Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

---

Programa de Posgrado en Ciencias en Ciencias de la Computación

---

Incorporación de capacidades semánticas en el Internet de las Cosas.

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Marco Antonio Calderón Pérez

Ensenada, Baja California, México

2016

Tesis defendida por

Marco Antonio Calderón Pérez

y aprobada por el siguiente Comité

_____

Dr. José Antonio García Macías

Dr. Jesús Favela Vara

Dr. Salvador Villarreal Reyes

_____

Dr. Jesús Favela Vara
Coordinador del Posgrado en Ciencias de la Computación

_____

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Marco Antonio Calderón Pérez © 2016
Queda prohibida la reproducción parcial o total de esta obra sin el permiso formal y explícito del autor

Resumen de la tesis que presenta Marco Antonio Calderón Pérez como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Incorporación de capacidades semánticas en el Internet de las Cosas.

Resumen aprobado por:

Dr. José Antonio García Macías
Director de Tesis

Conforme avanza el desarrollo del Internet de las Cosas se ha puesto atención a la incorporación de inteligencia en los sistemas. Sin embargo, muchos de estos desarrollos aún continúan siendo muy centrados en los objetos, dejando de lado las necesidades humanas. Un aspecto importante para hacer al Internet de las Cosas mas centrado en los humanos es considerar aspectos que resultan importantes para estos. Proponemos al espacio y al tiempo como dos dimensiones que son de gran importancia en las actividades humanas. Dichas dimensiones son representadas mediante tecnologías semánticas e incorporadas en una plataforma de Internet de las Cosas. Se realizan pruebas de concepto para mostrar como nuestra propuesta permite la realización de sistemas con conciencia espacial y temporal que ayuden a las actividades humanas.

Palabras Clave: **internet de las cosas, semántica, proxémica, razonamiento temporal, razonamiento espacial.**

Abstract of the thesis presented by Marco Antonio Calderón Pérez as a partial requirement to obtain the Master of Science degree in Master in Sciences in Computer Science.

Incorporating semantic capabilities to the Internet of Things.

Abstract approved by:

Dr. José Antonio García Macías
Thesis Director

As the Internet of Things evolves, attention has been paid to adding intelligence capabilities. However, many of these developments are still too device-centric nd more needs to be done to make the IoT more people-centric. An important aspect in this direction is to incorporate aspects that are relevant to humans; we propose time and space as dimensions that are of great importance for human activities. These dimensions are represented by means of semantic technologies and incorporated into an Internet of Things platform. We make some proof of concepts to show how our proposal can aid in realizing systems with spatial and temporal awareness that aid in human activities.

Keywords: **internet of things, semantics, proxemics, temporal reasoning, spatial reasoning.**

Dedicatoria

A mis padres, Marco Antonio y Alma Delia.

Agradecimientos

Para comenzar me gustaría agradecer a mi asesor, el Dr. J. Antonio García Macías, por mostrarme paciencia, apoyo, guía y por brindarme las herramientas que necesitaba para poder culminar con esta tesis.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de
maestría.

A Dios, por guiarme y darme las energías, la inteligencia, y todo lo que necesité para culminar con esta tesis.

A mis padres, por darme a lo largo de mi vida las herramientas y el apoyo necesario para llegar hasta aquí y ver un sueño realizado. Por proveer el apoyo económico que necesité durante meses difíciles, y por su comprensión a lo largo de esta etapa.

A mis compañeros, por sus ánimos y el apoyo mútuo para alcanzar nuestra meta.

A Nancy Rojas, por brindarme los ánimos que necesitaba en una etapa difícil de mi vida, por su amor, comprensión y amistad, y por ayudarme a aceptar mis debilidades y crecer como persona. Por convencerme de seguir adelante en esta tesis y por ponerme a pensar sobre lo que quería hacer de esta etapa y en adelante.

A mis tíos, José Ángel y Alma, quienes me apoyaron para terminar mis estudios profesionales.

A Angello Hoyos, por ayudarme a desarrollar un componente importante para probar

mi trabajo.

A Saúl Delgadillo, por el tiempo que se dió para pasarme la batuta y continuar con este tema tan interesante.

Y a todas las demás personas que formaron parte de esta etapa y me aconsejaron y guiaron para escribir este documento.

Tabla de contenido

Página

Resumen en español ii

Resumen en inglés iii

Dedicatoria iv

Agradecimientos v

Lista de figuras x

Lista de tablas xii

Capítulo 1 Introducción 1
1.1 Planteamiento del problema . . . 2
1.2 Preguntas de investigación . . . 4
1.3 Objetivo general . . . 4
1.4 Objectivos específicos . . . 4
1.5 Contribución . . . 5
1.6 Metodología . . . 5
1.7 Organización de la tesis . . . 6

Capítulo 2 Marco Teórico 7
2.1 Internet de las Cosas . . . 7
2.1.1 Orientado a Cosas . . . 8
2.1.2 Orientado a Internet . . . 9
2.1.3 Orientado a Semántica . . . 10
2.1.4 Semántica en el IoT . . . 10
2.2 Web Semántica . . . 11
2.2.1 URI/URL . . . 12
2.2.2 XML . . . 13
2.2.3 RDF/RDFS . . . 13
2.2.4 Ontologías . . . 14
2.2.5 OWL . . . 14
2.2.6 SPARQL . . . 15
2.2.7 Reglas en la Web Semántica . . . 16
2.2.8 Razonadores Semánticos . . . 16
2.3 Proxémica . . . 16
2.3.1 Dimensiones proxémicas y el Cómputo Ubicuo . . . 17
2.4 Objetos Inteligentes . . . 19
2.4.1 Concientes de actividad . . . 20
2.4.2 Concientes de políticas . . . 20
2.4.3 Concientes de procesos . . . 21
2.5 Inteligencia Ambiental . . . 21
2.5.1 Conciente del contexto . . . 23
2.5.2 Razonamiento . . . 23
2.5.3 Problemas relacionados . . . 24

Capítulo 3 Importancia del trabajo 25
3.1 Diferencias respecto a otros trabajos . . . 25
3.2 Tecnologías Requeridas . . . 25
3.3 Caso de uso . . . 26
3.4 Conclusiones . . . 28

Capítulo 4 Componentes y descripción de la propuesta 29
4.1 Requerimientos . . . 29
4.1.1 Razonamiento Temporal y Espacial . . . 31
4.1.2 Comportamiento Proactivo . . . 32
4.1.3 Adherencia a Plataforma del Internet de las Cosas (OpenThings) 33
4.2 Razonamiento semántico . . . 33
4.2.1 Razonamiento Temporal . . . 34
4.2.2 Razonamiento Espacial . . . 34
4.3 Dimensiones proxémicas . . . 35
4.3.1 Identidad . . . 35
4.3.2 Distancia . . . 36
4.3.3 Ubicación . . . 36
4.3.4 Orientación . . . 36
4.4 Cambios de distancias . . . 37
4.5 Cercanía entre entidades . . . 38
4.6 Partes interesadas en el desarrollo del trabajo . . . 38
4.7 Preferencia y privacidad de los usuarios . . . 39
4.7.1 Preferencias de Usuarios . . . 41
4.8 Reconocimiento de Entidades . . . 42
4.9 Ontología . . . 43
4.9.1 Privacidad . . . 43
4.9.2 Lugares . . . 44
4.9.3 Proxémica . . . 46
4.10 Reglas . . . 47
4.10.1 Razonador Semántico . . . 48
4.10.2 OWL/RDFS . . . 48
4.11 Conclusión . . . 48

Capítulo 5 Arquitectura 50
5.1 Conexión con ambiente virtual . . . 51
5.2 Registro de Ambiente Virtual . . . 54
5.3 Smart Objects . . . 56

Capítulo 6 Aplicación contextual y sus componentes 61
6.1 El componente TripleStoreServer . . . 62
6.2 El componente “Device” . . . 63
6.2.1 Servicio de Ubicación usando Bluetooth Low Energy . . . 65
6.3 Smart Objects . . . 67
6.4 Beacons . . . 69
6.5 Ambiente Virtual . . . 70
6.5.1 Interfaz de red . . . 71
6.5.2 Registro de eventos . . . 72

6.5.3 Identificación . . . 72
6.5.4 Aplicación Contextual . . . 73
6.5.5 Implementación de interfaces para la captura eventos . . . 73
6.5.6 Manejo de ontologías . . . 74
6.6 Mensajes . . . 75
6.6.1 Ubicación . . . 75
6.6.2 Distancias proxémicas . . . 76
6.6.3 Orientación . . . 77

Capítulo 7 Implementación y Resultados 78
7.1 Demostración de uso de la Aplicación Contextual (ContextApplication) con OpenThings . . . 78
7.2 Análisis de la prueba de concepto . . . 79
7.3 Roles de la prueba de concepto . . . 82
7.4 Módulos empleados de la plataforma OpenThings . . . 82
7.5 Dispositivo proxémico . . . 84
7.6 Resultados obtenidos a partir de la prueba de concepto . . . 85
7.6.1 Tabla comparativa con trabajos relacionados . . . 86
7.6.2 Ontologías . . . 89
7.6.3 Manejador de Base de Conocimientos . . . 90
7.6.4 Implementación de un razonador semántico . . . 90
7.6.5 Fundamentos para crear nuevas aplicaciones . . . 90
7.6.6 Técnicas propuestas para implementar razonadores semánticos 91
7.7 Conclusiones . . . 92

Capítulo 8 Conclusiones y trabajo futuro 93
8.1 Conclusiones . . . 93
8.2 Comparativa entre sistemas . . . 94
8.3 Aportaciones . . . 94
8.4 Limitaciones . . . 95
8.5 Trabajo futuro . . . 96

Capítulo 9 Apéndice 98
9.1 API . . . 98
9.2 Ontología . . . 98
9.3 Reglas . . . 101

Lista de referencias bibliográficas 103

Lista de figuras

Figura		Página
1	Tres visiones del Internet de las Cosas.	8
2	La pirámide del conocimiento.	12
3	Ejemplo de una tripleta.	14
4	Un ejemplo de una ontología en Protégé 5.0 beta.	15
5	Ejemplo de una consulta SPARQL.	15
6	Las zonas proxémicas de acuerdo con Edward Hall.	17
7	Cinco dimensiones proxémicas según trabajos de Saul Greenberg.	18
8	Las categorías de los smart objects, de acuerdo a sus características y el rol dentro del ambiente.	20
9	Ambient Intelligence y su relación con las tecnologías que contribuyen a la visión.	22
10	Las dimensiones proxémicas en el cómputo ubicuo	35
11	Individuos de la ontología que muestra una ubicación y una distancia proxémica de “mc-galaxy-alpha” respecto a otro objeto, el cual es el Beacon “UBC5739”.	37
12	Diagrama de secuencia para la conexión con un ambiente virtual.	52
13	Ejemplo de un objeto PushInfo que indica el tipo de tema al que un smart object se registrará en HubServer.	54
14	Ejemplo de código en el que se envía una subscripción en cuanto se ha encontrado HubServer en la red.	54
15	Registro de un Ambiente Virtual con HubServer.	55
16	Ejemplo de uso de una clase en la plataforma OpenThings.	56
17	Diagrama de secuencia para enviar un Topic a HubServer, y su correspondiente manejo y notificación a Smart Objects.	57
18	Diagrama de secuencia cuando el Ambiente Virtual recibe un evento y es procesado por la aplicación contextual.	58
19	Diagrama de secuencia que ejemplifica el flujo para la actualización de una ubicación.	59
20	Diagrama de secuencia para la suscripción de un Smart Object dentro de la plataforma OpenThings.	60
21	Ejemplo de un mensaje para controlar la iluminación.	60
22	Ejemplo del resultado de realizar una petición GET al servicio TripleStore-Server.	62

