores de temperatura, detectores de determinados tipos de gases o compuesto. El uso de este tipo de etiquetas, está destinado a monitorizar traslados, tanto en cuanto a las condiciones, por ejemplo para mover vacunas las temperaturas deben encontrarse entre 2 y 8 grados centígrados, como a la verificación, por ejemplo que un determinado producto de alto coste es el original que se ha enviado y no ha sido remplazado en el traslado. En estos casos, las etiquetas RFID almacenan además

Descripción	Sentencia configuración
Sensor: Temperatura - RFID reader: IN	< asp:inclusion > < ![CDATA [(&(data.type=temp) (sensor.location=IN))]] > < /asp:inclusion >
Sensor: Metano - RFID reader: OUT	< asp:inclusion > < ![CDATA [(&(data.type=gas)(sensor.location=OUT))]] > < /asp:inclusion >

Tabla 3.7: Configuración de inclusión de información de sensor en lector RFID

Descripción	Sentencia condición
Sensor: Temperatura	< asp:condition> <[CDATA [(temperature¿273K)]] > < /asp:condition >
Sensor: Metano	< asp:condition> <[CDATA [(val = TRUE)]] > < /asp:condition >

Tabla 3.8: Configuración de condiciones de lectura de la información de sensor

de la información de autoidentificación, la información del entorno en el que se han encontrado. Para realizar la lectura de la información sensorial se ha incorporado a DEPCAS una extensión en la adquisición de la información que hemos denominado: "SensorExt". Esta extensión se resuelve con dos operaciones: inclusión de la información del sensor y lectura de bloque de sensor. La inclusión de información del sensor se realiza a través de la configuración de la capa del MDM, introduciendo sentencias de inclusión como las que se incluyen en 3.7.

En el caso de disponer de las configuraciones de sensores, y si en un ciclo de lectura del sensor se activa la condición de valores disponibles, entonces la información se traslada siempre que se cumplan unas determinadas condiciones que interpreta el proceso del MDM de DEPCAS con la etiqueta "condition", tal y como se muestra en la 3.8.

3.3.1.2 Integración de lector RFID con sensores

La segunda opción que se plantea para resolver la fusión sensorial con identificación por RFID es el uso de un determinado lector RFID integrado con un equipo

sensor. En este caso, la solución se restringe por un diseño determinado de la arquitectura hardware que integra tres elementos: el lector RFID, el dispositivo sensor y el microprocesador que realiza el proceso de fusión de las dos fuentes. De forma adicional se han incluido en el diseño otros elementos adicionales necesarios para permitir la integración directa con DEPCAS, que son el módulo de memoria y el de interfaz (3.10).

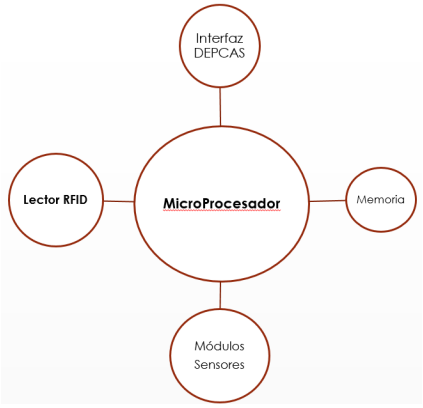

Figura 3.10: Sensor con lector RFID.

Integración de lector RFID embebido con sensor

El microprocesador controla tanto el lector de radiofrecuencia como el módulo sensor que se integra y realiza las operaciones relacionadas con la lectura de información RFID, el movimiento de información a la memoria, la conexión y la desconexión de los otros módulos, o del interfaz, etc. El equipo lector RFID incorpora tanto la función de lectura de RFID como el interfaz de conexión con el microprocesador para la transmisión de información, e incluso al tratarse de un dispositivo para integración incorpora la antena para activar el campo de radiofrecuencia. El módulo de interfaz está relacionado con el microprocesador que se elija y en la actualidad cualquiera de los mecanismos más habituales de conexión pueden integrarse fácilmente para tener enlace por Ethernet, Bluetooth, WLAN o Zigbee.

Entre las opciones disponibles, la propuesta descrita en la tabla 3.9 cumple con los procesos de adquisición realizados en DEPCAS.

En la tabla 3.9 se han incluido únicamente la asociación de la etiqueta RFID a la información sensorial de un módulo GPS y de un sensor temperatura, aunque se

Módulo	Tipo	Descripción
Microprocesador	ATmega1281	Placa Arduino con procesador Atmel - Procesador de bajo consumo de 8bits con arquitectura RISC - 128Kb memoria flash - 4 KB EEPROM - 8 KB RAM - 54 pins digitales - 16 analógicos - 1 Puerto SPI - 1 Puerto USB
Lector RFID	Winnix HYM360 Nano-UHF	Lector RFID UHF con protocolo EPC Gen2/ISO 18000-6C - Distancias de 2m - 40 etiquetas por segundo - Interfaz 24 bits - Puerto UART para conexión a procesador
Interfaz wlan	Zero ZG2100M	Módulo Wi-FI de bajo consumo - Protocolo 802.11 B/G y v2.0 802.11n - Puerto SPI (*Serial Peripheral Intergace* para conectar con el procesador
Memoria	Micro SDCard	Módulo de tarjeta Micro SD para Arduino SKU: DFR0229 de DFRobot con interfaz SPI para conexión con el procesador.
Sensor	GPS	Módulo GPS LEA-6H de DFRobot compatible con Arduino con memoria flash integrada y conexión por puerto UART.
Sensor	Temperatura	Sensor de temperatura y humedad DHT11 de DFRobot de bajo consumo para placas Arduino.

Tabla 3.9: Prototipo de Sensor con lector RFID

han planteado prototipos similares con el uso de información gráfica procedente de cámaras 2D.

El prototipo propuesto realiza el proceso de asociación de la identificación con la información sensorial que se define en el siguiente proceso:

- Primero se realiza la inicialización de los módulos para poner los recursos en disposición de ser usados por el microprocesador.
- Se entra en un bucle en el que el procesador se detiene hasta que: o bien se recibe señal de recepción de código RFID, o bien ser recibe señal de petición de mensaje por la WLAN.
- Si se ha recibido un tag, entonces se realizan las siguientes acciones: se lee el código, se activa el sensor correspondiente y se lee el valor del sensor (existen comprobaciones adicional que pueden llevar un tiempo para asegurar la calidad de los datos del sensor), y los dos valores, RFID más datos sensoriales, se almacenan en la memoria externa.
- Si se ha recibido un mensaje por la WLAN (procedente del MDM de DEPCAS) se atiende la lectura y se crea el mensaje de respuesta a partir de los datos de la memoria externa y los marca como leídos.
- Actualiza los datos de estado del lector RFID, del sensor y de la memoria externa y vuelve al estado inicial de espera.

Este proceso puede representarse con el esquema de flujo que se incluye en la figura 3.11.

3.3.1.3 **Integración en escenario**

La utilización de las dos soluciones anteriores exige la existencia de una integración entre la información de los sensores y la información de RFID. En el primer caso, integrando la información sensorial en la etiqueta y en el segundo integrando el equipo lector en el sensor. En determinadas ocasiones no será posible realizar ninguna de estas dos opciones por lo que es necesario establecer otro tipo de mecanismo. En esta caso DEPCAS, como mecanismo de lectura de información

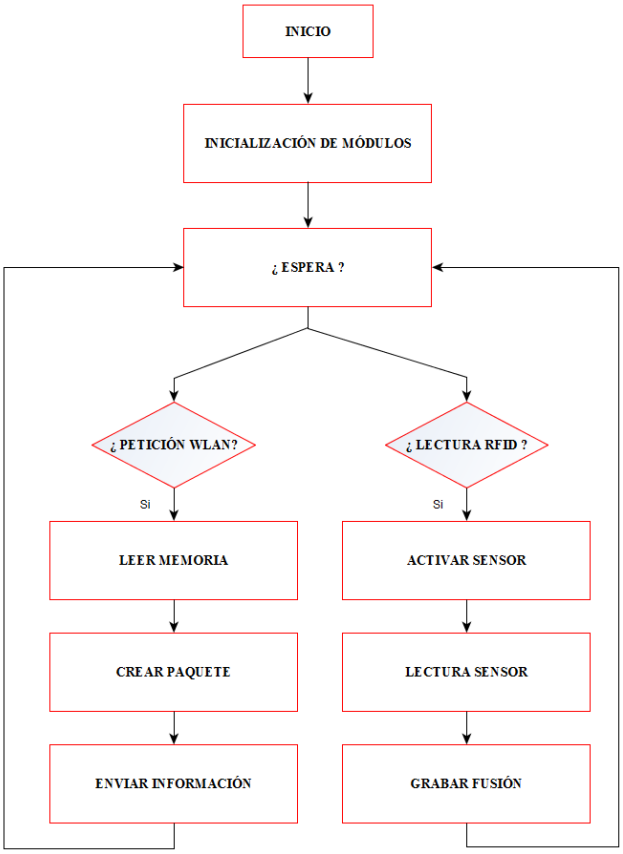

Figura 3.11: Proceso RFID más sensor

Integración de los datos de un lector RFID con información sensorial

del modelo metaDEPCAS incluye la opción de añadir la información del sensor a través de un servicio WAB (*Wire Admin Service*) que se accede desde el componente OSGi (*Open Services Gateway Initiative*) que existe en el MDM de DEPCAS 3.12.

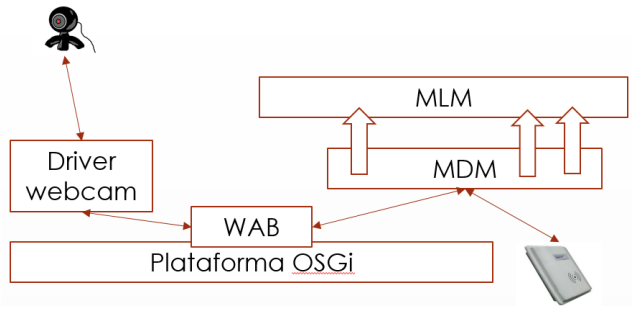

Figura 3.12: Sensores en metaDEPCAS.
Integración de información sensorial en DEPCAS por OSGi

En esta solución, se exige que se disponga del driver (comunicaciones y procesos) del sensor y que integre el servicio WAB al que se va a conectar DEPCAS para obtener la información (de forma similar al primer caso desde el punto de vista del *middleware*).

3.3.2 Multiescalado en metaDEPCAS. Distribución de la información

Una de las cuestiones más relevantes de los sistemas distribuidos, como los sistemas SCADA, los sistemas ERP, etc o las aplicaciones para el IoT es la gestión del contexto multiescalable en el que deben utilizarse. La gestión del contexto de las aplicaciones facilita resolver dos aspectos fundamentales: la adaptación de las aplicaciones a los entornos en donde se utilizan y la capacidad para resolver nuevas cuestiones que se plantean.

La característica heterogénea de los objetos que podemos manejar en el IoT y su interconectividad en todo tipo de entornos, exige disponer de mecanismos que se adapten de forma independiente a la escala en la que se quieren utilizar. El mecanismo que proponemos en metaDEPCAS se integra en la infraestructura

de la arquitectura de referencia y permite adaptarse a mecanismos de comunicación multiescala, la gestión y la protección de la información y facilitar el despliegue autónomo de forma independiente en las diferentes escalas. En este entorno, encontramos sistemas muy pequeños (redes de sensores de capacidad de proceso limitada, sin almacenamiento, y con bajo consumo y pocas opciones de comunicación) que deben funcionar en sistemas grandes y distribuidos, que incluyen gran capacidad de proceso, distribución y complejos sistemas de comunicación. Por lo tanto, en el IoT, la necesidad de asignar los componentes de gestión de contexto es mayor. De hecho, muchos objetos se pueden conectar a la infraestructura de una red global en cualquier momento durante su ciclo de vida y de forma temporal o permanente. Además, algunos objetos se encuentran en ubicaciones fijas mientras que otros son móviles. Los objetos ya no pueden considerarse sólo como los productores del contexto, sino que pueden ser actores independientes que intercambian, generan y transforman información de forma autónoma. En lo que concierne al valor de los datos de contexto, una solución consiste en su descripción a través de metadatos. La elección de los metadatos relevantes debe ser abierta y flexible.