Figura Página

23 Dos individuos en la ontología. Uno representa el Beacon con sus propiedades (como identificador único B108) y el otro el lugar (UbiquitousComputingLaboratory). 63

24 Diagrama UML de componentes del servicio TripleStoreServer y la aplicación para Android SmartNavi. 64

25 Pantalla tomada de la aplicación SmartNavi. 65

26 Ejemplo de uso de la clase MDNSHelper para buscar el servicio TripleStoreServer por mDNS. 65

27 Método para calcular la distancia entre dos objetos utilizando unos factores preestablecidos, propios de la antena Bluetooth con el que el device cueta. 66

28 Ejemplo del control de pines digitales en Java con la librería Mraa4j para Intel Edison. 67

29 Placa de desarrollo Intel Edison (2014). 68

30 Fotografía de un Beacon utilizando BLE Shield y la plataforma Arduino. Recuperado el 2 de noviembre del 2015 de http://redbearlab.com/getting-started-bleshield/. 69

31 Ejemplo de un mensaje en formato JSON para actualizar la ubicación. . . . 75

32 Ejemplo de un mensaje para un cambio de distancia proxémica. 76

33 Ejemplo de un mensaje para un cambio de orientación. 77

34 Dispositivo interactuando encendiendo o apagando la pantalla ante la posición del usuario. 79

35 Interacción de aplicación con Base de Conocimientos. 81

36 Dispositivo proxémico con acelerómetro, magnetómetro, comunicación por BLE y usando la plataforma FLORA (https://www.adafruit.com/products/659). 84

9.1 API del ambiente virtual para actualizar las dimensiones proxémicas. . . . 98

9.2 API de TripleStoreServer. 98

9.3 Ontología de lugares (Places). 99

9.4 Ontología de preferencias y privacidad de usuarios. 100

9.5 Ontología de las distancias proxémicas (Proxemics). 101

Lista de tablas

Tabla Página

1 8 reglas para las zonas proxémicas con sus fases. 48

2 Tabla que muestra las características para los trabajos relacionados revisados. 88

Capítulo 1. Introducción

A pesar de que el Internet no es la plataforma de comunicación adecuada para hacer realidad la visión de Mark Weiser, el cómputo ubicuo, su infraestructura es ampliamente utilizada en la actualidad y se torna pragmático utilizarla en vez de reinventar otra plataforma nueva (Avilés-López, 2012). Aunado a este problema, surge el paradigma del Internet de las Cosas a partir de la intersección de las visiones orientadas a Internet, a Cosas, y a Semántica (Atzori *et al.*, 2010).

Dos de las visiones del Internet de las Cosas, tomadas por diferentes iniciativas para definir este concepto y explorarlo tales como (Autor desconocido, 2015) (Singh *et al.*, 2014), requieren de una tercera visión, la Semántica, debido a la capacidad de dispositivos que se tendrán en el futuro y a la considerable cantidad de información que generarán. En la actualidad existen aproximadamente 9 billones de dispositivos conectados a Internet y se estima que en el 2020 la cifra aumente a 24 billones (Gubbi *et al.*, 2013). Es decir que en los próximos años, la cantidad de información que éstos van a compartir será grande a comparación con la que se maneja en la red de Internet actualmente. No es una tarea práctica almacenar su información sin obtener conocimiento a partir de ella. Esta problemática aumenta la necesidad de una visión semántica en el manejo de la información generada.

La visión Semántica es la clave para resolver el problema de la gran cantidad de información generada, y cuya finalidad se divide en: elaborar soluciones apropiadas para describir objetos inteligentes según sus atributos, inferir información a partir de los datos generados por el Internet de las Cosas, construir ambientes de ejecución semánticos, desarrollar y acoplar arquitecturas que se adaptan a los requerimientos del Internet de las Cosas, un almacenamiento escalable y una infraestructura para su comunicación (Atzori *et al.*, 2010) (Singh *et al.*, 2014).

Sin embargo, el trabajo realizado centrado en los objetos ha dejado de un lado el enfoque principal, que es en el usuario, pues la finalidad de estos objetos es ayudar a las personas a mejorar su calidad de vida y ayudarlas en actividades de la vida cotidiana (Sadri, 2011).

Este trabajo se enfoca en facilitar la creación de aplicaciones para ambientes inteligentes, con conciencia temporal y espacial, que busquen integrar ambas dimensiones dentro de estos y apoyar las actividades cotidianas de las personas que los habitan, como aplicaciones para AmI, AAL, y HCI. Así mismo, permitirá tener un enfoque centrado en el usuario, es decir, apoyará a los usuarios en sus actividades cotidianas (Sadri, 2011). De acuerdo a parámetros establecidos, los desarrolladores de aplicaciones para ambientes inteligentes pueden involucrarlos en ellas para que sea dotado de proactividad. Esto es usando la información generada por los dispositivos, usando una combinación de las dimesiones espaciales y temporales, y además del razonamiento que ofrecen las tecnologías semánticas.

La motivación principal es la implementación de las dimensiones espaciales y temporales usando tecnologías semánticas, siendo ambos factores importantes para conocer la actividad de una persona (Sashima *et al.*, 2006). Se cree que se puede facilitar la creación de aplicaciones enriquecidas en contexto para futuros ambientes del Internet de las Cosas y aplicaciones de HCI y AmI que requieran del razonamiento espacial y temporal del ambiente que les rodea.

1.1 Planteamiento del problema

Actualmente el Internet de las Cosas cuenta con algunos trabajos en la dirección de la visión semántica, como (Kim *et al.*, 2013) y (Jara *et al.*, 2014). En la diversidad de arquitecturas y plataformas que procuran integrar este paradigma, los esfuerzos se enfocan en este camino, sin embargo, pocos de ellos han procurado centrarse en el usuario, siendo que se han trabajado más por centrarse en los objetos o en las cosas. Las tecnologías semánticas deben procurar considerar también al usuario como el centro, y por lo tanto, unir sus esfuerzos para mejorar la calidad de sus actividades para ayudarles, y no centrar en la capacidad de actuación y sensado que tienen los objetos.

El razonamiento semántico en los trabajos relacionados generan conciencia contextual, en particular sobre la dimensión espacio-temporal. A menudo utilizan bases de datos relacionales con bases de datos espaciales (Soheili *et al.*, 2005) (Sashima *et al.*, 2006), pero el manejo de ontologías para representar esta información, de forma que se

obtenga un significado explícito, ha sido poco explorado. El término proxémica, acuñado por el psicólogo Hall (1990), describe una forma de comunicación no verbal entre las personas y su ambiente social y físico 2 de acuerdo a variantes en la distancia, mediando su interacción y la relación que existe entre ellas. Se encontraron cuatro zonas, que varían de una cultura a otra, las cuales muestran la relación que los humanos tienen con su ambiente, con sus objetos y otros individuos que les rodean. Integrar la teoría proxémica puede permitir el conocimiento aproximado que tienen los objetos y las entidades dentro de un sistema (Greenberg y Kuzuoka, 1999) (Diaz-Marino y Greenberg, 2010) (Marquardt *et al.*, 2011), y asimismo realizar inferencias con base en las distancias que les separan.

Para llevar a cabo este trabajo se propone desarrollar e implementar una entidad de software que cuenta con la capacidad de razonar en hechos del espacio y del tiempo usando la plataforma OpenThings, desarrollada para el Internet de las Cosas (Delgadillo-Rodríguez, 2015). Fue elegida por las características que ofrece, tales como:

1. Descubrimiento de servicios de forma automática (multicast DNS).
2. Servicios Virtuales (que pueden verse beneficiados con capacidades semánticas y de razonamiento espacial).
3. Acceso al código fuente de la plataforma.

El uso de la plataforma facilitará el desarrollo de aplicaciones conscientes del espacio y del tiempo, para proveer un comportamiento proactivo a las aplicaciones de acuerdo a preferencias de los usuarios que les utilicen.

Para integrar las tecnologías semánticas necesarias, se decidió utilizar el marco de trabajo Jena, pues facilita las herramientas semánticas necesarias (The Apache Foundation, 2014) a fin de lograr los siguientes objetivos:

1. Modelar los elementos importantes de un ambiente, tales como los lugares, las distancias, el tiempo, las preferencias de usuarios, etc.

2. Usar su razonador semántico basado en reglas que permita realizar inferencia en base a ciertos hechos del sistema (persona está en cierto lugar, iluminación encendida, cortinas cerradas, smart glass opacado).

1.2 Preguntas de investigación

Las preguntas de investigación surgen de la necesidad de comprender cuál es el problema y cómo se pretende atacar.

- ¿Cuales son los requerimientos para que una plataforma ofrezca un razonamiento espacial y temporal con tecnologías semánticas?
- ¿Qué requerimientos debe tener un sistema para que sea consciente del espacio y el tiempo?
- ¿Qué obstáculos existen en la plataforma al usar un razonamiento temporal y espacial y, en caso de que hubieran, cómo se pueden solucionar?

1.3 Objetivo general

Incorporación de capacidades semánticas en el Internet de las Cosas (Internet of Things, o por sus siglas en inglés IoT), para brindar conocimiento espacial y temporal a sistemas inteligentes.

1.4 Objectivos específicos

Para alcanzar el objetivo general propuesto, se identifican los siguientes objetivos específicos que ayudarán a alcanzar y a medir el progreso de este trabajo.

- Identificar las capacidades y limitaciones del IoT para ser consciente del espacio y del tiempo con semántica.
- Identificar, seleccionar, e implementar tecnologías semánticas en una plataforma del IoT.

- Investigar razonadores semánticos disponibles y proponer técnicas para introducirlos en una plataforma.
- Desarrollar una prueba de concepto e identificar obstáculos.

1.5 Contribución

El desarrollo de esta tecnología brindará conocimiento del tiempo y del espacio a la plataforma del Internet de las Cosas OpenThings (Delgadillo-Rodríguez, 2015) que facilita la comunicación semántica entre los dispositivos y entidades que le componen. El trabajo propuesto añadirá las siguientes características a la plataforma:

- Facilitar la creación de aplicaciones conscientes del espacio y del tiempo usando tecnologías semánticas.
- Integración de razonadores semánticos en un ambiente compuesto por smart objects.
- Cambios en la programación no dependen del firmware de los smart objects, por lo que no necesitan cambiar su funcionalidad para adaptarse a los cambios requeridos, sino que las modificaciones al comportamiento del ambiente se realizan directamente sobre la ontología.
- Implementar el término proxémica para representar el espacio y su significado en un ambiente del Internet de las Cosas.

1.6 Metodología

La metodología propuesta es la siguiente:

1. Revisión de trabajos relacionados con el uso del espacio y del tiempo en ambientes del cómputo ubicuo.
2. Búsqueda y desarrollo de ontologías para modelar tiempo y espacio con base en literatura.

3. Incorporar un servicio virtual dentro de la arquitectura OpenThings (Delgadillo-Rodríguez, 2015).

4. Realizar una prueba de concepto para validar y comprobar el funcionamiento en una aplicación real.

1.7 Organización de la tesis

Esta tesis está organizada de la siguiente manera: primero se procede a describir los conceptos principales que se ven en el contenido de la tesis en el capítulo Marco Teórico. Enseguida se indica en el capítulo sobre la Importancia del trabajo y una breve descripción de un caso de uso que ejemplifique el cómo se pretende alcanzar estos objetivos de una forma más aterrizada. Posteriormente se hablará de la Componentes y descripción de la propuesta y de cómo los conceptos abarcados en el Marco Teórico toman un rol dentro del trabajo. En el capítulo de Arquitectura y en Aplicación contextual y sus componentes se describe el funcionamiento de la propuesta. Posteriormente, se evalúa el trabajo en una aplicación real implementada en el capítulo Implementación y Resultados, junto con una descripción de cómo fue alcanzada y los pormenores encontrados en el transcurso del trabajo, y por último se concentra todo lo visto en la tesis en el capítulo de Conclusiones y trabajo futuro.

Capítulo 2. Marco Teórico

Antes de introducirnos en los detalles técnicos del trabajo, es importante definir una serie de conceptos para que al lector le facilite comprender los temas que se abarcarán, y entienda de manera más sencilla cuál es la contribución del trabajo y la visión de tener un ambiente inteligente del Internet de las Cosas que sea centrado en el usuario, considerando importantes las dimensiones temporales y espaciales de las mismas.

Este capítulo se divide en las siguientes secciones:

- Internet de las Cosas.
- Web Semántica.
- Proxémica.
- Objetos Inteligentes.
- Inteligencia Ambiental.

2.1 Internet de las Cosas

El Internet de las Cosas, Internet of Things (por sus siglas IoT), es un concepto que tiene sus raíces en el cómputo ubicuo, desarrollado en los laboratorios de Xerox en Palo Alto Research Center (PARC) por Mark Weiser. Se explicó en su tiempo que el cómputo iba a dejar de lado el escritorio y se iba a encontrar en las cosas de la vida cotidiana (Weiser, 1993). Esto implica que los elementos que le conforman, las “cosas” del IoT, tendrían capacidades de procesamiento y comunicación inalámbrica que permitirían llevar a cabo ambientes inteligentes, dotándole de inteligencia. El fundamento de la visión del Internet de las Cosas es dotar a los objetos de capacidades de procesamiento y de comunicación usando los estándares actuales que componen Internet.

El Internet de las Cosas está compuesto por dispositivos con capacidades de procesamiento, almacenamiento y comunicación limitados; manejan esquemas de direccionamiento e identificación únicos, y son capaces de comunicarse entre ellos de forma semántica a

través de los estándares definidos de Internet para conseguir una meta en común (Atzori *et al.*, 2010).

De acuerdo a las funciones que tienen, dotadas por los actuadores y sensores que poseen, cada dispositivo dentro del Internet de las Cosas toma un rol particular en la ejecución de alguna tarea específica. Ya que tienen una forma de percibir e interactuar en el dominio físico a través de dispositivos electrónicos, son capaces de medir fenómenos físicos y actuar sobre este dominio de acuerdo a su programación (A. Dunkels, 2008).

En la encuesta realizada por Atzori *et al.* (2010), el Internet de las Cosas está compuesto esencialmente por tres paradigmas: orientado a Internet, orientado a Cosas, y orientado a la semántica. A continuación se desarrollarán estos paradigmas.

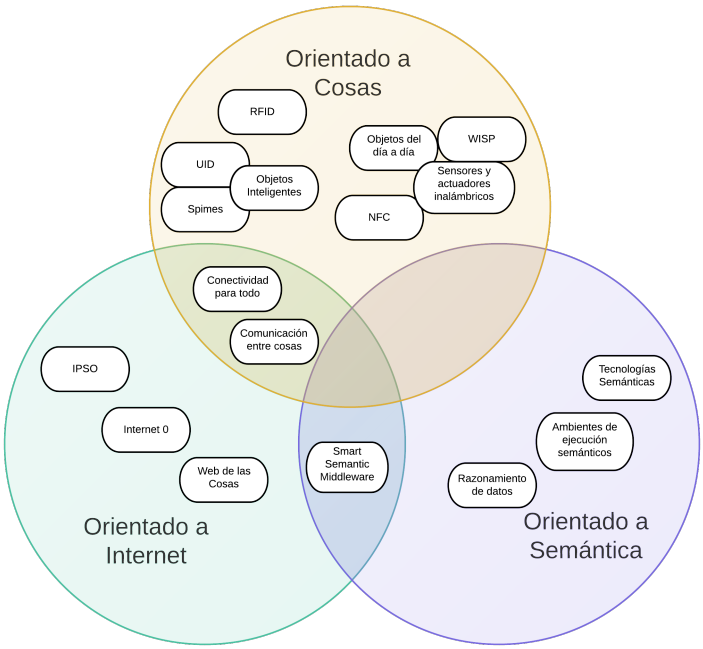

Figura 1**: Tres visiones del Internet de las Cosas.**

2.1.1 Orientado a Cosas

Cuando se colocan las palabras “Internet” y “cosas” asumen un significado que introduce un nivel disruptivo de innovación, que consiste en una red global de objetos interconectados, distinguibles de forma única, basados en protocolos de comunicación estándares. Esto implica un gran número de objetos heterogéneos involucrados en el

proceso.

Las “cosas” de las que se habla en el Internet de las Cosas anteriormente utilizaban la tecnología de Identificación por Radio Frecuencia (*Radio-Frequency IDentification*, o RFID), propuestas por Auto-ID Labs y EPCglobal (Atzori *et al*., 2010). Sin embargo, para un sentido más amplio del concepto del Internet de las Cosas, esta tecnología puesta en los objetos no es suficiente. En el artículo de M. Presser (2009) se menciona que la tecnología RFID, debido a su madurez y bajo coste, permite emplearse en las redes del Internet de las Cosas, pero a su vez mencionan un amplio portafolio de tecnologías que ampliarán la visión, como etiquetas de Near Field Communication (NFC) y Wireless Sensor and Actuactor Networks (WSAN), los cuales son llamados “los componentes atómicos que establecerán la brecha entre el mundo real con el mundo digital” (M. Presser, 2009).

Algunos organismos de tecnología enfocados en el Internet de las Cosas, han llevado su definición a un enfoque más inteligente (Atzori *et al.*, 2010). Han surgido nuevos conceptos como el *SPIME*, definido como un objeto que puede ser rastreado a través del tiempo y del espacio en su tiempo de vida y que puede ser sostenible, mejorable, e identificado de forma única (Sterling, 2005). A partir de aquí surge también el concepto de Smart Object, un dispositivo que además de contar con capacidades usuales de comunicación inalámbrica, memoria y actuación, tienen la capacidad de presentar comportamiento proactivo, conciencia contextual y comunicación colaborativa (A. Dunkels, 2008).

2.1.2 Orientado a Internet

En la visión del Internet se busca la interacción de los objetos desde una persectiva de comunicación y protocolos que son parte de la suite de Internet. Términos como la Web Semántica surgen a partir de ello, en el que la comunicación entre entidades tiene una perspectiva orientada a servicios, un uso extensivo del protocolo HTTP y del paradigma de comunicación REST (Tim Berners-Lee y Lassila, 2001).

El propósito de esta visión requiere un conjunto de objetos inteligentes conectados por procolos IP o similares, de acuerdo a la capa de transporte con el que cuentan. Se suman trabajos de la IPSO Alliance, como la integración de IP en dichos dispositivos a

través de 6LowPAN, o CoAP (A. Dunkels, 2008). La idea de que cada dispositivo esté conectado a Internet es que ofrezca un servicio a otros consumidores, como pueden ser tanto usuarios como otros servicios.

2.1.3 Orientado a Semántica

Debido a la enorme cantidad de dispositivos que se preve que se encuentren operativos en el año 2020 (Barnaghi *et al.*, 2012), los datos que provienen de sus sensores serán masivos (Singh *et al.*, 2014). Por lo tanto el procesamiento de los datos se torna crucial para evitar el congestionamiento de las redes inalámbricas y la sobrecarga de información para los dispositivos con baja capacidad de almacenamiento. Existe una necesidad inherente de herramientas que procesen los datos, con el fin de obtener un significado acerca del fenómeno que se está midiendo (Huang y Li, 2010).

Uno de los requerimientos más importantes para el Internet de las Cosas es la interoperabilidad entre las “cosas” que le componen, así como soportar el direccionamiento único entre objetos, seguimiento, y descubrimiento, representación, almacenamiento e intercambio de información. Algunas tecnologías que le componen son las ontologías representadas en formatos como RDF/RDFS, OWL, razonadores semánticos, y Linked Data (Berners-Lee, 2006). Mas adelante se describirán estas tecnologías.

Al aplicar tecnologías semánticas en el Internet de las Cosas promoverá interoperabilidad entre sus recursos, modelos de información, productores y consumidores de datos, y por tanto facilitar el acceso a los datos, su integración, descubrimiento de recursos, razonamiento semántico, y extracción de conocimiento, de forma inequívoca (Barnaghi *et al.*, 2012).

2.1.4 Semántica en el IoT

La visión semántica dentro del Internet de las Cosas busca que los objetos del Internet de las Cosas sean interoperables semánticamente (Atzori *et al.*, 2010). Tiene como finalidad describir los objetos inteligentes, sus procesos e interacciones con el ambiente, y otros conceptos que son de importancia en el contexto (Barnaghi *et al.*, 2012). Es importante notar que existen ciertas limitantes de utilizar tecnologías semánticas dentro del

Internet de las Cosas. Los puntos a tomar en cuenta en el manejo de descripciones semánticas son listadas en el trabajo de Tekli *et al.* (2013), y son:

- **Crear ontologías comprensivas.** Es complicado desarrollar una ontología que contenga todos los conceptos de un dominio. Una enorme cantidad de conceptos se relaciona con un procesamiento lento.
- **Balance entre expresividad y razonamiento.** Entre más completo y descriptivo es el lenguaje, más difícil se torna su razonamiento. Es por eso que debe elegirse con cuidado el DL a ser utilizado y el nivel de detalle semántico que se necesita.
- **Mapeo estructural y semánticamente rico de documentos.** Las ontologías deben evitar la redundancia y mejorar la eficiencia relacionando conceptos similares.
- **Simplificar el uso y manipulación de Bases de Conocimientos** y referencias semánticas para ayudar a crear, manipular, y consultar documentos semánticamente ricos, permitiendo de esta manera la interoperabilidad semántica.

2.2 Web Semántica

La Web Semántica es un movimiento colaborativo guiado por la World Wide Web Consortium (W3C), que busca extender la Web como la conocemos otorgándole a la información un significado bien definido para mejorar el acceso a los datos para usuarios y máquinas (Tim Berners-Lee y Lassila, 2001). En esta visión se busca mejorar la búsqueda de información entre máquinas, para introducir mecanismos para acceso a los datos que sea inteligente y efectivo. Esta misma corresponde a uno de los pilares del Internet de las Cosas (IoT), pues la cantidad de dispositivos crecerá de forma exponencial en los siguientes años (Atzori *et al.*, 2010) y se torna necesario un procesamiento de información más eficiente que consultas simples en los datos (Huang y Li, 2010).

Al hablar de la Web Semántica también se deben incluir cuatro conceptos clave que se relacionan entre sí.

La representación de información en la Web Semántica, la clave para su funcionamiento, requiere de los siguientes logros tecnológicos:

1. La representación de información dentro de Bases de Conocimiento (Knowledge Bases) en taxonomías, también conocidas como ontologías.

2. Tecnologías de representación de información basadas en XML o similares, como RDF, RDFS, OWL, y para consultas como SPARQL.

Estos dos puntos conforman el núcleo de la Web Semántica. Son tecnologías extensibles, interoperables e independientes de la plataforma, pensadas para mejorar el modelado de los datos, su connotación, manipulación, búsqueda e integración, permitiendo consultas inteligentes de información en la Web (Decker *et al*., 2000).