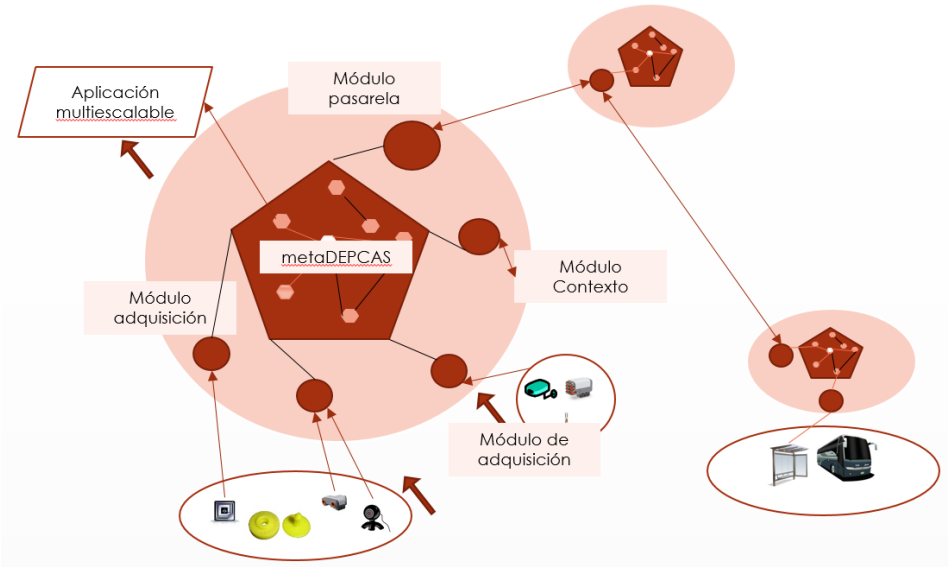

Figura 3.13: Extensión de metaDEPCAS.

Elementos de la extensión multiescalable de metaDEPCAS

La arquitectura para la gestión del contexto multiescalable se plantea en tres niveles en metaDEPCAS: entrega de datos del contexto, proceso de los datos de

contexto y presentación a las aplicaciones. La figura 3.13 muestra los principales componentes utilizando la terminología de [76]. Cada círculo completo se corresponde con un sistema o espacio. Cada espacio funciona en una escala que se vincula con su infraestructura, y se puede considerar como una fuente de datos. En el espacio disponemos de módulos que recogen información de adquisición, del contexto o son pasarelas hacia otros espacio. Se puede considerar que los módulos son elementos funcionales que gestionan el contexto en el que funciona el espacio. Esta gestión consiste, tanto en ofrecer la información que existe en el espacio como recibir información procedente de otro espacio. Por ejemplo, un módulo de adquisición puede ser un servidor COAP o un middlware DEPCAS. Un módulo de contexto es un encapsulador de información de los módulos de adquisición. Las informaciones de los módulos de contexto se distribuyen a través de la infraestructura de la aplicación a otros módulos de contexto que pueden ofrecerlo a aplicaciones sensibles al contexto. De esta forma, la información local que se ha adquirido ha pasado a los módulos de contexto y ha podido ser ofrecida a aplicaciones que estaban potencialmente separadas del origen. Se asegura así que los consumidores de la información puedan acceder a los datos independientemente del contexto desde el que accedan. Los módulos pasarela se dedican, si es necesario, a transmitir y recibir información y contexto, a o desde otros espacios.

The limits of my language are the limits of my world.

Ludwig Wittgenstein

CAPÍTULO

4

Lenguaje Genérico de Diseño: DGL

En este apartado se presenta la propuesta denominada DGL o lenguaje genérico de diseño para aplicaciones del IoT. Como se comentaba en el capítulo introductorio, el objeto de este lenguaje es proporcionar los elementos generales que podemos incluir en el desarrollo de las aplicaciones para el IoT. Las características del DGL se incluyen en el grupo de lenguajes específicos de dominio DSL (*Domain Specific Language*) puesto que se ha realizado a partir del análisis general de dominio de las aplicaciones para el IoT y contiene específicamente elementos propios de él. En este apartado se incluye además un proceso de desarrollo que se basa en el lenguaje de diseño proporcionado y en el modelo y conceptos comentados en el capítulo anterior.

4.1 Diseño del DGL

El lenguaje de diseño genérico es una propuesta de lenguaje de tipo DSL para la definición de aplicaciones para el IoT. El DGL está formado por tres lenguajes que aplican a cada fase del desarrollo de las aplicaciones del IoT: el DDL (*Design Domain Language*), el DFL (*Design Funtional Language*) y el DAL (*Design Application Language).* Cada uno de estos lenguajes que conforman el DGL incluye su propio vocabulario y sintaxis, que debe emplearse en el proceso genérico de desarrollo que proponemos basado en tres etapas: etapa del dominio, etapa de la aplicación (funcional, instalación y plataforma) y la etapa de instalación.

4.1.1 Conceptos sobre DSLs

Un lenguaje específico de dominio, DSL (*Domain Specific Language, DSL*) se define como un lenguaje de especificación, de diseño o de programación que ofrece, a través de notaciones y abstracciones particulares, capacidad expresiva orientada, y normalmente restringida, a un dominio de problema particular [77]. Se trata de lenguajes pequeños que ofrecen un conjunto reducido de notaciones y abstracciones más cercanas al dominio del problema que al dominio del entorno de la implementación, y de esta forma se permite usar un mayor nivel de abstracción para el desarrollo de las aplicaciones. Así se consigue que los desarrolladores de software trabajen con conceptos que comprenden los usuarios de las aplicaciones y que recojan sus necesidades, disponiendo de un lenguaje que contiene únicamente estos conceptos.

El intento de acercar los entornos de implementación a los entornos del problema que se resuelve, es una cuestión clásica en el desarrollo de la ingeniería de software. Tradicionalmente han existido más aproximaciones para el desarrollo de aplicaciones específicas del dominio, como el uso de librerías de subrutinas que son llamadas por las aplicaciones para realizar las tareas del dominio, o como el uso de marcos de trabajo que invocan a los métodos de la aplicación para realizar estas tareas. Frente a estas alternativas, los DSLs ofrecen una capacidad expresiva mayor. El estudio de estos DSL es un tema que se encuentra en auge gracias al importante papel que estos desempeñan dentro de la metodología de Desarrollo de Software Dirigido por Modelos. Sin embargo, el uso de estos lenguajes no es

Lenguaje	Valor del nivel
C	2.00
C++	6.00
HTML	23.00
JAVA	6.00
Lenguaje Máquina	0.50
MATHCAD	60.00
SQL	25.00
VisualBasic 5.0	11.00
VHDL	17.00

Tabla 4.1: Lenguajes y nivel numérico de acuerdo a sus características.

algo nuevo. Ya desde los años 50 se desarrollaron lenguajes del tipo DSL como podemos considerar al BNF (*Backus-Naur Form*) para la especificación de sintaxis. Y en la actualidad podemos encontrar un gran número de DSLs disponibles en diversos campos y alguno de ellos de uso tan extendido como otros lenguajes de carácter general. Por ejemplo, podemos citar el lenguaje de marcas HTML (*HyperText Markup Language*), el VHDL (*Very High Speed Integrated Circuits Hardware Description Language*) para el diseño de hardware, o el SQL (*Structured Query Language*) para el acceso a bases de datos. En estos lenguajes se puede observar de forma clara, la propiedad que se comentó anteriormente de elevar el nivel de abstracción al desarrollar software alejando a sus usuarios de la programación pura, hasta el punto de que muchos usuarios de algunos de estos lenguajes (como en el caso del HTML) no se consideran a sí mismos programadores. En la tabla 4.1 se pueden observar estos DSL y otros lenguajes, junto con su correspondiente nivel definido según la amplia lista incluida en [78]. Este nivel relaciona la cantidad de instrucciones lógicas que se emplean para ejecutar una misma tarea (mayor nivel indica mayor capacidad expresiva). Para poder comparar la capacidad expresiva de los programas desarrollados se han incluido en la tabla lenguajes que son de propósito general como C, C++, JAVA y ensamblador.

Otra alternativa al uso de los DSL la constituyen los lenguajes de modelado de propósito general, especialmente UML (*Unified Modeling Language*), extendido

y soportado por muchos tipos de herramientas. Este lenguaje ha sido usado para realizar el modelado de sistemas que se describen en función de diagramas de clases, máquinas de estados, etc. Sin embargo, no se suele emplear con el propósito de generar código en dominios específicos. El principal problema que presenta es su complejidad, debida al gran número de elementos con que cuenta el lenguaje, lo que dificulta su uso. Además, intentar describir aplicaciones de un dominio específico puede resultar inapropiado por la incapacidad de expresar ciertos conceptos del dominio en UML. Aunque éste dispone de mecanismos de extensión que pueden permitir modelar estos conceptos, estos tampoco constituyen una solución apropiada. En UML 2.0 se usan los perfiles (UML Profiles) para este fin, que permiten definir nuevos elementos que representan los conceptos del dominio en cuestión. Pero surgen una serie de problemas que complican su utilización. Por un lado, la capacidad de extensión está limitada por lo que ciertos dominios no pueden llegar a ser satisfactoriamente modelados. Y por otra parte, nos seguimos encontrando con un lenguaje todavía muy amplio en el que los expertos del domino que no tengan un conocimiento profundo de UML no son capaces de expresar los conceptos específicos.

Diversos artículos, como por ejemplo [77], [79] enumeran las ventajas que se obtienen al utilizar un DSL para la descripción de aplicaciones. En todos estos artículos se coincide en que la principal ventaja del uso de los DSLs es elevar el nivel de abstracción, de forma que las descripciones de las aplicaciones puedan ser realizadas por un experto del dominio, en lugar de por un programador o un equipo de programadores. En [79] se destacan dos características de los DSL que dan origen a las ventajas que se obtienen de su uso. En primer lugar, la comprensibilidad de las especificaciones realizadas con los DSL. Estas descripciones pueden ser fácilmente interpretadas por los expertos del dominio, facilitando su escritura y reduciendo el número de errores introducidos en ellas, más relacionados con los conceptos del dominio que con conceptos de programación de los GPL (*Generic Programming Language*). Una segunda característica que se destaca es el hecho de que las representaciones internas de las construcciones lingüísticas de los DSL sean más adecuadas para el tipo de procesado y análisis que ejecutan los analizadores, simuladores, animadores y traductores de los lenguajes que las empleadas por otros

tipos de lenguajes. Esto facilita en gran medida el desarrollo de herramientas auxiliares de los anteriores tipos adaptadas al dominio, no sólo reduciendo su tiempo de desarrollo sino también aumentando su calidad.

Teniendo en cuenta lo descrito en el párrafo anterior, se pueden concretar las ventajas del uso de un DSL de forma resumida con los anteriores comentarios y añadiendo alguna más de interés en:

- Los DSL permiten elevar el nivel de abstracción hasta el dominio del problema. De esta forma, los expertos del dominio pueden entender, validar y desarrollar programas haciendo uso del lenguaje.
- Las especificaciones creadas con estos lenguajes pueden ser concretas, autodocumentadas y se pueden reutilizar para diferentes propósitos, no limitándose a su ejecución.
- Se mejora la productividad, fiabilidad, mantenibilidad y portabilidad de las aplicaciones.
- Los DSL promueven la conservación y la reutilización del conocimiento del dominio.
- Se pueden realizar tareas de optimización y validación a nivel de dominio.
- Los programas realizados con los DSL pueden ser probados.
- Los DSL pueden ser usados para la generación de código, permitiendo que una parte importante del proceso sea automatizada.

Además de las ventajas, el uso de un DSL también puede introducir una serie de inconvenientes que se deben tener en consideración antes de abordar su desarrollo. El principal de estos problemas lo constituye el coste de diseño, implementación y mantenimiento del nuevo lenguaje, que requiere de un importante esfuerzo y de unas cualificaciones técnicas mayores que las exigidas a un programador convencional. Este coste es aún mayor si se tiene en cuenta el desconocimiento de los mecanismos para la elaboración de un DSL por parte de desarrolladores y la falta de documentación que se encuentra en muchos de los casos. Esto se suma a la complejidad de encontrar una solución que permita describir cada aplicación del

dominio concreto, más la complejidad introducida al gestionar otro tipo de problemas relacionados con el lenguaje (análisis sintáctico, control de tipos, restricciones, etc). A estos costes por parte del desarrollador del DSL hay que sumar los costes, sobre todo en términos de aprendizaje, que tendrían los usuarios del lenguaje, aunque teniendo en cuenta que el esfuerzo para aprender a usar un DSL es considerablemente menor que el empleado en el caso de un GPL.

4.1.2 Proceso de creación del lenguaje

El proceso de creación y definición de un DSL es una tarea que requiere un conocimiento experto tanto sobre el dominio de aplicación como sobre el dominio del desarrollo de lenguajes. Antes de tomar la decisión del desarrollo de un DSL, es conveniente realizar un estudio sobre la viabilidad de esta opción, y que se demuestre que el esfuerzo adicional que se va a emplear en el desarrollo del lenguaje va a ser compensado por las ventajas que éste ofrecerá. De acuerdo con la propuesta incluida en [80] este estudio se consideraría como la primera fase en la definición de un DSL, que denomina fase de decisión. En esta fase se tiene que considerar de un lado todas las dificultades y problemas asociados al desarrollo del DSL (sobre todo si es necesario lo relacionado con las cuestiones económicas y temporales) y la existencia de otros DSL para el dominio en cuestión. De otro lado se deben considerar los beneficios que se espera obtener de su uso. Para evaluar estos beneficios se deben buscar una serie de patrones en el dominio, que previamente han demostrado ser resueltos de forma eficaz por el DSL en otros dominios. Por ejemplo, patrones que plantean ventajas para la definición de un DSL son la ejecución de operaciones de análisis, verificación, optimización, transformación, representación de datos y la realización de tareas mecánicas o que impliquen interacción con el usuario. En general, cualquier propiedad de los DSL que permita simplificar el desarrollo de aplicaciones o rebajar los conocimientos exigidos a los usuarios potenciales del DSL debe ser considerada en esta fase.