A continuación se describirán las tecnologías más importantes que conforman los pilares de la Web Semántica.

2.2.1 URI/URL

Los identificadores de artículos en Internet, conocidos como Identificadores de Recursos Universales (Universal Resource Identifiers, URIs), son la base fundamental de la

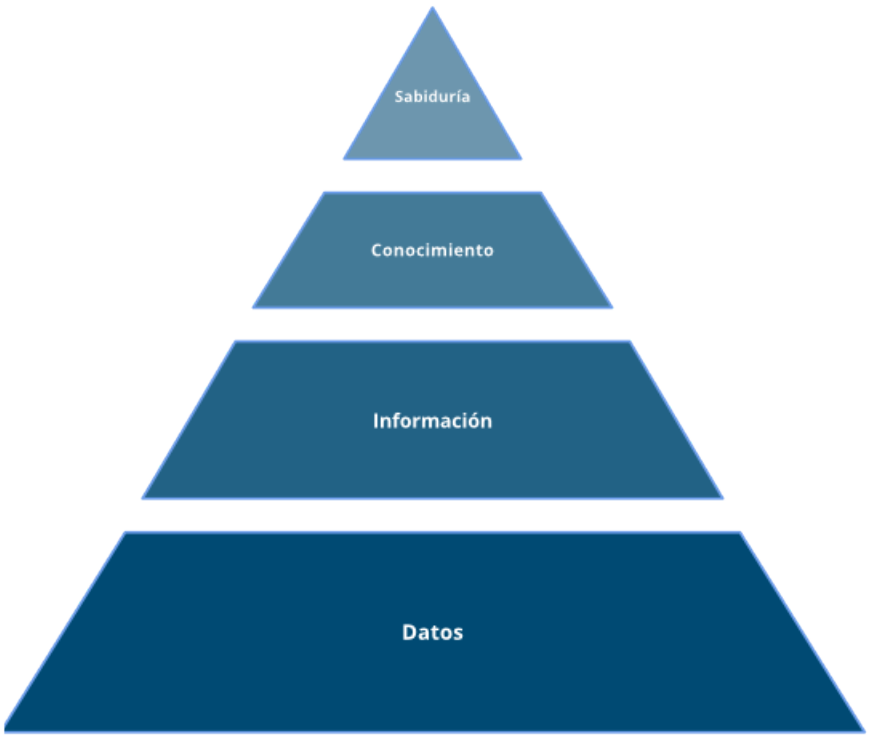

Figura 2**: La pirámide del conocimiento.**

Web Semántica, que tiene fundamento en el paradigma de Datos Ligados (Linked Data) propuesto por Berners-Lee (2006). En su estándar, definido en el RFC 3986 (Berners-Lee *et al.*, 1998), se establecen sus características para describir recursos en la Web de forma única. Una forma especial de un URI, el URL, permite identificar documentos en la Web empleando el mismo paradigma.

2.2.2 XML

La interoperabilidad semántica entre diferentes plataformas se torna un aspecto fundamental a considerarse, debido a la cantidad de información que se manejará, que proviene de los dispositivos que serán introducidos en los próximos años, como se mencionó anteriormente (Barnaghi *et al.*, 2012). Debe ser posible una interpretación única de los datos para darles un significado y producir resultados para un manejo eficiente de la información y aplicaciones de búsqueda.

Para este propósito se utiliza XML, un lenguaje de marcado que permite representar información de forma semántica en un documento semiestructurado: contiene sus jerarquías entre conceptos y es elaborada a partir de un conjunto de elementos atómicos y complejos, así como componentes atómicos, que incorporan estructura y contenido en una entidad. Simplifica la tarea de interoperabilidad e integración entre fuentes de información heterogéneas (Tekli *et al.*, 2013).

2.2.3 RDF/RDFS

Resource Description Framework es un modelo de datos diseñado para estandarizar el uso de metadatos de forma que se pudiese describir y organizar la información semántica. Utiliza XML para representar la información semántica (RDF Working Group, 2014). Este modelo está basado en estructuras llamadas tripletas (Objeto - Atributo - Valor). Una tripleta relaciona un valor a un objeto, dando una relación con significado semántico. Los Objetos, atributos, y valores pueden relacionar cualquier recurso en la Web (URIs), aunque pueden contener también valores literales, como nombres, valores, etc (Tekli *et al.*, 2013).

Es importante remarcar que RDF es sólo un modelo de datos. Lo que realmente

define el vocabulario semántico detrás de este modelo es el esquema RDF (RDFS), que permite modelar y manipular clases, de manera similar como lo hace un lenguaje de programación orientado a objetos. A diferencia de RDF, éste describe propiedades en términos de clases, también llamados recursos (Tekli *et al.*, 2013). En conclusión, RDFS es una extensión semántica de RDF que provee mecanismos para describir grupos de recursos relacionados y sus relaciones semánticas entre ellos.

2.2.4 Ontologías

Las ontologías son una formalización y definición de tipos, propiedades e interrelaciones entre entidades que existen en un dominio particular. Es una aplicación práctica de las ontologías filosóficas con una taxonomía. Proveen una forma efectiva de compartir y concordar en vocabularios comunes y modelos de conocimiento para describir los datos, lo que ayuda a las máquina a interpretar la información y presentarla en formas interoperables y reusables (Gruber, 1995). En la figura 4 se observa un ejemplo de una ontología simple.

2.2.5 OWL

A pesar del alto grado de expresividad que tiene RDF y RDFS, aún quedan huecos por llenar para representar información semántica, como construir fuentes de información semántica llamadas ontologías Web, que sirven como referencia para agentes de software cuando procesan los recursos Web de manera automática (similar a diccionarios o enciclopedias). La W3C introduce OWL (Ontology Web Language) como estándar para describir ontologías en la Web (OWL Working Group, 2012). Está construido a partir de RDF y RDFS, y hereda los elementos más básicos de RDFS, permitiendo así la especialización de propiedades, identificación o clases disjuntas, especificación de cardinalidad o restricciones de datos, etc.

Figura 3**: Ejemplo de una tripleta.**

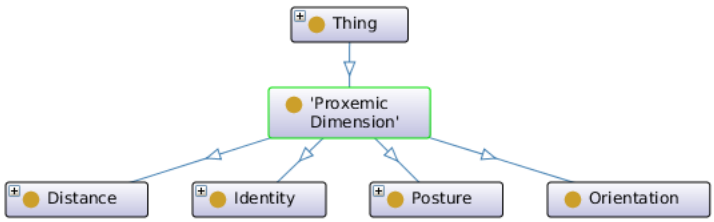

Figura 4**: Un ejemplo de una ontología en Protégé 5.0 beta.**

Debido a que el enriquecimiento semántico de OWL puede ser muy complejo para algunos escenarios, se han presentado tres variantes dependiendo del nivel de razonamiento: OWL Lite, OWL-DL y OWL full (Tekli *et al.*, 2013).

2.2.6 SPARQL

SPARQL es un lenguaje de consultas para bases de datos semánticas, mejor conocidas como *triplestore*, y es usado para consultar descripciones semánticas. Está diseñado para reemplazar SQL en su gramática para facilitar su uso a los programadores (Eric Prud'hommeaux, 2008). Es posible añadir otros dominios como en RDFS o en OWL, y puede ser extendido para consultar documentos con ontologías expresivas.