En el artículo anterior se proponen tres fases para el desarrollo de un DSL. Estas fases son la de análisis, diseño e implementación del lenguaje. En la fase de análisis se recopila la información del dominio tomando como entradas documentos técnicos, el conocimiento de los expertos del dominio, código fuente existente y estu-

dios de los usuarios del dominio. La salida de esta fase dependerá de la forma en la que sea abordada, normalmente de manera informal aunque también se pueden usar metodologías formales de análisis como FODA (Fortalezas-Oportunidades-Debilidades-Amenazas) o FAST (*Functional Analysis System Technique*). Con éstas se puede obtener un modelo del dominio formado por una definición y terminología del dominio, una descripción de los conceptos del mismo y modelos de características del dominio. Este modelo del dominio serviría a su vez como entrada para la fase de diseño, que también puede afrontarse de manera formal e informal, optando por desarrollar un DSL desde cero o bien por extender a otro lenguaje existente. Finalmente, en la fase de implementación se distinguen diversas alternativas como el uso de compiladores, intérpretes, preprocesadores, extensión de lenguajes existentes o de compiladores.

Aunque la clasificación propuesta en [80] encuentra un gran número de patrones presentes en el desarrollo de un DSL, no propone unas líneas claras a seguir para la definición de un DSL. Sin embargo, en el contenido de [81] se propone una serie de pasos sencillos para el desarrollo de DSL. Estos pasos sencillos se consideran como un proceso incremental que se repite refinando el lenguaje con cuatro fases:

- Identificación de abstracciones y relaciones entre ellas.
- Especificación de los conceptos del lenguaje y sus reglas.
- Elaboración de la representación del lenguaje.
- Creación de las herramientas.

Si se comparan las dos propuestas se puede considerar que la identificación de abstracciones y relaciones equivale a la fase de análisis, las fases de especificación y elaboración de representaciones equivalen a la fase de diseño, y por último la fase de creación de las herramientas correspondería a la fase de implementación. A la hora de ejecutar estos pasos, sobre todo en la especificación de los conceptos y su representación, se debe tener en cuenta una serie de consideraciones de diseño como por ejemplo puede ser el evitar todas las decisiones de restricción del uso del lenguaje. Así si se vincula el DSL a una tecnología concreta de implementación se

obliga a los usuarios del DSL a esa tecnología, o que sea necesario usar otras herramientas externas para funcionar. De igual forma siempre hay que tener en cuenta quiénes serán los usuarios del lenguaje y cuáles son sus conocimientos y habilidades para poder reducir los tiempos de formación con el DSL y los empleados en el desarrollo de aplicaciones.

En esta misma línea de trabajo, en [82] se comentan cuáles deberían ser los requisitos de diseño necesarios para elaborar un DSL. En él se menciona que cualquier DSL debe contar con una serie de requisitos obligatorios más algún requisito adicional. Señala en primer lugar a los conceptos del lenguaje, que necesariamente deben corresponderse con los conceptos importantes del dominio, con una relación de uno a uno, es decir, cada concepto del lenguaje representa sólo a un concepto del dominio. Además, el conjunto del lenguaje debe ser simple, pero también debe permitir mejorar la calidad de los sistemas generados. En este sentido otras características no funcionales como la fiabilidad, la seguridad o el rendimiento (entre otras) son las que se considera que mejoran la calidad de los productos. De igual manera, en esta aportación se cita como requisito fundamental, para la viabilidad de un DSL su uso futuro. Si existen opciones de que su utilización pueda dar lugar a un número importante de aplicaciones, entonces se compensa el esfuerzo consumido tanto en su desarrollo, como en la construcción de herramientas que lo soporten, en incluso el trabajo necesario para que se facilite en todo lo posible la integración con otras herramientas existentes. Otros requisitos que considera opcionales en un DSL, pero en cualquier caso deseables, son la escalabilidad del lenguaje la comprensibilidad, la economía del espacio o la accesibilidad que se engloban, de forma más genérica, en el requisito de usabilidad.

Una vez creado el DSL, debe ser implementado mediante el desarrollo de herramientas como un entorno de desarrollo, generadores, compiladores, editores, traductores y depuradores. La dificultad de esta tarea ha sido un grave inconveniente que muy frecuentemente ha hecho en muchos casos tener que abandonar el desarrollo de nuevos DSL. En los últimos años han aparecido nuevas tecnologías que simplifican el desarrollo de DSL y permiten su integración dentro de un entorno MDE. En el estudio [83] se presenta una comparativa entre algunas de estas tecnologías. De todas las tecnologías analizadas se destaca Eclipse, junto con sus marcos EMF y GEF, como las alternativas con más potencia para el desarrollo de entornos

para DSL nuevos. Cuenta con el respaldo del entorno Eclipse, lo que añade gran cantidad de funcionalidad a su editor del lenguaje y ofrece la posibilidad de integrar el lenguaje con un gran número de herramientas externas. Como contrapartida, esta tecnología es la más compleja de utilizar de entre todas las analizadas. El hecho de que haya aparecido el framework GMF (*Graphical Modeling Framework*) para la generación de editores gráficos basados en EMF y GEF, aporta herramientas de representación más accesibles pero no disminuye la complejidad de los conceptos incluidos en estos marcos de trabajo.

4.2 Lenguaje Genérico de Diseño: DGL

En este apartado se presentan los tres lenguajes que forman parte del DGL: el lenguaje del dominio, el lenguaje funcional de las cosas y el de la aplicación. Además y como punto de partida es necesario comentar diferentes elementos relacionados con la concepción del DGL. Primero y relacionado con la división del DGL en tres recursos, es necesario justificar que los tres recursos se corresponden, a que, cada uno de ellos es una herramienta que se utiliza en los diferentes pasos del proceso de desarrollo de aplicaciones que se presenta en el siguiente apartado. De esta forma se ha planteado como mejor forma para la elaboración del DSL, en lugar de plantear un todo único, presentar los recursos en torno a los tres lenguajes que estructuran el DGL y que se explican a continuación. Un segundo aspecto es relativo a la acrónimo en si mismo. La expresión "genérico" se utiliza para significar la capacidad del lenguaje para poder adaptarse a los diferentes dominios en los que podemos aplicar el lenguaje, de hecho, la primera parte del lenguaje se ha concebido para tener capacidad de construir vocabularios de dominios. Y por último, y relacionado con los adjetivos incluidos en el título de esta tesis, multi-paradigmático y ejecutable es necesario comentar que la estructura de los recursos del DGL se ha realizado de forma independiente de los paradigmas habituales (en los conceptos que forman parte de los recursos aparecen elementos de varios paradigmas tradicionales) con un DSL externo, y con el objetivo de poder lograr que los diseños realizados puedan convertirse en aplicaciones en funcionamiento.

4.2.1 Introducción

Como se ha comentado en el capítulo del estado del arte, las aplicaciones que se desarrollan para el IoT incluyen características que diferencian sus procesos de realización. Una forma de resolver estas características es seguir una estrategia de separación de los asuntos de interés (*concern*) del proceso de desarrollo de las aplicaciones del IoT. En el caso de las aplicaciones para el IoT, podemos considerar que existen cuatro asuntos vinculados a la construcción de aplicaciones: los conceptos del dominio, los conceptos específicos de las funciones, los conceptos de las aplicaciones, y los de las plataformas.

Los conceptos del dominio son aquellos que son específicos del dominio en el que se van a utilizar la aplicación. Por ejemplo, en el caso del ejemplo 2.5.1 del capítulo de estado del arte sobre domótica, hablabamos de edificios, plantas, salas, etc. mientras que en otro dominio podemos hablar de otros conceptos como por ejemplo en la cadena de suministro hablamos de suministrador, punto de venta, paso intermedio, etc. En cualquiera de los dominios en los que planteamos desarrollar aplicaciones para el IoT aparecen conceptos sobre las cosas que nos interesan que son controladas con sensores y modificadas por actuadores. Además, aparecen otros dos elementos: el almacenamiento y la interfaz de usuario. En cualquier dominio es común que nos interese el almacenamiento de la información que se genera sobre estas cosas de interés y que los usuarios quieran poder interactuar sobre las cosas. Para estructurar estos conceptos podemos definir tres elementos fundamentales del dominio (genéricos a cualquier dominio): los objetos o cosas, los medios y los lugares.

Los objetos son cualquier elemento que es de interés en un dominio particular (por ejemplo, en el dominio anterior, las salas, las plantas o los edificios) incluidos los atributos que lo describen (por ejemplo el valor de la temperatura de una sala) y los estados que pueden ser relevantes para una aplicación o un usuario. Los objetos tienen una propiedad denominada *identificación* que los diferencia del resto.

Los medios son la representación conceptual de los sensores, de los actuadores, de los almacenes o de las interfaces de usuario. Estos cuatro tipos de medios se caracterizan por:

- Sensor. Son los tipos de medios que tienen la capacidad de detectar cambios en el entorno. Los sensores tiene la capacidad de percibir una propiedad de un objeto, en el ejemplo, un sensor de temperatura mide la temperatura de una sala.

- Actuador. Este tipo de medio produce cambios sobre el entorno. Los actuadores modifican las propiedades de los objetos con alguna acción.

- Almacén. Los almacenes son tipos de medios que nos permiten guardar de forma permanente información sobre las propiedades de los objetos.

- Interfaz. La interfaz es un tipo de medio que representa cualquier tarea que permite interactuar al usuario con las cosas.

En relación a los lugares nos permiten especificar el sitio en el que se encuentra un objeto, y así disponer de una referencia para relacionar las cosas con su uso en la aplicación. En el ejemplo sobre la domótica, los lugares pueden ser un edificio, una planta del edificio o una sala, o referido a la sala puede ser el techo, el suelo o una pared.

Los conceptos específicos de las funciones describen los elementos que necesitan recursos de computación en la aplicación que definimos para el IoT y las relaciones entre ellos, en otras palabras, aquellas partes que tenemos que programar para resolver el problema. En este caso incluimos tres conceptos relacionados con la realización de las funciones: la lógica, el servicio y el enlace. La lógica refiere a una funcionalidad de un software, por ejemplo si se detecta incendio abrir las llaves del agua del sistema antiincendios. El servicio es el tipo de software que utiliza la información de los medios, la procesa y genera nueva información. Y por último el enlace es el componente software que intercambia información o que controla el intercambio de información.

Los conceptos específicos de la aplicación describen la información sobre los recursos. Cada recurso incluye cero o varios medios. Por ejemplo, un recurso "sala" puede incluir el medio para medir la temperatura con el sensor correspondiente, un termostato para controlar la temperatura, un monitor para mostrar el valor de la temperatura, etc. Cada medio está localizado en un determinado lugar.

Los conceptos específicos de la plataforma son programas que interactuan con los elementos físicos de la plataforma en la que desarrollamos la aplicación. A partir de los conceptos específicos del dominio podemos considerar que los conceptos específicos de la plataforma serían:

- Controlador de sensores. Es el tipo de software que enlaza un sensor con la aplicación. Accede a la información del sensor y produce información que puede ser gestionada por otros componentes software.

- Controlador de actuadores. Es el software que controla y ordena desde una aplicación un actuador. Recibe una orden desde la aplicación y la traduce a los comandos del actuador.

- Controlador de almacen. Es tipo de componente que enlaza la aplicación con los almacenes. Proporciona la capacidad para leer y escribir información. También incluye algún mecanismo de control de acceso para controlar el uso de la información.

- Aplicación de usuario. Es el componente software que se presenta al usuario de la aplicación. Presenta la información e incluye aquellos elementos que permiten al usuario acceder a las tareas que debe tener que realizar con la aplicación.

En los siguientes apartados se utilizan todos estos conceptos específicos para la creación de los lenguajes que conforman el DGL, según los diferentes asuntos de interés en que se estructuran las aplicaciones para el IoT.

4.2.2 DDL: Design Domain Language

Como se ha comentado en el apartado anterior los conceptos del dominio definen los elementos fundamentales del dominio de la aplicación. Estos conceptos del dominio se definen con el lenguaje de diseño del domino o DDL (*Design Domain Language*). El diseño del DDL se ha realizado para permitir que los expertos describan el vocabulario del dominio. En él se han incluido construcciones que permiten especificar los conceptos que se relacionan con los objetos, y que pueden resumirse con el diagrama de la figura 4.1 .

Los conceptos del dominio que forman parte del vocabulario pueden describirse con una tripleta formada por el conjunto de lugares, el conjunto de estructuras y el conjunto de medios.

- Conjunto de lugares. Representa el conjunto de lugares que se utilizan para especificar los emplazamientos de los objetos. Se caracterizan por una descripción y un tipo de lugar. En el ejemplo, podríamos considerar los lugares de nuestro dominio a los edificios, a las plantas y a las salas. Utilizaremos la palabra reservada **wheres** en el DDL.

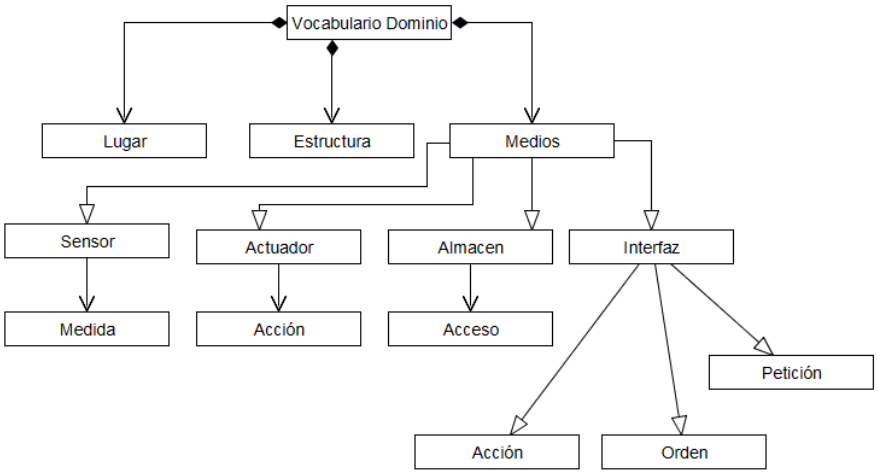

Figura 4.1: Elementos del DDL.

Vocabulario de los conceptos del dominio

- Conjunto de estructuras. Cada medio se caracteriza por los tipos de información que necesitan. El conjunto de estructuras se define con la palabra reservada **datastructs**. Por ejemplo, una estructura para un sensor de temperatura tendrá el valor y las unidades de medida.

- Medios. Define los medios que existen en nuestro dominio. Los medios se definen a partir de la palabra reservada **supports** . Y cada uno de los cuatro tipos que hemos comentado: sensores, actuadores, almacenes o interfaces, incluye:

  - Sensores. Define los tipos de sensores del dominio. Por ejemplo, sensor de luz, sensor de temperatura. Para definir estos sensores utilizamos la palabra clave **sensors**. Cada sensor toma medidas de datos que corresponden con alguna de las estructuras definidas en el conjunto de estructuras. Cada medida del sensor se asocia a una estructura con la palabra reservada **produce**.

  - Actuadores. Define los tipos de actuadores. Por ejemplo, una alarma o un motor. Para definir los actuadores utilizamos la palabra clave **actuators**. Cada actuador realiza una acción que se declara con la palabra reservada **action**. Estas acciones de los actuadores pueden incluir

parámetros de la acción.

- Almacenes. Los almacenes son cualquier medio disponible para almacenar información generada por el usuario o por los sensores. En el DDL consideramos que siempre tenemos información identificada por una clave. La palabra clave para definir los almacenes es **repos**. A los almacenes se accede para realizar alguna operación sobre ellos. Para definir los accesos utilizamos la palabra reservada **accessed-by**.
- Interfaces. Representan un conjunto de tareas disponibles para los usuarios que interactuan con los objetos, por ejemplo, un control de un termostato de temperatura o una señal luminosa de alarma. Las interfaces se definen con la palabra reservada **interfaces**. Las interfaces realizan tres tipos de tareas: peticiones, órdenes y acciones. Las palabras reservadas son **demmand**, **do** y **action**. Las peticiones son solicitudes de datos a otros medios. Las órdenes son los comandos que un usuario puede realizar a un actuador. Y las acciones son las llamadas que realizan los objetos para notificar alguna información a los usuarios.

El resumen de las 13 palabras reservadas del lenguaje son: *accessed-by action actuators datastructs demmand interfaces order produce repos sensors string supports wheres*

Por simplificación se han incluido únicamente los cuatro tipos primitivos: *bool float int string*

Con todos estos elementos definimos la sintaxis del DDL, con las reglas de producción que se incluyen en el anexo A.

Un ejemplo de uso del DDL para el ejemplo del control de edificios que comentamos para los teletrabajadores con la gramática anterior podría ser el contenido mostrado en el siguiente listado:

```
1 wheres =
2   Edificio : int;
3   Planta : int;
4   Sala : int;
5
6 datastructs =
7   LecturaRFID (
```

```
8         identificador : string ,
9         timestamp : int  );
10
11    TemperaturaSala (
12         valor : float ,
13         unidadmedida : string  );
14
15    TeleTrabajador (
16          identificador : string ,
17         codigopersonal : int );
18
19     PerfilPersonal (
20         codigopersonal : int,
21         preferenciatemperatura : TemperaturaSala
22         );
23
24 supports =
25
26    sensors =
27        LectorRFID produce etiquetadetectada : LecturaRFID;
28        SensorTemperatura produce medidatemperatura:
             TemperaturaSala;
29
30    actuators =
31         Termostato
32              action Apagar ( );
33              action PonerTemperatura ( temperatura: TemperaturaSala
                    );
34
35    repos =
36         Teletrabajador produce perfiltemperatura: TemperaturaSala
             accessed-by identificador : string;
37
38    interfaces =
39         Panel
40         [
41             order Apagar ();
42             action MostrarTemperatura ( tempertatura:
                  TemperaturaSala );
43             demmand perfilTemperatura ( identificador );
44         ]
```

```
45          BotonTermostato
46          [
47              action PonerTemperatura ( temperatura: TemperaturaSala
                    );
48          ]
```

A partir de los elementos de los lugares, las estructuras y los medios que expresamos con el DDL podemos definir el diseño funcional de la aplicación con el DFL que se comenta a continuación.

4.2.3 DFL: Design Functional Language

A partir de los conceptos del dominio diseñados con el DDL del apartado anterior disponemos de recursos de definición que nos permiten describir las funciones en términos de interacciones y servicios que se producen entre ellos. En este caso, la definición que realizamos la expresaremos con el lenguaje de diseño funcional que acerca los términos del dominio a términos de implementación de aplicaciones.

Los términos del DDL: lugares, sensores, actuadores, interfaces y almacenes, se convierten en el DFL en posibles valores que manejan los servicios que queremos definir. Cada servicio se expresa en términos de las entradas y de las salidas que maneja. Las entradas pueden ser datos de salida de otros servicios o datos de un sensor. Las salidas pueden ser: los datos creados en el servicio por algún tipo de cálculo, las acciones (a actuadores o interfaces), o peticiones (a almacenes) que se envían desde el servicio.

La relación de estos conceptos se representa con el diagrama de la figura 4.2.

En este caso aparecen dos conceptos nuevos asociados a los servicios: el gasto y el producto. El "gasto" representa a los eventos asociados a notificaciones procedentes de los sensores o a salidas producidas por otros servicios. La palabra reservada en el DFL es **input**. El "producto" representa el conjunto de resultados que se obtienen de las operaciones del servicio con las entradas. La palabra reservada del DFL es **produce**.

En cuanto a las salidas de tipo petición se utilizan para recuperar información de los almacenes definidos en el vocabulario del DDL. La petición representa una solicitud síncrona que devuelve valores. El peticionario envía la petición con algún

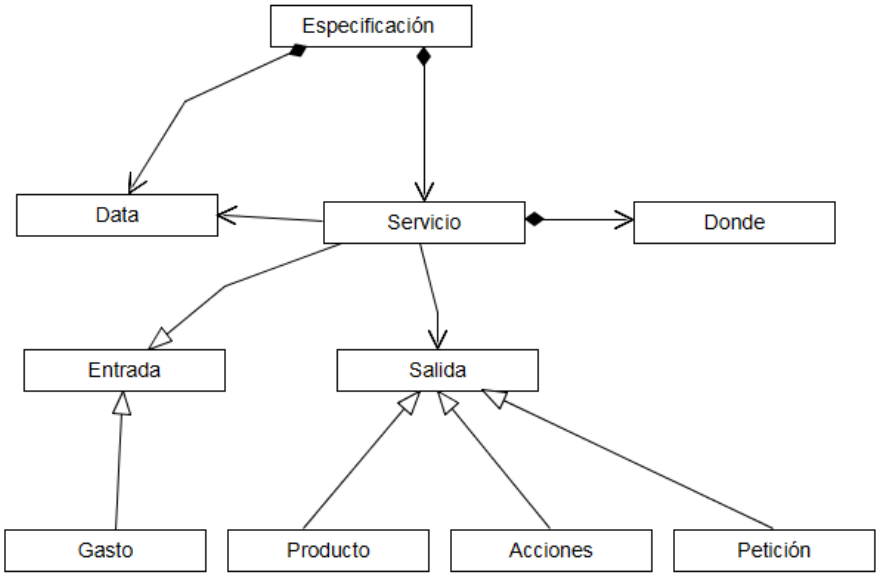

Figura 4.2: Elementos del DFL.

Relaciones de los conceptos funcionales del DGL

tipo de parámetro de acceso para el almacén que responde con la información apropiada a la petición como respuesta. La palabra reservada en este caso es: **demmand**. Las salidas de tipo acción realizadas por un servicio son *triggers* que lanzan acciones sobre interfaces o actuadores. Estas acciones pueden incluir parámetros y estar asociadas a un determinado lugar. La palabra reservada en este caso es: **do**.

El último concepto que se incluye en el DFL es el "donde". Este concepto permite asociar alguno de los lugares del DDL con los servicios de forma que un servicio aplica en el lugar en el que se integra. La palabra reservada en este caso es **in-where**.

Con todos estos elementos definimos la sintaxis del DDL, con las reglas de producción que se incluyen en el anexo B.

A partir del vocabulario del ejemplo anterior de los teletrabajadores, un ejemplo de uso del DFL para el servicio de acceso al edificio podría ser el contenido mostrado en el siguiente listado. Básicamente, se trata de una aplicación que permite que al acceder el teletrabajador al edificio, se consulte el estado de las salas y se le asigne según sus preferencias de temperatura de trabajador aquella que mejor se adecua a su perfil.

```
1  data =
2       Acceso ( trabajador TeleTrabajador ,
3                timestamp int )
4
5  services =
6
7       [ CalculoTemperatura =
8              produce temperaturamedia: medidatemperatura ;
9              input TemperaturaSala origin Sala ;
10             demmand;
11             do;
12             in-where Sala ;
13      ]
14      [ AccesoTeleTrabajador =
15             produce acceso : etiquetadetectada ;
16             input LecturaRFID origin Edificio ;
17             demmand PerfilPersonal ( identificador ) ;
18             do PonerTemperatura ( temperatura ) destination Sala;
19             in-where Edificio ;
20      ]
```

4.2.4 DAL: Design Architecture Language

El tercer lenguaje que está incluido en el DGL es el DAL. Este lenguaje se utiliza para expresar la información relacionada con la instalación de la arquitectura y los dispositivos que maneja. Los principales elementos que se utilizan en este lenguaje permiten describir los atributos de los dispositivos y cómo se vinculan con los servicios que resuelven la aplicación en una instalación. El DAL incluye los medios y los sitios que se han definido en el DDL para determinados atributos de sentencias de la arquitectura de la aplicación. En este caso el concepto fundamental es el de "dispositivo". Un dispositivo en el DAL puede ser cualquier elemento que incluye y gestiona los medios de la aplicación. El dispositivo se define en términos del lugar o lugares en donde se usa, los medios que gestiona y el tipo de dispositivo.

La representación de los conceptos del DAL se muestra en la Fig. 4.3.

En el caso de los dispositivos que se expresan con el DAL tanto los lugares como los medios proceden de las definiciones que se han realizado en el lenguaje

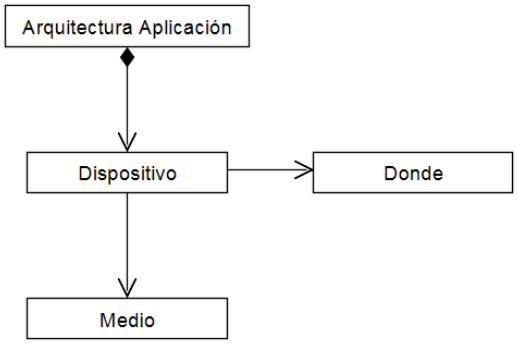

Figura 4.3: Elementos del DAL.