La forma en que se elabora una consulta en SPARQL es similar a SQL, con la diferencia que se busca información en forma de tripletas, como objetos con atributos y valores dentro de un rango. En la figura 5 se puede observar un ejemplo de una consulta simple en SPARQL.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX timeEvents: <http://www.intelligence.tuc.gr/2011/timeEvents#>
PREFIX time: <http://www.w3.org/2006/time#>
SELECT ?x ?start ?end
WHERE
{   ?x timeEvents:hasUID "Event1029"^^xsd:string ;
    time:hasDurationDescription ?y .
    ?y timeEvents:startTimestamp ?start .
    ?y timeEvents:endTimestamp ?end .
}
```

Figura 5**: Ejemplo de una consulta SPARQL.**

2.2.7 Reglas en la Web Semántica

Las reglas de la Web Semántica son una parte central de la representación del conocimiento en la Web Semántica. Proveen la expresividad requerida para habilitar la interpretación de agentes no humanos, procesamiento automatizado y traducción a otros lenguajes Web (Barnaghi *et al.*, 2012). Existen diferentes trabajos que son propuestos por la W3C como SWRL (Semantic Web Rule Language) (Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, 2004), RIF (Rule Interchange Format) (Kifer, 2008), RuleML (Rule Markup Language) (Boley *et al.*, 2001) y R2ML (Working Group I1, 2006). Estos tres lenguajes son representados en XML, empleando URIs e IRIs para constantes y sirven de interfaces con los lenguajes de ontologías (Paschke y Boley, 2010).

2.2.8 Razonadores Semánticos

Los razonadores semánticos, una generalización de las máquinas de inferencias utilizadas en inteligencia artificial, son herramientas capaces de realizar tareas de razonamiento, típicamente basados en RDFS, OWL, o alguna máquina basada en reglas (Autor desconocido, 2011), y utilizan la lógica de predicados para obtener conclusiones de los hechos. El razonamiento es un mecanismo poderoso que puede sacar conclusiones a partir de relaciones entre entidades de una ontología. Es también una de las herramientas más importantes de la Web Semántica ya que permite derivar hechos a partir de lo ya existente (Autor desconocido, 2012) y de axiomas definidos, ampliando de esta forma el conocimiento de un dominio o aplicación de una ontología.

La lógica de predicados (description logics, DL) es usada para el razonamiento formal de conceptos dentro de un dominio de aplicación (Baader *et al.*, 2003). De la misma forma que en la lógica de predicados, los razonadores poseen un componente donde se encuentra almacenada la información. Ésta es conocida como Base de Conocimientos.

2.3 Proxémica

En 1966, el antropólogo Edward Hall acuñó el término “proxémica” y la definió como el área de estudio que identifica las formas en la que las personas usan su distancia inter-

personal para entender y mediar sus interacciones con otras personas, que acuerdo a factores culturales (Hall, 1990). A pesar de que su teoría describe varios conceptos, quizá el más interesante es el relacionado con las interacciones dentro de la Interacción Humano-Computadora (Human-Computer Interaction, o HCI). En su libro *The Hidden Dimension* describió cuatro zonas "proxémicas" las cuales ayudan a comprender las relaciones entre las personas, cada una con su respectiva fase cercana y lejana: íntima (cercana: 0-15cm, lejana: 15-46cm), personal (cercana: 46-76cm, lejana: 76-122cm), social (cercana: 1.2-2.1m, lejana: 2.1-3.7m) y pública (cercana: 3.7-7.6m, lejana: ¿ 7.6m) (Hall, 1990). En la figura 6 se puede observar cómo es que están organizadas estas zonas respecto a un individuo.

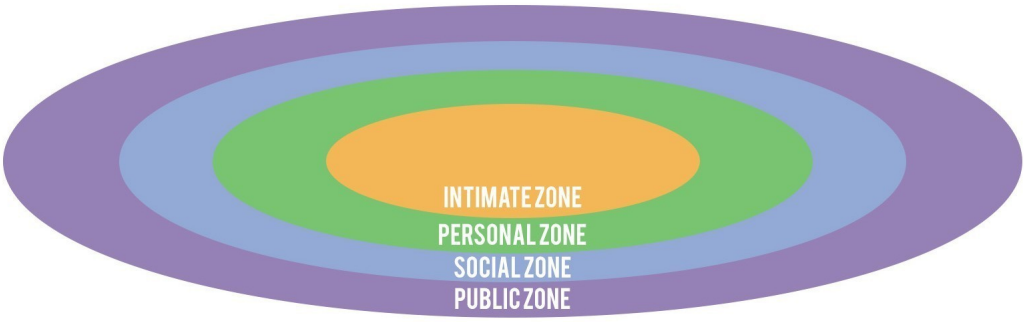

Figura 6**: Las zonas proxémicas de acuerdo con Edward Hall.**

Además de estas cuatro zonas, Hall (1990) propuso características del espacio que afectan las interacciones entre las personas (Hall, 1990). Las "características fijas" incluyen aquellos que marcan límites, como entradas a un cuarto, donde las personas tienden a concertar actividades sociales dentro de ellos. Las "características semifijas" abarcan entidades cuya posición puede afectar el espacio y hacer que las personas se acerquen o se alejen respecto a ellas mismas (e. g. el arreglo de sillas).

2.3.1 Dimensiones proxémicas y el Cómputo Ubicuo

Saul Greenberg ha tomado y operacionalizado estas cuatro zonas y las características del espacio para adoptarlas dentro del cómputo ubicuo. Es decir, su objetivo fue hacer de la proximidad entre individuos medible, interpretando las zonas no sólo por su distancia, sino elementos sociales y culturales también. La proxémica en el cómputo ubicuo involucra distancia entre individuos, donde éstos pueden ser personas, dispositivos dig-

itales, y cosas no digitales. La noción que expresan en su trabajo se explican en cinco dimensiones que pueden ser medidas y obtener valores de ellas discretos o continuos, dependiendo de la aplicación.

1. **Distancia entre entidades.** La distancia puede ser discreta, en vez de tener una medida continua. Así como Hall definió rangos para crear las zonas proxémicas, otras personas las han empleado con las implicaciones de lo que significan, considerándole así como una medida discreta para conocer la zona en la que un objeto se encuentra respecto a otra entidad.

2. **Orientación.** Captura los matices que no son provistos por la distancia. Puede ser continuo (ángulo de un objeto con respecto a otro) o discreto (de frente, ligeramente enfrente, o mirando en otra dirección respecto a otro objeto). Se debe considerar que la orientación implica que la entidad tiene una cara frontal.

3. **Movimiento.** Captura los cambios de distancia y orientación de una entidad en el tiempo. Es posible inferir información como la velocidad con la que se mueve la entidad e interpretarlo como un gesto.

4. **Identidad.** Describe únicamente a una entidad. Abarca una medida detallada como sus atributos, así como más general que simplemente permite distinguir una entidad de otra.

5. **Ubicación.** Describe el contexto físico en el que una entidad reside. La medida de ubicación puede capturar aspectos contextuales como cuando una entidad cruza un umbral, marcando su presencia dentro de una habitación.

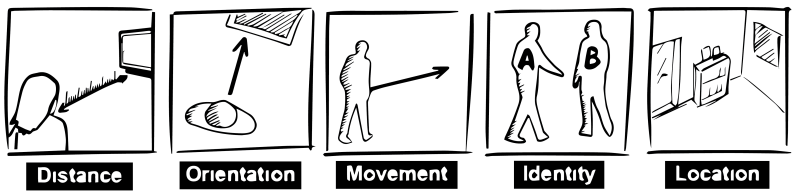

Figura 7**: Cinco dimensiones proxémicas según trabajos de Saul Greenberg.**

La idea propuesta por Greenberg *et al.* (2011) de establecer una nueva definición dentro del cómputo ubicuo surge del uso de estas cinco dimensiones dentro del diseño

de sistemas. Su propuesta de interacción proxémica es una estrategia para mediar la interacción entre personas en una ecología del tamaño de cuartos, permitiendo tener relaciones proxémicas con los dispositivos digitales cercanos y facilita una interacción más natural e ininterrumpida (Marquardt *et al.*, 2011).

Debido a este trabajo y la importancia dentro de las nuevas interacciones con ambientes inteligentes, para este trabajo se decidió tomar estas dimensiones e incluirlas dentro de una arquitectura. La finalidad es que los aspectos discretos y continuos (distancia y orientación) estén disponibles para aplicaciones que deseen controlar variables físicas del ambiente de acuerdo a las funciones de los objetos inteligentes y a la preferencia de los usuarios. Estas dimensiones se muestran en la figura 7.

2.4 Objetos Inteligentes

La composición del Internet de las Cosas es precisamente ofrecer una inteligencia en los objetos de la vida cotidiana. Estos objetos, conocidos como **smart objects**, son entidades físicas y digitales aumentadas que cuentan con sensores, procesamiento y capacidades de comunicación (A. Dunkels, 2008). Ellos son capaces de sensar, registrar e interpretar lo que está ocurriendo dentro de ellas y en el mundo que les rodea, actuar bajo su propia programación, intercomunicarse con otros, e intercambiar información con las personas (Kortuem *et al.*, 2010).

Son los que brindan las capacidades virtuales de la computación tradicional a un nivel físico, en el cuál interactúan, sensan, y son accionados de acuerdo a sus propias capacidades (A. Dunkels, 2008). Las personas se relacionan con su ambiente de distintas maneras físicas, y éstos objetos son encargados de interpretar la información que recaban de dicha interacción para colaborar entre ellos y actuar de forma acorde. Esto implica que no sólo son capaces de interpretar información de los sensores, sino que pueden comunicarse entre ellas para cooperar y alcanzar metas comunes (Kortuem *et al.*, 2010).

Es decir, que los objetos inteligentes que se componen por sensores y comunicación inalámbrica, no se encargan de procesar e intercambiar información solamente, sino que además ayudan a las personas en sus tareas cotidianas. Las capacidades de entrada y de salida de los objetos inteligentes son la clave para su éxito.

En el trabajo realizado por Kortuem *et al.* (2010) categorizan los objetos inteligentes en tres divisiones, las cuáles tienen tres dimensiones (conciencia, representación e interacción) que las hace diferentes entre ellas. En la figura 8 se ilustra un diagrama que indica estas tres categorías de acuerdo a su nivel de involucramiento dentro de las actividades de las personas. Más adelante se describen dichas categorías para describir las características de los smart objects que pertenecen a ellas.

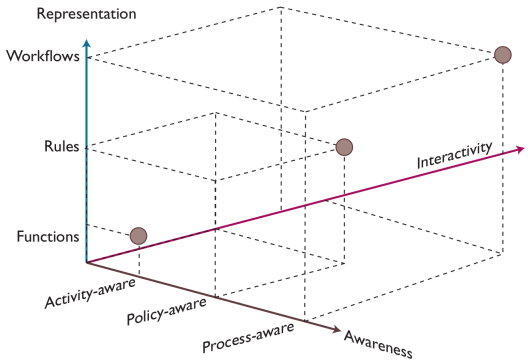

Figura 8**: Las categorías de los smart objects, de acuerdo a sus características y el rol dentro del ambiente.**

2.4.1 Concientes de actividad

Son los más simples de los tres tipos. Analiza el flujo de datos de sus sensores, utiliza algoritmos de reconocimiento para detectar actividades y eventos asociados directamente con el manejo del objeto, y aplica funciones de agregación específicas de aplicaciones. Utiliza funciones de agregación para acumular las actividades y eventos realizados por el tiempo. Ya que son los objetos más simples, sólo se encargan de manejar información y no provee capacidades de interacción.

2.4.2 Concientes de políticas

Un objeto que es conciente de políticas es aquel que puede interpretar eventos y actividades respecto a políticas organizacionales predefinidas. Utiliza una serie de reglas que operan sobre los eventos y actividades para crear acciones. Provee información sensible al contexto acerca del manejo del objeto y su desempeño en la actividad, lo cual permite crear consciencia contextual con base en la información proveniente de sensores para así describir una situación particular en el tiempo.

2.4.3 Concientes de procesos

Un proceso es una colección de actividades relacionadas o tareas que están ordenadas de acuerdo a su posición en el tiempo y el espacio. Un objeto conciente del proceso representa el más completo de los tres tipos de objetos inteligentes. Entiende procesos organizacionales y puede relacionar la ocurrencia entre actividades del mundo real y eventos a estos procesos. Su modelo de aplicación consiste en un flujo de trabajo basado en contexto que define el tiempo y el ordenamiento de las actividades a realizar. Provee a las personas que le rodean con guía basada en el contexto sobre las tareas, decisiones, etc., de la información que es recabada por los otros objetos que corresponden a las categorías anteriores.

2.5 Inteligencia Ambiental

La inteligencia ambiental (o AmI por sus siglas en inglés) es la visión del futuro en el cuál los ambientes ayudan a las personas que los habitan (De Ruyter y Aarts, 2004). Es una de las visiones que parten del cómputo ubicuo propuesto por Mark Weiser en Xerox PARC (Weiser, 1993). La integración de inteligencia en cada objeto del ambiente, ya sea nuestra ropa, las sillas, puertas, ventanas, y todo lo demás permite que se pueda promover una ayuda a las personas que interactúan con él, de una forma amigable, y con la capacidad de expresar, reconocer y responder a las emociones (Sadri, 2011).

Un ambiente que está caracterizado por tener inteligencia está basado en hardware miniaturizado y de bajo coste, que provee redes complejas de información heterogénea de electrodomésticos o artefactos inteligentes. Debe también ser ubicuo, es decir, estar presente en donde sea, y debe estar distribuido en un área o un grupo de personas (Sadri, 2011).

En el trabajo realizado por Cook *et al.* (2009), se establecen 5 tecnologías que contribuyen al desarrollo e implementación de la AmI, las cuales se muestran en la figura 9. Cada una de ellas ofrece lo necesario para poder brindar esta visión dentro de los ambientes que ayudarán a las personas a realizar sus tareas cotidianas. En general, para que un ambiente sea dotado de inteligencia debe ser sensible a estímulos de sensores, responsivo y adaptativo.

La inteligencia ambiental está basada en estímulos creados por una persona, y una respuesta al estímulo. Ya que está basada para ambientes de la vida real, el uso de sensores es crucial, ya que no tendría ningún sentido utilizar algo virtual para promover la ayuda a las personas de un ambiente si toda la información y los eventos se generaran y tuvieran repercusión a nivel computacional (Sadri, 2011). Para que un ambiente pueda ser considerado como inteligente debe tener ciertas características. En la literatura podemos encontrar varias definiciones y conceptos clave que nos permiten identificar a este tipo de ambientes, sin embargo, la mayoría concuerdan en las mismas (Sadri, 2011) (Cook *et al.*, 2009) (Ducatel *et al.*, 2003). Los siguientes puntos se encuentran en la encuesta realizada por Sadri (2011). Más adelante se abarcarán algunas características de importancia para detallar este concepto.

- Embebido.
- Consciente del contexto.
- Personalizado.
- Adaptativo.
- Anticipatorio.

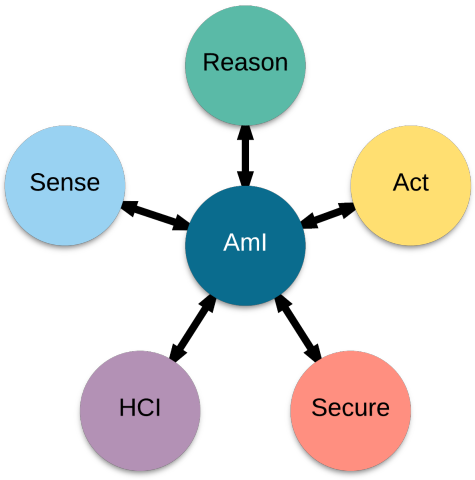

Figura 9**: Ambient Intelligence y su relación con las tecnologías que contribuyen a la visión.**

2.5.1 Conciente del contexto

La clave de los ambientes inteligentes es que no requieran de un agente humano para que el sistema se adecúe a las necesidades de las personas que se encuentran dentro de ellos. Para que pueda actuar acorde a la situación en la que se encuentra un usuario o un grupo de usuarios en un ambiente inteligente, el sistema debe conocer los factores de interés para ofrecer recomendaciones o actuar de forma que apoye sus actividades. La conciencia contextual, también conocida como cómputo sensible, es la clave para construir aplicaciones de inteligencia ambiental (Cook *et al.*, 2009). A partir de lo que una persona hace, como una tarea cotidiana ya sea moverse de un lugar a otro dentro de un edificio, el ambiente debe inferir hechos a partir de la situación para reaccionar acorde a los estímulos causados por lo general por un agente humano.

Es una de las características inherentes de la inteligencia ambiental. El contexto define el comportamiento que deben tener de acuerdo a las preferencias de los usuarios. Es importante considerar que para que un sistema sea consciente del contexto requiere de múltiples sensores, algunos de los cuales pueden combinarse para crear información enriquecida.

2.5.2 Razonamiento

El simple hecho de que un ambiente sea “sensible” a los estímulos del ambiente no es suficiente para que un ambiente sea considerado inteligente. Debe haber detrás mecanismos que provean un razonamiento basados en los hechos del ambiente. Se torna necesario tomar un curso de acción cuando existen cambios en los ambientes, en los planos temporales y espaciales.

El razonamiento provee al sistema de un comportamiento adaptativo y anticipatorio a las actividades de los usuarios, y además es posible hacerlo personalizado si se conoce previamente la identidad de las personas que lo usan. Permite reconocer el tipo de actividad que está haciendo un usuario con base a la información del ambiente. El razonamiento es tan importante como la conciencia contextual en el Ambient Intelligence. Crear contexto a partir de lo que existe en el ambiente, y con información inferida a partir de los hechos, permite a las aplicaciones ofrecer un servicio a un nivel más elevado pues

conoce los factores que intervienen en dicho lugar y puede enviar órdenes necesarias para adecuarse a las necesidades de una persona o grupos de personas en particular. Sin embargo, el manejo de esta clase de datos suele estar acompañado de problemas que le convierten en un tema debatido y en los ambientes que se están desarrollando en esta área (De Ruyter y Aarts, 2004).

2.5.3 Problemas relacionados

La **aceptación social de la inteligencia ambiental** tiende a ser negativa cuando se explica al usuario la información que es interpretada por el sistema (Cook *et al.*, 2009). Para los usuarios, la privacidad es un tema delicado que debe tratarse. En general éste es un problema abierto en el área del cómputo ubicuo pues se maneja información sensible de los usuarios para proveer una mejor calidad de servicio. Asimismo, hay riesgos de seguridad en la información pues los ambientes captan información que puede ayudar a terceros a conocer los patrones de comportamiento que tienen, la hora del día en que se encuentran en sus casas, qué es lo que hacen en su tiempo libre, etc.

Otro problema relacionado está asociado con el manejo de tecnología, ya que pueden ocasionar problemas de seguridad, desde la precisión de la información que los sensores proveen hasta problemas dentro de la instalación. Así como en la privacidad de la información, se debe considerar mecanismos y protocolos de seguridad usando el mínimo de recursos (Cook *et al.*, 2009).

Capítulo 3. Importancia del trabajo

Dentro de este capítulo se abarcará y resaltará la relevancia que tiene este trabajo, las posibilidades que ofrecerá, así como las áreas en las que los interesados pueden beneficiarse ampliamente, especialmente en la conciencia, representación, y razonamiento de las dimensiones espaciales y temporales que un humano percibe y el cual es importante considerar.

3.1 Diferencias respecto a otros trabajos

La diferencia respecto a otros sistemas es el manejo de tecnologías semánticas y el uso del paradigma del Internet de las Cosas como plataforma. Aunque se han estudiado las dimensiones proxémicas y temporales en diversos trabajos de la literatura, no existe una implementación actual con el Internet de las Cosas usando las tecnologías semánticas, siendo éste el principal objetivo del presente trabajo de tesis.

Se han elaborado trabajos que consideran el plano espacio temporal para establecer relaciones entre las actividades que realiza un ser humano de acuerdo en su tiempo y en la ubicación que se encuentran (Augusto y Nugent, 2004) (Augusto *et al.*, 2008). Sin embargo, éstas se efocan en el uso de tecnologías semánticas sin una ontología desarrollada con los conceptos de la proxémica. Con el presente trabajo se pretende integrar las zonas discretas de las distancias proxémicas y las dimensiones proxémicas definidas por Greenberg y Kuzuoka (1999), así como la representación del plano temporal a través de herramientas de la Web Semántica para la obtención de un contexto más preciso y relacionado con la actividad de las personas dentro de un ambiente.

3.2 Tecnologías Requeridas

Para que la pieza de software a desarrollar cuente con tecnologías semánticas, se decidió utilizar las herramientas que ofrece el marco de trabajo Jena (The Apache Foundation, 2014) para:

1. Modelar los elementos importantes de un ambiente, tales como los lugares, las distancias, el tiempo, las preferencias de usuarios, etc.

2. Usar su razonador semántico basado en reglas que permita realizar inferencia en base a ciertos hechos del sistema.

3. Emplear su API para la actualización de la ontología, y para enviar notificaciones de cambios importantes en las propiedades de los individuos que contiene.

3.3 Caso de uso

Para ejemplificar el uso de esta tecnología implementada dentro de una plataforma se presenta un ejemplo:

Una familia vive dentro de una casa que contiene distintos smart objects, que controlan el clima (aires acondicionados, abanicos, ventanas), la iluminación (cortinas, lámparas, focos), y entretenimiento (televisión inteligente, sistema de sonido). Para realizar actividades cotidianas, prefieren la iluminación natural sobre la iluminación artificial. El padre de familia, al llegar del trabajo, se toma una hora de su tiempo para disfrutar de la lectura cerca de la ventana que da al patio. El sistema que coordina los dispositivos determina la distancia de la persona con respecto a los objetos cercanos (como una lámpara que está en la mesita de noche, al lado de la ventana, una ventana y las cortinas). Al ser las cuatro de la tarde, hora en que el padre realiza una actividad, el sistema envía las órdenes necesarias a los objetos para adaptarse a las preferencias predefinidas por la persona (abrir las cortinas, apagar la iluminación, y abrir ligeramente la ventana para permitir la entrada de aire).

A continuación se explican los pasos que se llevan a cabo para que el ambiente se adapte a las preferencias.

- Cambios de distancia para diversos smart objects (SmartGlass, Cortinas, Lámpara).
- Actualización y categorización de distancias y dimensiones proxémicas (zonas proxémicas íntimas/personales).
- Razonamiento sobre la identidad de la persona y el cambio de dimensiones proxémicas.
- Adaptación del ambiente para adecuarse a las preferencias obtenidas.

En este ejemplo sería muy práctico desarrollar dispositivos que sepan a través de sensores la cercanía que tiene la persona respecto a ellos, y actuar de forma acorde, ya que se conoce cual va a ser su tarea de antemano. Sin embargo, uno de los paradigmas de los ambientes inteligentes es que sea conciente del contexto, es decir, que pueda utilizarse tanto en una sala de conferencias como en una cocina o una habitación, y esto requiere de un razonamiento. Resulta impráctico tener que volver a programar los dispositivos dedicados para cada ambiente en particular. Emplearse sin involucrar tecnologías semánticas se volvería complicado por las siguientes razones:

- La programación dentro de la plataforma tendría que modificarse cada vez que la persona intente realizar una actividad nueva, debido a la dificultad de representar actividades de manera dinámica o automática.
- La introducción de nuevos dispositivos requiere de terceros para registrarlos. El usuario debe conocer de antemano diferentes factores del ambiente, lo que le vuelven más dependiente de intervención humana para que funcione de acuerdo a lo esperado.

Además que utilizar tecnologías semánticas facilita la interoperabilidad semántica entre los elementos que le controlan, se encuentran las siguientes ventajas:

- La representación de preferencias para adecuar un sistema se tornan más sencillas usando semántica. Los desarrolladores encontrarán mayor facilidad para poder representar dicha información relacionando conceptos como la iluminación y la actividad en vez de dispositivos y APIs.
- Al dotar de razonamiento espacial y temporal, se otorga a la plataforma la capacidad de desarrollar aplicaciones enriquecidas en contexto, ya que ofrece avisos de eventos (cambios de distancias) a las entidades interesadas que intervienen en su funcionamiento.
- Con tecnologías semánticas es posible introducir nuevos smart objects en el ambiente y, con una intervención mínima, pueden registrarse y adaptarse de acuerdo a las preferencias de las personas.

3.4 Conclusiones

Con la tecnología actual, es posible realizar esto con plataformas existentes o que están en fase de desarrollo. Sin embargo, no existe alguna plataforma que integre estas características dentro de su funcionalidad para facilitar el desarrollo de nuevas aplicaciones para ambientes inteligentes. El motivo del presente trabajo es introducir tecnologías semánticas y facilitar la programación y la escalabilidad de smart objects que son conscientes del espacio y del tiempo con dichas tecnologías. Al ser parte de una plataforma, todos los objetos pueden ser coordinados con base en reglas y aplicaciones que lleven la batuta del comportamiento de los ambientes de interés.

Gracias a esto, la idea de conciencia contextual puede ser portada a tecnologías semánticas e implementada en plataformas del Internet de las Cosas, considerando como la clave las distancias entre objetos e individuos y la hora del día para ajustarse a su comportamiento.

Capítulo 4. Componentes y descripción de la propuesta

En este capítulo se describirá la propuesta de tesis y las consideraciones importantes que describen el cómo se construirá, con qué características deberá contar, así como una explicación a grosso modo en cuanto a su funcionamiento, de manera que el lector pueda darse una idea del contenido de este trabajo. Este capítulo se compone de las siguientes secciones:

- **Requerimientos**. En esta sección se describe a grandes rasgos el funcionamiento y las características del trabajo desarrollado para esta tesis.
- **Razonamiento semántico**. Se describirá el funcionamiento del razonador requerido.
- **Dimensiones proxémicas**. Dentro del razonamiento se utilizan las dimensiones proxémicas para razonar sobre el espacio de acuerdo a la teoría del espacio de Hall (1990), y las dimensiones de este concepto introducidas por Greenberg *et al.* (2011).
- **Preferencia y privacidad de los usuarios**. En este apartado se describen las preferencias de los usuarios que permiten al sistema tomar cursos de acción y coordinar los smart objects dentro de la plataforma para que se adecúen acordemente.
- **Reconocimiento de Entidades**. Se detallará cómo se plantea descubrir la identidad de agentes humanos y virtuales dentro de la plataforma, útil para determinar el tipo de interacciones que se llevan a cabo dentro de un ambiente inteligente.
- **Ontología**. El modelo ontológico es descrito en esta sección, junto con las características de las clases implementadas y el motivo del por qué se encuentran dentro del trabajo para crear contexto.

4.1 Requerimientos

De acuerdo a Gárate *et al.* (2005), un ambiente dotado de inteligencia debe ser capaz de demostrar lo siguiente, si bien no de forma completa, al menos en cierta manera:

- Integrarse dentro del ambiente físico del usuario sin que se de cuenta.
- Reconocer a los usuarios y la circunstancias en la que se encuentran (actividades, estado anímico, etc.) y operar acorde.
- Tener un comportamiento predictivo basado en el conocimiento del ambiente (conciencia contextual), de los hábitos a quienes sirve y de actividades específicas del mismo cuando actúa.
- Producir nuevos servicios en tiempo real en los campos tales como entretenimiento, seguridad, salud, quehaceres, ambiente de trabajo, acceso a la información, cómputo, comunicación, etc., para mejorar la calidad de vida creando atmósferas adecuadas.
- Permitir el acceso a tantos servicios y características como se pueda, sin importar dónde se encuentre el usuario, o la posición de la cuál el usuario demanda dichos servicios y los artefactos disponibles en ese momento en particular (ubicuidad).