Relaciones de los conceptos del lenguaje para la arquitectura de la aplicación

de dominio y lo que se incorpora es el tipo y la naturaleza. El tipo se utiliza para poder enlazar los servicios que se han definido en el modelo funcional con los entornos concretos en los que debe funcionar la aplicación, por ejemplo valores de este atributo pueden ser: Android, TinyOS, Mantis, Java_SE, Android, Windows Mobile, etc. Este atributo se puede utilizar para relacionar software de conexión o esqueletos de controlador que en cada caso, permiten resolver las diferencias existentes entre las plataformas de funcionamiento.

Con todos estos elementos definimos la sintaxis del DAL, con las reglas de producción que se incluyen en el anexo C. Y un ejemplo a partir de los elementos anteriores sería un dispositivo de gestión de salas que integrara los servicios de control de temperatura sobre un termostato, cálculo de la temperatura media de la sala, mostrar la temperatura del sensor, etc., que se expresaría como se muestra en el siguiente listado.

```
1 devices =
2     [ Gestorsala :
3             where Sala ;
4             type  WindowsMobile ;
5             supports ( SensorTemperatura , Panel , BotonTermostato)
                  ;
6     ]
```

A partir de esta expresión de los dispositivos de la arquitectura de la aplicación, sería necesario como parte del proceso de desarrollo, realizar un mapeo de

los servicios definidos con el DFL en los dispositivos definidos en el DAL. Este proceso de mapeo debe considerar, tanto las definiciones específicas que se han realizado a través de los lugares en donde deben utilizar los servicios y donde se localizan los dispositivos, como los aspectos relacionados con la heterogeneidad de las infraestructuras.

4.3 Proceso de desarrollo con DGL

A partir de la definición del modelo de metaDEPCAS y del lenguaje genérico de diseño de aplicaciones, en este apartado se incluyen los contenidos relacionados con el proceso de uso. Como se comenta en la introducción, una de las carencias actuales es la falta de mecanismos de desarrollo de aplicaciones para el IoT con un planteamiento específico para este entorno. La aplicación del proceso se estructura en torno a los cuatro aspectos que se vinculan en el proceso: los conceptos del dominio, los conceptos funcionales, los conceptos de la arquitectura de la aplicación y los específicos de la plataforma. Los tres primeros se corresponden con todos los elementos incluidos en el DGL, mientras que el cuarto, los elementos de la plataforma son específicos de cada entorno y quedan vinculados desde el DAL y desde el DFL como instancias del diseño realizado.

4.3.1 Introducción al proceso de desarrollo con DGL

La propuesta de proceso incluye tres fases principales: dominio, aplicación e integración que agrupan las funciones principales del proceso de desarrollo y está basada en las ideas que se plantean en [63]. En este artículo se propone la separación de las tareas del proceso de desarrollo en áreas específicas y separadas que trabajan sobre sus propios componentes software y con sus plataformas hardware, para que en una segunda fase se pueda realizar las tareas de integración basadas en abstracciones comunes. Para conseguir esta separación transparente es necesario disponer de los mecanismos de integración que faciliten este proceso, en nuestro caso el modelo con los elementos de metaDEPCAS y del DGL.

Las ocho tareas fundamentales que forman parte del proceso son:

- Especificación del dominio con el DDL. A partir de los conceptos modelados en metaDEPCAS obtenemos la base de los conceptos con los que generamos el vocabulario para el diseño de los elementos de la aplicación con el DDL. Esta tarea pertenece a la fase del dominio.

- Creación de los elementos diseñados con el DDL. Una vez diseñados los elementos del dominio con el DDL podemos procesarlos para obtener los

elementos del dominio que son necesario en el diseño funcional y en el de la arquitectura. Esta tarea pertenece también a la fase del dominio.

- Especificación de las funciones con el DFL. Con las capacidades del DFL y los recursos del dominio tenemos que realizar el diseño de los servicios funcionales de la aplicación. Esta tarea pertenece a la fase de la aplicación.

- Creación de los servicios a partir de la especificación del DFL. Una vez que han sido definidos los servicios por el diseñador, podemos crear los servicios que conforman la aplicación. Esta tarea pertenece a la fase de la aplicación.

- Implementación de la lógica de los servicios. A partir de los servicios construidos, el diseñador puede proponer el conjunto de lógicas específicas de cada servicio que se va a utilizar en la aplicación. Esta tarea pertenece a la fase de la aplicación.

- Especificación de la instalación. A partir de la gramática del DAL, el experto de integración definirá la arquitectura de la aplicación. Esta tarea pertenece a la fase de la aplicación.

- Implementación de *drivers* del entorno. El experto en la realización de los controladores recibirá la información del dominio, que incluye la abstracción de alto nivel de cada dispositivo y podrá incorporar a la base de datos de controladores aquellas operaciones que sean necesarias para la aplicación. Esta tarea pertenece a la fase de la aplicación.

- Integración de los diseños de la aplicación. Por último, y a partir de la realización de los servicios de la aplicación, el mapeo entre estos servicios y la arquitectura, y las interfaces y los controladores específicos, el experto en la integración debe enlazar cada elemento en su dispositivo fragmentando aquellos elementos que sean necesarios.

En el siguiente apartado se comentan todos estos elementos y cómo se relacionan entre ellos.

Perfil	Conoce sobre ..	Participa en..
Experto del dominio	Los elementos que forman el dominio en el que vamos a resolver la aplicación. Es necesario que conozca el entorno de funcionamiento y en particular los elementos sensores y actuadores que se van a utilizar en el dominio.	La realización del diseño con DDL a partir de la que se generarán los conceptos que manejamos en la aplicación.
Diseñador de funciones	Los servicios necesarios para resolver las aplicaciones en el IoT, así como las formas de relacionarlos e integrarlos.	Definir la estructura de los servicios y el desarrollo de la lógica en la aplicación.
Experto Instalación	El desarrollo y la integración de los servicios de la aplicación y los elementos que se manejan en ella.	La integración y la instalación de la aplicación. Tiene capacidad para manejar el DAL e integrar el diseño realizado con la base de datos de controladores disponibles.
Diseñador de los dispositivos	EL desarrollo específico necesario de los diferentes elementos que se integran en el sistema: sensores, actuadores, interfaces, etc.	Desarrollo de controladores e intefaces.

Tabla 4.2: Perfiles del proceso de desarrollo

4.3.2 Elementos del proceso

En el proceso consideramos cuatro perfiles de actores que intervienen: el experto del dominio, el diseñador de las funciones, el experto en la instalación y el diseñador de los dispositivos. En la tabla 4.2 se comentan brevemente cada uno de estos perfiles y cuál es su participación en el proceso.

Cada diseño que se realiza queda asociado a un determinado dominio (la gestión de edificios, la domótica doméstica, la instalación industrial, etc) sobre el que se definen los conceptos específicos con el lenguaje genérico. Una vez definidos los elementos del dominio, cada perfil que interviene en el proceso puede manejar

estos conceptos específicos aplicados a su segmento del desarrollo, pero siempre con la base común. Además, la aproximación de los conceptos de dominio facilita que se pueda disponer de entornos que acumulen conocimiento sobre los mismos y faciliten la reutilización.

Otro aspecto importante es que la separación de las tareas en el proceso ayuda a que los cambios y las evoluciones que se produzcan en las diferentes facetas de la aplicación se puedan aplicar de forma más sencilla. Por ejemplo, los cambios de dispositivos o la incorporación de nuevos recursos con funcionalidades añadidas, o cambios en la red, afectarán a puntos diferentes del proceso que podrán gestionar de forma integral sin afectar a la aplicación completa.

La fase del dominio está formada por:

- Especificación del dominio con el DDL.
- Creación de los elementos diseñados con el DDL.

Podemos expresar con la notación SPEM (*Software Process Engineering Metamodel*) 4.4 los diferentes elementos que están incluidos en esta fase.

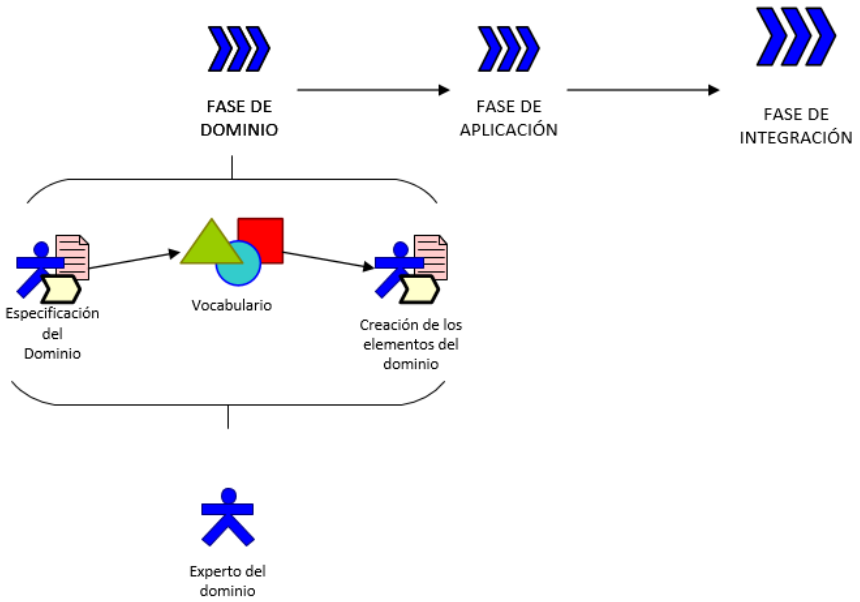

Figura 4.4: Fase de Dominio.

Tareas y productos principales de la fase de dominio

El experto del dominio utiliza los elementos del DDL para especificar el vocabulario del dominio estableciendo las relaciones entre los conceptos. Estos conceptos de alto nivel, facilitan convertir el conocimiento del dominio del experto en un vocabulario, que se puede gestionar para el desarrollo de las aplicaciones.

La creación de los elementos del dominio facilita, primero al diseñador de los dispositivos, poder disponer de un vocabulario común en el dominio; segundo, proporciona al diseñador que maneja los recursos del DFL el conjunto de elementos específicos propios del dominio y que necesita para la especificación de los servicios; y tercero, al experto de la instalación que maneja los recursos del DAL, la gramática de elementos del dominio para el diseño de la arquitectura de la misma.

La fase de la aplicación se divide en tres procesos que se pueden realizar en paralelo a partir de los productos obtenidos de la fase de dominio. Los tres procesos están relacionados con el diseño y realización de las funciones, el diseño de la arquitectura de la aplicación, y los temas relacionados con los controladores de los dispositivos. Las tareas que están incluidas en esta fase son:

- Especificación de las funciones con el DFL.
- Creación de los servicios a partir de la especificación DFL.
- Implementación de la lógica de los servicios.
- Especificación de la instalación.
- Implementación de drivers del entorno.

Podemos expresar con la notación SPEM 4.5 los diferentes elementos que están incluidos en esta fase con los tres bloques de tareas que se realizan en paralelo.

A partir de la gramática del DFL que incorpora el vocabulario del dominio, el diseñador de las funciones debe completar los servicios que forman la aplicación y las relaciones entre ellos. La estructura del DFL está basada en los lenguajes ADL (*Architecture Definition Language*) e incluye la opción de vincular los servicios que se definen a lugares. Así podemos relacionar los dispositivos de un lugar o de un conjunto de lugares con un determinado servicio, de forma que se facilita la escalabilidad de la aplicación.

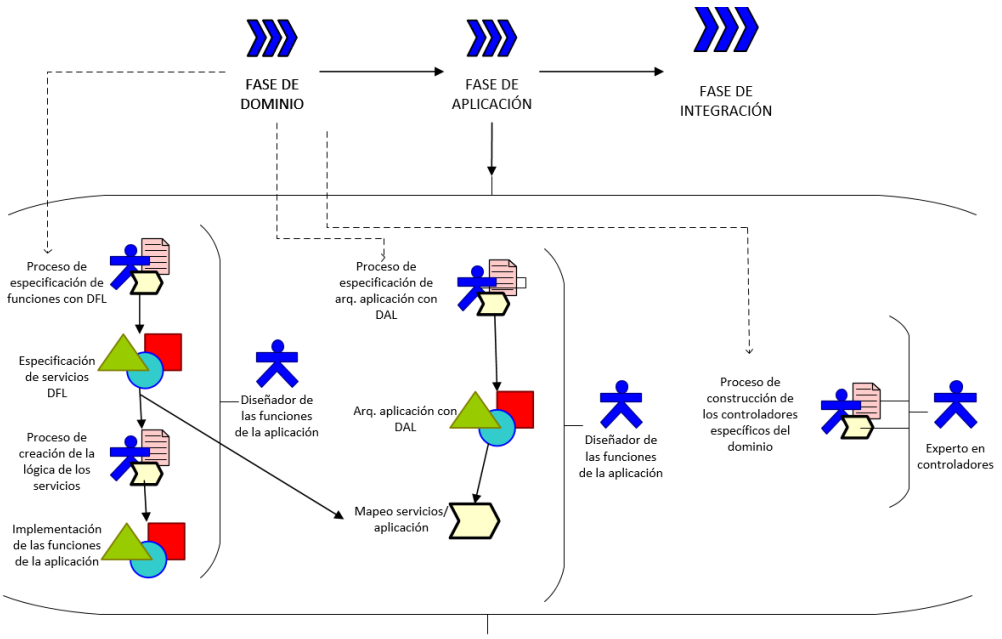

Figura 4.5: Fase de Aplicación.