- Relacionar de una forma natural a los usuarios en términos de interfaces multimodales basadas en voz; leer movimientos y gestos; generar, emitir y proyectar imágenes; generar hologramas, etc.

Como se cita en el trabajo de Sadri (2011), se han identificado algunas características necesarias de los ambientes para ser aceptados de forma social. Sugieren los siguientes cuatro puntos:

- Facilitar el contacto humano.
- Orientado a la mejora de la comunidad y la cultura.
- Inspirar confianza y seguridad.
- Ser controlable por personas (switch de apagado al alcance).

Teniendo estas características en mente, se propone realizar un trabajo apegado a estos lineamientos, con el fin de brindar un módulo de programación dentro de una plataforma del Internet de las Cosas que tenga un comportamiento proactivo con base en las dimensiones temporales y espaciales de un ambiente determinado:

- **Razonamiento Temporal y Espacial**.
- **Comportamiento Proactivo**.
- **Adherencia a Plataforma del Internet de las Cosas (OpenThings)**.

4.1.1 Razonamiento Temporal y Espacial

Las dimensiones temporal y espacial proveen pistas importantes en el tipo de actividad que el usuario realiza, y de esta manera un sistema que sea consciente de ambas dimensiones puede reaccionar de la forma más oportuna (Cook *et al.*, 2009).

El razonamiento temporal y espacial abarca dos áreas bien establecidas en la Inteligencia Artificial, y han sido estudiadas a profundidad desde hace un par de décadas. Se han desarrollado formalismos y algoritmos bien establecidos para tratar con el razonamiento temporal, espacial, y espacio temporal (Sadri, 2011).

Actualmente se han realizado trabajos que emplean el razonamiento temporal y espacial para establecer una relación entre la actividad que están efectuando los usuarios de una casa inteligente de acuerdo a un conjunto de reglas, a partir de los estados de sensores que se encuentran en el hogar. En Augusto y Nugent (2004) se demuestra la importancia del plano temporal para determinar las decisiones a tomar en un lugar donde personas requieren ayuda debido a que suelen dejar aparatos operativos en el hogar (como la estufa) y que originan situaciones de peligro. Emplean el uso de Active Databases (ADB) (Paton y Díaz, 1999) y un razonamiento basado en reglas de la forma Evento-Condición-Acción (ECA) para establecer una acción de acuerdo a un conjunto de condiciones y eventos. Para representar el tiempo utilizan las 13 relaciones temporales propuestas por Allen (1983) y con base en un razonador temporal que utiliza la ADB es capaz de realizar una acción conforme a los estados de los sensores desplegados en la casa. Hay trabajos en los que se ha utilizado la lógica temporal de Allen para detección de anomalías dentro de ambientes, incluso se ha propuesto un algoritmo para su detección en un tiempo determinado (Jakkula *et al.*, 2007).

En un trabajo posterior, Augusto *et al.* (2008) incluyen el plano espacial para saber con cierto grado de certidumbre la ubicación de un usuario dentro de una casa y reconocer

el tipo de actividad que está realizando, de forma que aumentan la complejidad de las reglas y su predefinida sintaxis para integrar conceptos espacio temporales.

Estas dos dimensiones son útiles para entender elementos claves de una situación en desarrollo, con base en respuestas de sensores que están conectados a un sistema y que permiten reconocer variaciones entre los sensores como la actividad que una persona lleva a cabo.

4.1.2 Comportamiento Proactivo

En la AmI, uno de los objetivos que se pretenden alcanzar en este trabajo, es la visión de un futuro en el que los ambientes que rodean a las personas les ayuden a realizar actividades cotidianas. Esta visión de los ambientes es inobtrusivo, es interconectado, adaptable, dinámico, embebido e inteligente (Sadri, 2011).

La mayoría de los autores describen la inteligencia ambiental de una forma similar. La definición propuesta por Gaggioli (2005) describe la AmI “un ambiente digital inteligente, embebido, que es sensible y responsivo a la presencia de personas”. Este trabajo pretende mostrar un comportamiento proactivo para diferentes áreas del ambiente empleando razonamiento semántico, automatizando las tareas repetitivas y que pueden emplearse para aplicaciones de ahorro energético.

Al recabar información del contexto del ambiente, es decir, a través de sensores, de la ubicación de las personas dentro de un edificio, o las inferencias de una actividad como caminar, trabajar en un escritorio, etc., es posible que un sistema interactúe con una persona facilitando sus tareas, haciendo aquellas que se repiten. El propósito de esto es que el uso de razonadores semánticos dentro de aplicaciones que monitorizan estos ambientes provean la capacidad de tomar una decisión de acuerdo a la actividad que está ocurriendo. Sin embargo, es importante considerar que la naturaleza de los sensores y actuadores están limitadas por la veracidad de la información y su grado de certidumbre (**?**).

4.1.3 Adherencia a Plataforma del Internet de las Cosas (OpenThings)

Los requerimientos antes mencionados son la motivación principal para que los ambientes del Internet de las Cosas sean revestidos de semántica, sean capaces de reaccionar ante estímulos de entrada disparados por las personas. Existen dos componentes esenciales que no pueden faltar en esta clase de sistemas: una base de conocimientos y una máquina de inferencias (**?**).

Para poder implementar esta clase de tecnologías en ambientes que estén provistos por smart objects, se requiere de una plataforma que facilite la comunicación entre ellos con base en estándares definidos, así como usando tecnologías de la web semántica. Es por ello que se opta por trabajar con la plataforma para el Internet de las Cosas OpenThings (Delgadillo-Rodríguez, 2015), debido a que ofrece las características que se requieren para el presente trabajo:

- La comunicación entre servicios representa la información con el uso de tecnologías semánticas: RDF, OWL y SPARQL.
- Integra el modelo de Linked Data para enlazar datos generados en la plataforma con otros servicios semánticos en línea.
- La propuesta de Ambientes Virtuales que agrupan servicios en función de su ubicación física u otros conceptos abstractos, los cuales sirven como mecanismos para analizar el estado de los servicios y las interacciones que tienen entre ellos.

Dicha plataforma cuenta con la base requerida para dotarla de razonadores semánticos que puedan realizar inferencias sobre la información que los dispositivos representan en su funcionamiento.

4.2 Razonamiento semántico

En este apartado se describen las características de razonamiento principales que esta propuesta proveerá dentro de la plataforma. Éstas se componen de dos tipos de razonamiento esenciales que deben cubrirse para que un sistema pueda crear contexto a partir de hechos del ambiente en las dimensiones espacial y temporal.

- Razonamiento temporal
- Razonamiento espacial

4.2.1 Razonamiento Temporal

El razonamiento temporal surge de la necesidad de conocer de forma aproximada el tiempo en el que suceden los eventos dentro de un lugar (**?**). A partir de éste se pueden hacer inferencias, o determinar el tipo de acción que se debe tomar de acuerdo al lugar y al momento en que se encuentre una persona.

Si bien la complejidad de la temporalidad de los eventos surge como un tema de tesis por sí mismo, se decide considerarse dentro del estudio debido a que es posible representarlo por medio de ontologías y proveer razonamiento con base en la información provista por los eventos. En conjunto con el razonamiento espacial, el razonamiento temporal puede proveer un reconocimiento de actividades aún más acertado (**?**).

En este trabajo se tomó una ontología que es parte de de los fundamentos de OWL (OWL Working Group, 2012): *Time*, la cual permite representar instantes del tiempo así como intervalos de tiempo en los que suceden los eventos temporales para la Web Semántica (Hobbs y Pan, 2004). Los razonadores semánticos soportan esta ontología y por tanto fue seleccionado debido a su simplicidad y cumple con las características de la representación de eventos en el tiempo con base en un intervalo creado por dos instantes en el tiempo.

4.2.2 Razonamiento Espacial

Para determinar la importancia de los objetos en la cercanía, así como sus características y su identidad, es necesario el razonamiento que involucre la diferencia de distancia entre los objetos. Al tener en cuenta dicha información, se pueden establecer relaciones entre objetos y asimismo inferir una actividad con base en reglas definidas por los desarrolladores de aplicaciones para ambientes.

Estos cambios de distancias determinan el curso de acción a tomar, definido por las aplicaciones mencionadas. Sin embargo, mencionan el disparo de un evento cuando

una entidad humana se encuentra dentro de dicho ambiente (ya que son las entidades más móviles y el enfoque de dichos lugares). Los mismos objetos pueden proporcionar información importante para conocer otro tipo de relaciones con actividades causados por la interacción de las personas con ellos.

Con este propósito se desarrollaron varias ontologías que permitieran describir el espacio de forma adeucada para integrarlos dentro de un ambiente pensado originalmente para un edificio. Las ontologías desarrolladas pueden encontrarse más adelante en el capítulo Ontología.

4.3 Dimensiones proxémicas

Greenberg y Kuzuoka (1999) tomó la teoría de las distancias proxémicas dentro del cómputo ubicuo y encontró 5 categorías diferentes en las que se puede separar. Esto ayuda a encontrar un significado del espacio para apoyar a los dispositivos a adecuarse a las necesidades de un individuo. En la figura 10 podemos observar las cinco categorías que descubrió.

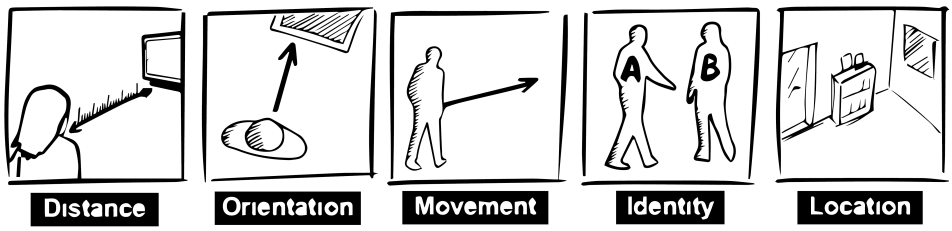

Figura 10**: Las dimensiones proxémicas en el cómputo ubicuo**

De las dimensiones proxémicas (Greenberg y Kuzuoka, 1999) que se consideran dentro del funcionamiento de este módulo de razonamiento espacial están las siguientes:

4.3.1 Identidad

Cada objeto dentro del ambiente posee una identidad que le diferencia de los demás. Es definido en cuanto a sus propiedades internas o externas, es decir, información relacionada con lo que tiene e información que lo describe en términos de pertenencia.

La forma en que se puede identificar un objeto va de acuerdo a las características individuales, si tiene capacidades de procesamiento o requiere de ayuda para poder es-

tablecer su identidad. En caso de contar únicamente con un código de barras o un código QR y sin capacidad de procesamiento, necesita de otro dispositivo para reconocer su identidad, ya que por sí mismo no es capaz de establecer comunicación con el Módulo Contextual.

4.3.2 Distancia

La distancia es un factor fundamental para conocer las implicaciones de las zonas proxémicas y su significado. Es una medida discreta en la que se define la zona en la que se encuentra una entidad respecto a otra. En este caso es utilizada para conocer qué tipo de relación tienen dos objetos, y sobre reglas definidas, se pueden establecer parámetros y delimitar un comportamiento proactivo de acuerdo a las necesidades y al contexto en el que se encuentran.

4.3.3 Ubicación

La ubicación describe el contexto físico en el que una entidad reside. Puede captar aspectos de su contexto, como cuando cierta entidad pasa cierto umbral para indicar su presencia dentro de una habitación. El presente trabajo pretende ofrecer un grado mayor de contexto al conocer el lugar físico en donde se encuentre cierta entidad. A partir de aquí se puede establecer opciones de privacidad y preferencias de usuarios, e. g. una caja de galletas es pertenencia de la persona que es dueña del cubículo.

4.3.4 Orientación

La orientación es la forma de definir la posición y el ángulo que una entidad tiene respecto a un lugar o entidad. Tiene sentido únicamente si algo tiene una cara frontal, o un lugar en donde existe el término de “frente” y “detrás”. Con este dato se puede conocer la actividad de una persona respecto a otro objeto, ya que ofrece una idea de acuerdo a la atención que se le está prestando. Además de que puede dar una idea aproximada de la tarea del usuario, permite ahorrar energía y tener un comportamiento proactivo ante estímulos del usuario.

4.4 Cambios de distancias

Para determinar cuando un objeto entra a cierta zona o se aleja de ella es necesario establecer parámetros que permitan conocer con cierta frecuencia establecidas por la aplicación que requiere conocer este dato o por las capacidades de procesamiento y de energía con que los dispositivos cuentan. Un cambio de distancia discreto es un evento que funciona como disparador de un evento, y es representado como un cambio de zona proxémica.

El proceso que se sigue para este propósito es el siguiente:

1. Entidad envía a una aplicación la distancia respecto a otro objeto.
2. El módulo actualiza la distancia entre dos objetos.
3. Corre las reglas necesarias para categorizar la distancia dentro de las cuatro zonas proxémicas.
4. Una aplicación se encarga de verificar, de acuerdo a su comportamiento, si necesita enviar una orden a otros objetos para adecuarse de acuerdo a su programación.