Tareas y productos principales de la fase de la aplicación

Cada uno de los servicios que se diseñan con el DFL puede convertirse en una clase abstracta de tipo servicio, que oculte las particularidades de la implementación y la lógica del servicio. El desarrollo de esta funcionalidad se incluye como actividad final del proceso relacionado con la definición y realización de la lógica de los servicios. Las clases abstractas que pueden generarse incluyen dos tipos de métodos: operaciones concretas y operaciones abstractas. Los métodos concretos que se incorporan a los servicios permiten establecer las relaciones con otros servicios, mientras que los métodos abstractos encapsulan la lógica de las operaciones.

En paralelo a las anteriores tareas del diseñador, el experto en la integración puede utilizar el lenguaje de definición de la arquitectura y la especificación de las funciones de los servicios para definir con el DAL. Este diseño incluye los detalles de los dispositivos y su referencia al lugar (en función de los lugares del vocabulario del dominio) y al tipo de dispositivo.

La última fase del proceso, la integración, puede representarse con la figura 4.6, en la que se recogen los resultados del diseñador de funciones, del experto en la integración y del experto en dispositivos, y se enlazan para dar resultado al

despliegue de la aplicación.

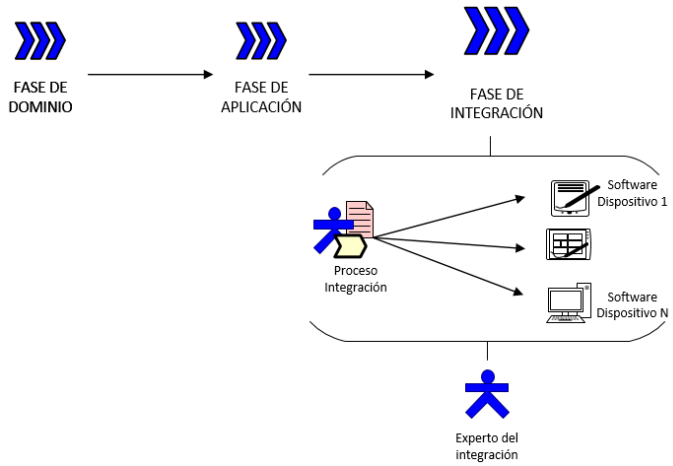

Figura 4.6: Fase de Integración.

Tareas y productos principales de la fase de integración

En este caso, y como soporte a la integración de todos los elementos debemos suponer la existencia de DEPCAS como middleware que adquiere la información y que sirve para unir la identificación y la información sensorial para que pueda ser explotada por los servicios definidos con el DGL.

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Winston Churchill

CAPÍTULO

5

Conclusiones y Trabajos futuros

Este capítulo está dedicado a las conclusiones obtenidas después de la realización de este trabajo de Tesis, así como los resultados que, hasta el momento, se han obtenido de ella. A continuación se enumeran las principales alternativas de investigación y desarrollo que se inician a partir de la realización de este trabajo y que permitirán su mejora y ampliación.

5.1 Conclusiones

El trabajo de Tesis expuesto ha planteado las cuestiones actuales relacionadas con el desarrollo de aplicaciones software para el IoT. A partir de las alternativas más actuales que existen, se ha elaborado una propuesta basada en las arquitecturas de software y una herramienta que la lleva a la práctica, para facilitar la reducción de la complejidad del proceso de desarrollo de software en este entorno de problemas.

Las aplicaciones para el IoT son una tecnología aplicada que está emergiendo y que será, en el futuro más cercano, un tema de amplia cobertura en diferentes ámbitos, no sólo en la investigación, sino también en el desarrollo industrial de software. Como consecuencia, y debido a la complejidad que implica el desarrollo

de software en este entorno, surge la necesidad de plantear soluciones que ayuden a resolver las dificultades particulares. En el marco de esta tesis, se ha propuesto un lenguaje genérico de diseño según las características de los lenguajes específicos de dominio (DSL) que permite no sólo diseñar una aplicación en función de conceptos generales del IoT sino que proporciona una base adecuada e integral para resolver los aspectos de desarrollo más específicos, y que más dificultan la programación de este tipo de aplicaciones.

A diferencia de otras propuestas, la realizada en este trabajo ha incluido dos elementos diferenciados que son propios y particulares. En cuanto al modelado de las aplicaciones para el IoT, el uso de un enfoque basado en arquitecturas de software frente a la tendencia más habitual del uso de ontologías, facilita que se pueda integrar el modelado metaDEPCAS con el lenguaje de diseño genérico, de forma que no es necesario buscar mayor abstracción a través de otros mecanismos de metamodelado. Un segundo aspecto particular es la conceptualización del IoT como un entorno identificado, y en particular identificado por mecanismos automáticos. Esta característica será un aspecto fundamental en el desarrollo de aplicaciones para el IoT en los próximos años y, por lo tanto, las propuestas realizadas en torno a la adquisición de información por RFID como DEPCAS, pueden aportar un contenido relevante.

La integración del DGL y el conjunto de las transformaciones pueden dar lugar a una herramienta que cumpla con las expectativas generadas en torno a la simplificación del proceso de desarrollo de software para el IoT. La estimación inicial de la mejora que puede introducir el uso de desarrollar una aplicación específica para el IoT o utilizar la aproximación basada en el DGL, es de entre cinco a diez veces, en función de la complejidad de la aplicación desarrollada y de los conocimientos y los roles involucrados. Además, se estima que el tiempo necesario para poder usar la aproximación basada en el DGL es significativamente menor que el necesario para poder poder aprender las bases necesarias para desarrolos mínimos de aplicaciones en el IoT.

5.2 Trabajos Futuros

A consecuencia de las características del problema que se plantea en este trabajo de Tesis no todos los aspectos involucrados han sido tratados con igual atención, y nos hemos centrando en los que se han considerado fundamentales y básicos para la propuesta. Así, se ha dado prioridad a la descripción del modelo y los aspectos relacionados con la adquisición de información sensorial autoidentificada, y la descripción de aplicaciones con el DGL como aproximación multi-paradigmática y con el objetivo de dar lugar a un resultado ejecutable, y que faciliten el proceso de desarrollo propuesto. A partir de esta base, pueden surgir distintos aspectos que quedan abiertos y que plantean cuestiones que esperamos continuar resolviendo en el futuro en el marco de los proyectos de investigación en los que participamos.

A continuación se comentan algunos de estos aspectos:

- Elaboración de las guías de metaDEPCAS. Según la orientación de la propuesta incluida en metaDEPCAS, para obtener las arquitecturas específicas es necesario disponer de la colección de guías que nos permitan convertir el modelo de referencia a la arquitectura de referencia, y desde esta arquitectura de referencia obtener el diseño con el DGL. Estas guías, incluyen los elementos necesarios para facilitar al arquitecto de la aplicación disponer de recursos adicionales que permiten adaptar el modelo a las circunstancias de un entorno particular, como por ejemplo puede ser un entorno M2M, un entorno H2M, etc.

- Extracción de los conceptos del modelo de metaDEPCAS sobre el DGL. El resultado del modelado con metaDEPCAS aporta una estructura de arquitectura que debe utilizarse por los expertos del dominio para extraer la información inicial que se utiliza en el DDL. En la medida que esta extracción pueda automatizarse, la posibilidad de utilización de los conceptos más abstractos del dominio será más sencilla en el desarrollo de las aplicaciones para el IoT.

- Generadores automáticos de código para los servicios diseñados con el DFL y la arquitectura con el DAL. Los productos finales que obtenemos tanto del

DFL y el del DAL son diseños que facilitan de forma destacable la generación automática de código. En este momento se pueden aplicar dos opciones: una es establecer un contexto o entorno operativo que se comenta en metaDEPCAS y que determina cómo debe realizarse esta generación, y otra opción es dirigir la generación a entornos de prueba o simulación.

- Tratamiento de los recursos específicos de plataformas. Un elemento adicional sobre el que se pueden desarrollar proyectos de investigación es la resolución de los controladores de los dispositivos y su integración con las aplicaciones y con el sistema de adquisición de DEPCAS. En los próximos años se van a continuar desarrollando nuevos dispositivos con entornos más o menos propietarios, y de características heterogéneas, por lo que es imprescindible buscar aportaciones para resolver de forma uniforme los desarrollos que quieran utilizar estos dispositivos.

- Herramienta para el desarrollo con el DGL. Mediante una herramienta se espera poder dar soporte a las tareas de los perfiles del proceso y automatizar aquellos elementos que faciliten su trabajo. El primer paso consistiría en generar automáticamente, desde los diseños del lenguaje de dominio, las estructuras que se utilizan en el lenguaje de diseño de las funciones y de la arquitectura. También sería necesario disponer de algún mecanismo que permitiera enlazar los elementos del lenguaje de funciones con la arquitectura, a través del mapeo de los servicios y los dispositivos. Un último paso sería facilitar la integración y las operaciones de enlace entre los resultados del DGL, los controladores, y la adquisición identificada con DEPCAS.

- Modelización completa de todo el proceso de desarrollo basada en SPEM. El proceso de desarrollo de aplicaciones para el IoT debe especificarse de forma completa indicando todos los elementos que se deben utilizar. El uso de un modelo de proceso basado en el metamodelo SPEM garantiza el uso de conceptos y recursos ampliamente conocidos y facilita su aplicación en las organizaciones interesadas en el uso del proceso.

- Tratamiento de las interfaces de usuario en el IoT como recurso del DGL. Los elementos incorporados con el DGL tratan los elementos de las aplicaciones

del IoT: vocabulario, servicios y arquitectura. Algunos de los conceptos del vocabulario que se definen están relacionados con partes o con operaciones procedentes de interfaces de usuario, como por ejemplo: teclados de acceso, botones de alarma; sin embargo no se incluye la opción de qué aplicación del IoT realice o modifique la interfaz del usuario propiamente dicha. Por este motivo, podría integrarse dentro de los recursos del DGL, un lenguaje de tipo UIDL (*User Interface Design Language*) que facilitase al perfil del integrador el desarrollo de interfaces específicas para las plataformas destino sobre las que tuvieran que operar las aplicaciones.

- Desarrollo de los elementos de escalabilidad con el RRTL de DEPCAS y su integración en el DGL. Uno de los aspectos más relevante de DEPCAS es la capacidad para resolver adquisición en diferentes entornos de escalabilidad: locales, en red, distribuidos, por red pública, etc. Esta característica es propia, no sólo de la adquisición de información, sino que también se encuentra en las aplicaciones del IoT como se ha comentado en este trabajo. En este sentido, existe la posibilidad de integrar en el ámbito del DGL recursos que faciliten el desarrollo multiescalar de las aplicaciones y el uso distribuido de la información.

Bibliografía

[1] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991. 1

[2] Kevin Ashton. MIT Auto-ID Lab, http://autoid.mit.edu/ , 1999. 1, 13

[3] International Telecommunication Union. ITU. *The Internet of Things*. ITU, 2005. 1

[4] Christian Floerkemeier, Marc Langheinrich, Elgar Fleisch, Friedemann Mattern, and Sanjay E. Sarma. The internet of things, first international conference, IoT 2008, zurich, switzerland, march 26-28, 2008. proceedings, 2008. 1

[5] Dave Evans. The internet of things: How the next evolution of the internet is changing everything, 2011. 1

[6] Rose Schooler. The internet of things: Exploring the next technology frontier. *United States House of Representatives.*, 2015. 1

[7] ISO/IEC JTC 1. Internet of things (iot). preliminary report. Technical report, International Standard Organization (ISO), 2014. 2

[8] Rolf H Weber and Romana Weber. *Internet of Things*. Springer, 2010. 2

[9] International Telecommunication Union. ITU. Serie Y: Descripcion general de internet de los objetos. Technical report, International Telecommunication Union (ITU), 2014. 2

BIBLIOGRAFÍA

[10] Jan Newmarch and Paulo Tam. Issues in ownership of internet objects, 2002. 2

[11] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelffle. *Vision and Challenges for Realising the Internet of Things*. European Commision, 2010. 2

[12] Jean-Philippe Vasseur and Adam Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010. 4

[13] Mingfu Li and Hung-Ju Lin. Design and implementation of smart home control systems based on wireless sensor networks and power line communications. *Industrial Electronics, IEEE Transactions on*, 62(7):4430–4442, July 2015. 4

[14] Aravind Kailas, Valentina Cecchi, and Arindam Mukherjee. A survey of communications and networking technologies for energy management in buildings and home automation, 2012. 4

[15] Dixon, Colin, Ratul Mahajan, Sharad Agarwal, AJ Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl. An operating system for the home, 2012. 4

[16] NSF. Funding Report 2008, http://www.nsf.gov/ , 2008. 4

[17] WhiteHouse. President's Council of Advisor and Technology, http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-nitrd-report-2010.pdf , 2010. 5

[18] J. Buckley. From rfid to the internet of things: pervasive networked systems. Final Report on the Conference organised by DG Information Society and Media, Networks and Communication Technologies Directorate, 2006. 5

[19] CERP. Internet of Things: Strategic Research Agenda, http://ec.europa.eu/ , 2009. 5

[20] L. Atzori, A. Iera, F. Morabito, and M. Nitti. The social internet of things (siot) - when social networks meet the internet of things: concept, architecture and network characterization., 2012. 5

[21] K.A. Hribernik, Z. Ghrairi, C. Hans, and K. Thoben. Co-creating the internet of things: first experiences in the participatory design of intelligent products with arduino, 2011. 5

[22] D. Miorandi, S. Sicari, F. De Pellegrini, and Chlamtac I. Internet of things: Visiom, applications and reserach challenges. Ad Hoc Networks, 2012. 5

[23] P. Patel, A. Pathak, T. Teixeira, and V. Issarny. Towards application development for the internet of things. Proceedings of the 8th Middleware Doctoral Symposium. ACM, 2011. 6

[24] I. Abad, R. Heradio, C. Cerrada, and J.A. Cerrada. A scada oriented middleware for rfid technology. Expert Systems with Applications, 2012. 7

[25] Julien Mineraud, Oleksiy Mazhelis, Xiang Su, and Sasu Tarkoma. Contemporary internet of things platforms. *CoRR*, abs/1501.07438, 2015. 16

[26] C. Perera, C.H. Liu, and S. Jayawardena. The emerging internet of things marketplace from an industrial perspective: A survey. *Emerging Topics in Computing, IEEE Transactions on*, PP(99):1–1, 2015. 16

[27] HP-EMEA. Big data, 2013. 17

[28] Annabelle Gawer, Michael A Cusumano, and Developing Strategy Strategy. How companies become platform leaders. 2012. 18

[29] Marc Langheinrich. A survey of rfid privacy approaches. *Personal and Ubiquitous Computing*, 13(6):413–421, 2009. 19

[30] Daniel M Dobkin. *The RF in RFID: UHF RFID in Practice*. Newnes, 2012. 20

[31] Beecham-Research. M2m sector map. Reports & downloads, 2011. 24

[32] J. Landt. The history of rfid. *Potentials, IEEE*, 24(4):8–11, Oct 2005. 25

[33] Ari Juels. Rfid security and privacy: A research survey. *Selected Areas in Communications, IEEE Journal on*, 24(2):381–394, 2006. 25

[34] Yan Zhang, Rong Yu, Shengli Xie, Wenqing Yao, Yang Xiao, and Mohsen Guizani. Home m2m networks: architectures, standards, and qos improvement. *Communications Magazine, IEEE*, 49(4):44–52, 2011. 27, 31

[35] V.P. Kafle, Y. Fukushima, and H. Harai. New mobility paradigm with id/locator split in the future internet. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pages 163–169, Jan 2014. 27

[36] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008. 27

[37] Belur V. Dasarathy. Information fusion special issues - an overview. *Information Fusion*, 14(3):227–228, 2013. 28

[38] David L Hall and James Llinas. *Handbook of multisensor data fusion*. Boca Raton, FL : CRC Press, 2001. 28

[39] G. Gate, A. Breheret, and F. Nashashibi. Centralized fusion for fast people detection in dense environment. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 76–81, May 2009. 28

[40] V. Vaidehi, K. Kalavidya, and S.Indira Gandhi. Cluster-based centralized data fusion for tracking maneuvering targets using interacting multiple model algorithm. *Sadhana*, 29(2):205–216, 2004. 28

[41] Michael Munz, Klaus C. J. Dietmayer, and M. Mahlisch. A sensor independent probabilistic fusion system for driver assistance systems. In *International Conference on Intelligent Transportation*, 2009. 28

[42] E. Zervas, A. Mpimpoudis, C. Anagnostopoulos, O. Sekkas, and S. Hadjiefthymiades. Multisensor data fusion for fire detection. *Inf. Fusion*, 12(3):150–159, July 2011. 28

[43] D. Bickson, Y. Tock, A. Zymnis, S. Boyd, and D. Dolev. Distributed large scale network utility maximization. In *In the International symposium on information theory (ISIT)*, pages 829–833, July 2009. 28

[44] Federico S Cattivelli, Cassio G Lopes, and Ali H Sayed. Diffusion strategies for distributed kalman filtering: formulation and performance analysis. *Proceedings of the Cognitive Information Processing*, pages 36–41, 2008. 28

[45] GS1. Epcglobal: Gs1's innovation network v 1.6, 2014. 29

[46] GS1. Epc information services - v 1.1, 2013. 29

[47] Loc Ho, Melody Moh, Zachary Walker, Takeo Hamada, and Ching-Fong Su. A prototype on rfid and sensor networks for elder healthcare: progress report. In *Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, pages 70–75. ACM, 2005. 33

[48] Louie F Cervantes, Young-Seok Lee, Hyunho Yang, and Jaewan Lee. A hybrid middleware for rfid-based parking management system using group communication in overlay networks. In *Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on*, pages 521–526. IEEE, 2007. 33

[49] Zhou Xiaoguang and Long Wei. The research of network architecture in warehouse management system based on rfid and wsn integration. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 2556–2560. IEEE, 2008. 33

[50] Inácio Fonseca and Fernando Lopes. An embedded system and an architecture for access control and access management: An application to the educational environment. In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*, pages 1–5. IEEE, 2010. 33

[51] Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010. 34

[52] Ovidiu Vermesan and Peter Friess. *Internet of Things-Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT*. River Publishers, 2011. 34

[53] Nicola Guarino, Massimiliano Carrara, and Pierdaniele Giaretta. An ontology of meta-level categories. *KR*, 94:270–280, 1994. 35

[54] David J Russomanno, Cartik R Kothari, and Omoju A Thomas. Building a sensor ontology: A practical approach leveraging iso and ogc models. In *IC-AI*, pages 637–643, 2005. 35

[55] Michael Compton, Holger Neuhaus, Kerry Taylor, and Khoi-Nguyen Tran. Reasoning about sensors and compositions. In *SSN*, pages 33–48. Citeseer, 2009. 35

[56] Suparna De, Tarek Elsaleh, Payam Barnaghi, and Stefan Meissner. An internet of things platform for real-world and digital objects. *Scalable Computing: Practice and Experience*, 13(1), 2012. 35

[57] Sefki Kolozali, Maria Bermudez-Edo, Daniel Puschmann, Frieder Ganz, and Payam Barnaghi. A knowledge-based approach for real-time iot data stream annotation and processing. In *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE*, pages 215–222. IEEE, 2014. 35

[58] Michael Blackstock and Rodger Lea. Wotkit: a lightweight toolkit for the web of things. In *Proceedings of the Third International Workshop on the Web of Things*, page 3. ACM, 2012. 36

[59] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. In *Mobile Data Management, 2007 International Conference on*, pages 198–205, May 2007. 36

[60] Nan Li, Min Liu, and Jun-Wei Yan. Framework for industrial internet of things oriented to steel continuous casting plant mro. *Computer Integrated Manufacturing Systems*, 17(2):413–418, 2011. 37

[61] G. Blair and P. Grace. Emergent middleware: Tackling the interoperability problem. *Internet Computing, IEEE*, 16(1):78–82, Jan 2012. 42

[62] M. Kessis, C. Roncancio, and A. Lefebvre. Dasima: A flexible management middleware in multi-scale contexts. In *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, pages 1390–1396, April 2009. 42

[63] Chao Chen, Sumi Helal, Scott de Deugd, Andy Smith, and Carl K. Chang. Toward a collaboration model for smart spaces. In *Proceedings of the Third International Workshop on Software Engineering for Sensor Network Applications*, SESENA '12, pages 37–42, Piscataway, NJ, USA, 2012. IEEE Press. 43, 110

[64] E. Serral, P. Valderas, J. Munoz, and V. Pelechano. Towards a model driven development of context-aware systems for ami environments. In *Developing Ambient Intelligence*, pages 114–124. Springer Paris, 2008. 43

[65] N. Rozanski and E. Woods. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2011. 50, 53, 64

[66] Stephan Haller. The things in the internet of things. *Poster at the (IoT 2010). Tokyo, Japan, November*, 5:26, 2010. 56

[67] A. Serbanati, Carlo M. Medaglia, and Ugo Biader Ceipidor. Building blocks of the internet of things. 2011. 56

[68] Kay Romer, Friedemann Mattern, Thomas Dubendorfer, and Jurg Senn. Infrastructure for virtual counterparts of real world objects. Technical report, 2002. 56

[69] Peter F Brown and Rebekah Metz Booz Allen Hamilton. Reference model for service oriented architecture 1.0, 2006. 61

[70] E. Woods and N. Rozanski. The system context architectural viewpoint. In *Software Architecture, 2009 European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*, pages 333–336, Sept 2009. 63

[71] ISO/IEC JTC 1. *ISO/IEC 14443. identification cards — contactless integrated circuit(s) cards — proximity card*, 2014. 74

[72] GS1. Epcglobal: Gs1's tag data standard 1.9, 2014. 74

[73] ISO/IEC JTC 1. *ISO/IEC 15693. identification cards — contactless integrated circuit(s) cards — vicinity card*, 2006. 74

[74] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, Inc., New York, NY, USA, 2 edition, 2003. 74

[75] Thomas Weishaupl and Erich Schikuta. Open language approach for dynamic service evolution. In Hai Jin, Yi Pan, Nong Xiao, and Jianhua Sun, editors, *Grid and Cooperative Computing - GCC 2004 Workshops*, volume 3252 of *Lecture Notes in Computer Science*, pages 132–139. Springer Berlin Heidelberg, 2004. 76

[76] Thomas Buchholz and Michael Schiffers. Quality of context: What it is and why we need it. In *In Proceedings of the 10th Workshop of the OpenView University Association: OVUA?03*, 2003. 88

[77] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, June 2000. 90, 92

[78] C. Jones. Programming language table, release 8.2, Marzo 1996. 91

[79] David S. Wile. Supporting the DSL spectrum. *Journal of Computing and Information Technology*, 9(4):263–287, 2001. 92

[80] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, December 2005. 94, 95

[81] Juha-Pekka Tolvanen. Creating domain-specific modelling languages that work: Hands-on. In *Modelling Foundations and Applications - 7th European Conference, ECMFA 2011, Birmingham, UK, June 6 - 9, 2011 Proceedings*, pages 393–394, 2011. 95

[82] Nuno Oliveira, Maria Joao Pereira, Pedro Henriques, and Daniela Cruz. Domain specific languages: A theoretical survey. 2009. 96

[83] Parastoo Mohagheghi and A. Haugen. Evaluating domain-specific modelling solutions. In Juan Trujillo, Gillian Dobbie, Hannu Kangassalo, Sven Hartmann, Markus Kirchberg, Matti Rossi, Iris Reinhartz-Berger, Esteban Zimányi, and Flavius Frasincar, editors, *Advances in Conceptual Modeling and Applications and Challenges*, volume 6413 of *Lecture Notes in Computer Science*, pages 212–221. Springer Berlin Heidelberg, 2010. 96

APÉNDICE

A

Sintaxis DLL

Sintaxis

```
Program : Wheres DataStructs Supports

Wheres : wheres = Where

Where : IdentifierCAP : Data_type ;
      | IdentifierCAP : Data_type ; Where

DataStructs : datastructs = DataStruct

DataStruct : DeclareStruct DataStruct
           | DeclareStruct

DeclareStruct : IdentifierStruct ( DescriptionStruct ) ;

DescriptionStruct : DescriptionS , DescriptionStruct
                  | DescriptionS

DescriptionS : Identifier : Data_type

IdentifierStruct : IdentifierCAP

Supports : supports = Support

Support : Sensors Actuators Repos Interfaces
        | Sensors Actuators
        | Sensors Actuators Repos
        | Sensors Actuators Interfaces

Sensors : sensors = Sensor

Sensor : IdentifierCAP SensorDescription ; Sensor
       | IdentifierCAP SensorDescription ;

SensorDescription : produce Identifier : IdentifierCAP

Actuators : actuators = Actuator

Actuator : IdentifierCAP Action ;
         | IdentifierCAP Action ; Actuator

Action : action IdentifierCAP
       | action IdentifierCAP ( Parameters )

Parameters : Identifier : IdentifierCAP
           | Identifier : IdentifierCAP , Parameters
```

```
          | null

Repos : repos = Repo

Repo : IdentifierCAP RepoDescription ;
     | IdentifierCAP RepoDescription ; Repo

RepoDescription : GenerationDescription accessed-by IndexDefinition

GenerationDescription : produce Identifier : IdentifierCAP

IndexDefinition : Identifier : Data_type

Interfaces : interfaces = Interface

Interface : IdentifierCAP [ InterfaceDescription ] Interface
          | IdentifierCAP [ InterfaceDescription ]

InterfaceDescription : InterfaceOpt InterfaceDescription
                     | InterfaceOpt

InterfaceOpt : Order
             | Do
             | Demmand

Order : order IdentifierCAP ( ParameterOrder ) ;

ParameterOrder : IdentifierCAP , ParameterOrder
               | IdentifierCAP
               | null

Do : Action ; Do
   | Action ;

Demmand : demmand Identifier ( ParameterDemmand ) ;

ParameterDemmand : Identifier , ParameterDemmand
                 | Identifier

Data_type : int
          | bool
          | string
          | float
          | IdentifierStruct
```

Gramática

```
"Name"     = 'DDL'
"Author"   = 'ISMAEL ABAD'
"Version"  = '0.1'
"About"    = 'Lenguaje de Diseño de Dominio para el IoT'
"Case Sensitive" = True
```

```
"Start Symbol" = <Program>



! -------------------------------------------------
! Conjuntos de carácteres
! -------------------------------------------------

{String Chars} = {Printable} + {HT} - ["\]
{LetterCAP}   = { #65 .. #90 }
{LetternoCAP} = { #97 .. #122 }

! -------------------------------------------------
! Símbolos Terminales
! -------------------------------------------------

StringLiteral = '"' ( {String Chars} | '\' {Printable} )* '"'
IdentifierCAP =  {LetterCAP}{Letter}*
Identifier    = {LetternoCAP}{Letter}*


! -------------------------------------------------
! Reglas
! -------------------------------------------------


<Program> ::= <Wheres> <DataStructs> <Supports>

! Reglas WHERE
<Wheres> ::= wheres '=' <Where>

<Where>  ::= IdentifierCAP ':' < Data_type> ';'
         | IdentifierCAP ':' < Data_type> ';' <Where>

! Reglas DATASTRUCTS
<DataStructs> ::=  datastructs '=' <DataStruct>


<DataStruct> ::=  <DeclareStruct> <DataStruct>
              |  <DeclareStruct>


<DeclareStruct> ::=  IdentifierStruct  '(' <DescriptionStruct> ')' ';'

<DescriptionStruct> ::= <DescriptionS> ',' <DescriptionStruct>
                    |  <DescriptionS>

<DescriptionS> ::= Identifier ':' <Data_type>

<IdentifierStruct> ::= IdentifierCAP

! Reglas SUPPORTS
<Supports> ::=  supports '=' <Support>

<Support>  ::=  <Sensors> <Actuators> <Repos> <Interfaces>
            |  <Sensors> <Actuators>
            |  <Sensors> <Actuators> <Repos>
            |  <Sensors> <Actuators> <Interfaces>

! Reglas SENSORES
<Sensors>  ::= sensors '=' <Sensor>

<Sensor>   ::= IdentifierCAP <SensorDescription> ';' <Sensor>
            | IdentifierCAP <SensorDescription> ';'

<SensorDescription> ::= produce Identifier ':' IdentifierCAP

! Reglas ACTUADORES
<Actuators> ::= actuators '=' <Actuator>

<Actuator> ::= IdentifierCAP <Action> ';'
             | IdentifierCAP <Action> ';' < Actuator>

<Action> ::= action IdentifierCAP
          | action IdentifierCAP '(' <Parameters> ')'
```

```
<Parameters> ::= Identifier ':' IdentifierCAP
          | Identifier ':' IdentifierCAP ',' <Parameters>
          |

! Reglas ALMACENES
<Repos> ::=  repos '=' <Repo>

<Repo> ::= IdentifierCAP <RepoDescription> ';'
        | IdentifierCAP <RepoDescription> ';' <Repo>

<RepoDescription> ::=  <GenerationDescription> accessed-by <IndexDefinition>

<GenerationDescription> ::= produce Identifier ':' IdentifierCAP

<IndexDefinition> ::= Identifier ':' <Data_type>

! Reglas INTERFACES
<Interfaces> ::= interfaces '=' <Interface>

<Interface> ::= IdentifierCAP '[' <InterfaceDescription> ']' <Interface>
             | IdentifierCAP '[' <InterfaceDescription> ']'

<InterfaceDescription> ::=  <InterfaceOpt> <interfaceDescription>
                         | <InterfaceOpt>

<InterfaceOpt> ::= <Order> | <Do> | <Demmand>

<Order> ::=  order IdentifierCAP '(' <ParameterOrder> ')' ';'


<ParameterOrder> ::= identifierCAP ',' <ParameterOrder>
                  | identifierCAP
                  |

<Do> ::= <Action> ';' <Do>
      | <Action> ';'

<Demmand> ::=  demmand Identifier '(' <ParameterDemmand> ')' ';'

<ParameterDemmand> ::= Identifier ',' <ParameterDemmand>
                    | Identifier


! Tipos
<Data_type> ::= int | bool | string | float | <IdentifierStruct>
```

APÉNDICE

B

Sintaxis DFL

Sintaxis

```
ProgramFunctions : Data ServicesBlock

Data : data = Tables

Tables : Table
       | Table Tables

Table : TableName ( Fields )

TableName : IdentifierCAP

Fields : Identifier Data_type
       | Identifier Data_type , Fields

ServicesBlock : services = Services

Services : [ Service ]
         | [ Service ] Services

Service : IdentifierCAP = ProductionService InputService DemmandService DoService WhereService ;

ProductionService : Product ;
                  | Product ; ProductionService

Product : produce Identifier : Domain_vocabulary
        | produce

InputService : Input ;
             | Input ; InputService

Input : input Domain_datastruct origin Domain_wheres

DemmandService : Demmand ;
               | Demmand ; DemmandService

Demmand : demmand Domain_datastruct ( DemmandParameters )
        | demmand

DemmandParameters : DemmandParameter
                  | DemmandParameter , DemmandParameters

DoService : Do ;
          | Do ; DoService

Do : do Domain_action ( DoParameters ) destination Domain_wheres
```

```
   | do

DoParameters : DoParameter
             | DoParameter , DoParameters

WhereService : in-where = Domain_wheres

Domain_vocabulary : medidatemperatura
                  | etiquetadetectada
                  | perfiltemperatura

Domain_datastruct : TemperaturaSala
                  | LecturaRFID
                  | TeleTrabajador
                  | PerfilPersonal

Domain_wheres : Edificio
              | Planta
              | Sala

Domain_action : Apagar
              | PonerTemperatura
              | MostrarTemperatura

DoParameter : temperatura

DemmandParameter : identificador

Data_type : int
          | bool
          | string
          | float
          | TableName
          | Domain_datastruct
```

Gramática

```
"Name"     = 'DFL'
"Author"   = 'ISMAEL ABAD'
"Version"  = '0.1'
"About"    = 'Lenguaje de Diseño de Funciones para el IoT'
"Case Sensitive" = True
"Start Symbol" = <ProgramFunctions>



! -------------------------------------------------
! Conjuntos de carácteres
! -------------------------------------------------

{String Chars} = {Printable} + {HT} - ["\]
{LetterCAP}    = { #65 .. #90 }
{LetternoCAP} = { #97 .. #122 }

! -------------------------------------------------
! Símbolos Terminales
```

```
! -------------------------------------------------

StringLiteral = '"' ( {String Chars} | '\' {Printable} )* '"'
IdentifierCAP =  {LetterCAP}{Letter}*
Identifier    = {LetternoCAP}{Letter}*


! -------------------------------------------------
! Reglas
! -------------------------------------------------


<ProgramFunctions> ::= <Data> <ServicesBlock>

! Reglas DATOS
<Data> ::= data '=' <Tables>

<Tables> ::= <Table>
          | <Table> <Tables>

<Table> ::=  <TableName> '(' <Fields> ')'

<TableName> ::= IdentifierCAP

<Fields> ::= Identifier <Data_type>
          | Identifier <Data_type> ',' <Fields>

! Reglas SERVICIOS
<ServicesBlock> ::=  services '=' <Services>

<Services> ::= '[' <Service> ']'
            | '[' <Service> ']' <Services>

<Service> ::= IdentifierCAP '=' <ProductionService> <InputService> <DemmandService> <DoService> <WhereService> ';'

<ProductionService> ::= <Product> ';'
                      | <Product> ';' <ProductionService>

<Product> ::= 'produce' identifier ':' <Domain_vocabulary>
           | 'produce'

<InputService> ::= <Input> ';'
                | <Input> ';' <InputService>

<Input> ::=  'input' <Domain_datastruct> 'origin' <Domain_wheres>

<DemmandService> ::= <Demmand> ';'
                   | <Demmand> ';' <DemmandService>

<Demmand> ::= 'demmand' <Domain_datastruct> '(' <DemmandParameters> ')'
           | 'demmand'

<DemmandParameters> ::= <DemmandParameter>
                      | <DemmandParameter> ',' <DemmandParameters>

<DoService> ::= <Do> ';'
             | <Do> ';' <DoService>


<Do> ::= 'do' <Domain_action> '(' <DoParameters> ')' 'destination' <Domain_wheres>
      | 'do'

<DoParameters> ::= <DoParameter>
                | <DoParameter> ',' <DoParameters>

<WhereService> ::= 'in-where' '=' <Domain_wheres>



!Reglas procedentes del diseño del DOMINIO con el DLL
!Específicas del dominio de la aplicación para acceso de teletrabajadores
<Domain_vocabulary> ::= 'medidatemperatura' | 'etiquetadetectada' | 'perfiltemperatura'

<Domain_datastruct> ::= 'TemperaturaSala' | 'LecturaRFID' | 'TeleTrabajador' | 'PerfilPersonal'

<Domain_wheres> ::= 'Edificio' | 'Planta' | 'Sala'

<Domain_action> ::= 'Apagar' | 'PonerTemperatura' | 'MostrarTemperatura'
```

<DoParameter> ::= 'temperatura'

<DemmandParameter> ::= 'identificador'

! Tipos
<Data_type> ::= int | bool | string | float | <TableName> | <Domain_datastruct>

APÉNDICE

C

Sintaxis DAL

Sintaxis

```
App : DeviceSet

DeviceSet : devices = Devices

Devices : Device
        | Device Devices

Device : [ DeviceName : DeviceDescription ]

DeviceName : IdentifierCAP

DeviceDescription : Wheres DeviceKind Supports

Wheres : where DomainWheres ;

DeviceKind : type IdentifierCAP ;

Supports : supports ( DomainDevices ) ;

DomainDevices : DomainDevice
              | DomainDevice , DomainDevices

DomainWheres : Edificio
             | Planta
             | Sala

DomainDevice : LectorRFID
             | SensorTemperatura
             | Panel
             | BotonTermostato
```

Gramática

```
"Name"     = 'DAL'
"Author"   = 'ISMAEL ABAD'
"Version"  = '0.1'
"About"    = 'Lenguaje de Diseño de Arquitectura de Aplicaciones para el IoT'
"Case Sensitive" = True
"Start Symbol" = <App>


! -------------------------------------------------
! Conjuntos de carácteres
! -------------------------------------------------

{String Chars} = {Printable} + {HT} - ["\]
{LetterCAP}    = { #65 .. #90 }
```

```
{LetternoCAP} = { #97 .. #122 }

! -------------------------------------------------
! Símbolos Terminales
! -------------------------------------------------

StringLiteral = "" ( {String Chars} | '\' {Printable} )* ""
IdentifierCAP =  {LetterCAP}{Letter}*
Identifier    = {LetternoCAP}{Letter}*


! -------------------------------------------------
! Reglas
! -------------------------------------------------


<App> ::= <DeviceSet>

! Reglas Devices
<DeviceSet> ::= devices '=' <Devices>

<Devices> ::= <Device>
          | <Device> <Devices>

<Device> ::=  '[' <DeviceName> ':' <DeviceDescription> ']'

<DeviceName> ::= IdentifierCAP

<DeviceDescription> ::= <Wheres> <DeviceKind> <Supports>

<Wheres> ::= where <DomainWheres> ';'

<DeviceKind> ::= type IdentifierCAP ';'

<Supports> ::= supports '(' <DomainDevices>  ')' ';'

<DomainDevices> ::= <DomainDevice>
                  | <DomainDevice> ',' <DomainDevices>


!Reglas procedentes del diseño del DOMINIO con el DLL

<DomainWheres> ::= Edificio | Planta | Sala

<DomainDevice> ::= LectorRFID | SensorTemperatura | Panel | BotonTermostato
```