```
<rdf:Description rdf:about="http://www.semanticweb.org/mc/ontologies/2015/2/context-ontology#mc-galaxy-alpha">
    <locations:hasPlace rdf:resource="http://www.semanticweb.org/mc/ontologies/2014/9/locations#UbiquitousComputingLaboratory"/>
    <proxemics:canBeMoved rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</proxemics:canBeMoved>
    <proxemics:hasUID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">mc-galaxy-alpha</proxemics:hasUID>
    <rdf:type rdf:resource="http://www.semanticweb.org/mc/ontologies/2015/1/proxemics#SemiFixedFeature"/>
    <rdf:type rdf:resource="http://www.semanticweb.org/mc/ontologies/2015/1/proxemics#Object"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.semanticweb.org/mc/ontologies/2015/1/proxemics#mc-galaxy-alphaUCB5739Distance">
    <rdf:type rdf:resource="http://www.semanticweb.org/mc/ontologies/2015/1/proxemics#IntimateClosePhase"/>
    <proxemics:hasDistanceValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</proxemics:hasDistanceValue>
</rdf:Description>
```

Figura 11**: Individuos de la ontología que muestra una ubicación y una distancia proxémica de "mc-galaxy-alpha" respecto a otro objeto, el cual es el Beacon "UBC5739".**

Los desarrolladores de objetos inteligentes o de los intermediarios que se utilizan para controlarlos requieren enviar constantemente la actualización de las distancias. La frecuencia y la precisión con la que estos datos se transmiten a una aplicación determinará el tiempo de respuesta del sistema. De la misma forma, es aconsejable que la información de las distancias se provea en forma de dimensión proxémica (es decir, tomando los valores y categorizándolos previamente antes de enviarlo), así como se recomienda

establecer umbrales de actualización para evitar problemas, como un comportamiento con falsos positivos o falsos negativos.

4.5 Cercanía entre entidades

A partir de los cambios de distancias, la cercanía o lejanía entre entidades permite conocer la actividad llevándose a cabo con la finalidad de adecuarse a cierto ambiente, de acuerdo a las preferencias de una entidad (usuario o administrador del mismo).

Estos eventos de entrada/salida de zonas proxémicas son capturadas por una aplicación del ambiente inteligente para enviar órdenes a los smart objects para adecuarse conforme al rol que tienen.

Asimismo, se emplean otras dimensiones auxiliares para establecer un contexto enriquecido con otros aspectos del ambiente, tal como cuando una persona entra dentro de un laboratorio y tiene permisos para controlar la iluminación y el uso de las computadoras, de los cuales se detallan más adelante en la sección Preferencia y privacidad de los usuarios.

A través de la identidad, la distancia, y la ubicación, se pueden establecer reglas más complejas para la interacción del humano con su ambiente. En este caso se pueden considerar los permisos de las personas para que los smart objects actúen de forma acorde, o por el contrario, sólo respondan a estímulos de una persona o un rol en particular.

4.6 Partes interesadas en el desarrollo del trabajo

Quienes podrían interesarse en este trabajo y en su desarrollo son aquellos que consideran el uso de dimensiones proxémicas, una representación y obtención del significado del espacio que existe entre los usuarios respecto a los objetos de su ambiente. Las ramas interesadas son interacción humano-computadora, Ambient Intelligence, Ambient Assisted Living, entre otras, que requieran establecer aplicaciones conscientes del tiempo y del espacio en conjunto. Con lo que esta propuesta define se puede ofrecer un comportamiento proactivo, para cuestiones en las que resulta impráctico o muy complicado el accionar dispositivos, o controlar manualmente la iluminación, cortinas, y demás

factores que pueden ser delegados a objetos inteligentes, cuya función sería servir de actuadores para con el ambiente. Más adelante se hablará con más detalle la justificación del por qué utilizar este trabajo en vez de otros enfoques menos pragmáticos.

4.7 Preferencia y privacidad de los usuarios

Para definir aspectos de la privacidad de los usuarios de un ambiente inteligente se requieren conocer sus preferencias a diferentes tareas que lleve a cabo en un lugar. Para ello, se debe relacionar el individuo con la Base de Conocimientos de la pieza de software con capacidades de razonamiento.

Debido a que resulta una tarea complicada y contraproducente el modelar cada detalle de cada preferencia de una persona en cuanto a sus tareas debido a lo enriquecida que es, y es una cuestión importante a considerar al momento implementar esto dentro de una ontología (Gruber, 1995), se ha optado por poner dentro de la Base de Conocimientos las clases generales y los roles para colocarse en las entidades, acorde a lo que dice el autor en el artículo mencionado anteriormente. Los desarrolladores son los encargados de establecer las relaciones y describirlas de acuerdo a preferencias estándar o por defecto, o si es necesario incluir alguna descripción que describa de forma más detallada la preferencia del usuario. El nivel de detalle le corresponde al desarrollador considerarlo. Sin embargo, es importante considerar los siguientes lineamientos para las ontologías de preferencia, tomado del trabajo de Gruber (1995):

- **Claridad**. Una ontología debe comunicar de forma efectiva el significado de los términos.
- **Coherencia**. Debe ser coherente. Es decir, debe poder obtener inferencias que son consistentes con las definiciones.
- **Extensibilidad**. Debe ser creada pensando para usarse en un vocabulario compartido con otros contextos.
- **Parcialidad mínima de la codificación**. Es importante cuidar una conceptualización a nivel ontológico e independiente de símbolos que son parte de una codificación, ya que otros vocabularios pueden utilizar otro tipo de codificación.

- **Compromiso ontológico mínimo**. Una ontología debe hacer pocas aserciones del mundo a ser modelado.

Dentro de la ontología se describen tres tipos de preferencias básicas que pueden definir las preferencias más generales y comunes para los usuarios de un edificio inteligente en el que se busca el máximo de ahorro energético.

Cabe destacar que estos perfiles son generales y deben ser considerados por defecto, y que en caso de que se desee tener cierto perfil para una actividad, ésta debe ser expresada en dicha actividad o evento que sea impartido o del cuál tenga prioridad. Se debe considerar prioritario el perfil establecido en el evento sobre el que la persona tiene.

Hay trabajos como el de Kabir *et al.* (2014) que utilizan taxonomías pero para representar actividades de un trabajador en una oficina, de manera que los interesados en su actividad puedan ver si está ocupada o no. De esta manera también es posible denotar que cuando una persona se encuentra ocupada, es posible inferir que no desea ser interrumpida a menos que el asunto que desea tratar es de importancia.

Las siguientes preferencias fueron tomadas del trabajo de Ahmed *et al.* (2008) que utiliza una taxonomía similar para la información y el acceso a datos. Contiene dos tipos de preferencias básicas (permitir acceso y denegar acceso), pero se ha decidido añadir una tercera intermedia que permita ser más flexibles y están compuestas por dos tipos de privacidad: pública y privada.

i. **Preferencia Pública**. La preferencia pública define varios aspectos del ambiente y las preferencias del usuario. Cuando se define esta preferencia en el perfil de una persona, es posible establecer lo siguiente:

   - Tiene una consideración no privada en cuanto a sus tareas.
   - No maneja información confidencial o es de poca relevancia para otras personas.

   A partir de este perfil se establece que los aspectos de la seguridad son disminuidos

al mínimo. En este caso puede ejemplificarse en la opacidad de un cristal inteligente que se encuentra en una sala de juntas.

ii. **Preferencia Privada**. Una preferencia privada establece que cualquier actividad de la persona debe ser considerada como confidencial, y se tienen que tomar las medidas para ocultar información sensible.

- Se debe cuidar la privacidad en la medida de lo posible.
- Se maneja información confidencial siempre, a menos que sea declarado lo contrario.

iii. **Preferencia Semi Pública**. En ciertas ocasiones es necesario establecer una preferencia semi pública debido a que hay actividades que una persona realiza que son confidenciales mientras que otras pueden tratarse como públicas. Este perfil establece que:

- La persona debe expresar el perfil en el evento con el que se desea trabajar. Por defecto, su actividad se considera como privada.

Debido a esto, es necesario establecer el tipo de perfil que tiene un evento. El software razonador toma el perfil por defecto de las actividades que se realicen, de acuerdo a parámetros establecidos previamente por un administrador al momento de dar de alta los perfiles de actividades, e. g. juntas, presentaciones, reuniones de negocios, etc.

4.7.1 Preferencias de Usuarios

De la misma forma que la privacidad, los gustos de las personas cambian el ambiente de una forma más notoria. Se pueden ver reflejados en los Smart Objects que controlan la iluminación, las persianas, la temperatura de la habitación, entre otros factores que se consideran para llevar a cabo la actividad de un usuario.

Es imposible modelar una preferencia detallada para cada aspecto del ambiente, pero se deja establecida una preferencia general la cuál puede ser usada por los desarrolladores o el administrador de la Base de Conocimientos para establecer la diferente gama de perfiles que se requieran.

En esta propuesta se muestra el uso de un perfil para modelar la preferencia de iluminación y del apagado o encendido de una pantalla de una computadora, de acuerdo a la cercanía y a la ubicación de la persona respecto a éste. De acuerdo a parámetros del ambiente, se establece que en cierto horario, en el que se sabe que hay iluminación natural suficiente para trabajar, la iluminación artificial se mantendrá apagada, y fuera de ese horario, si la persona está dentro de la oficina, la iluminación artificial es encendida de forma automática. De la misma manera se puede establecer el tipo de actividad que está realizando una persona en su escritorio dependiendo de la hora del día.

Los perfiles mencionados anteriormente sirven para propósitos definidos. Siempre que se desee tener un perfil por defecto para cierta actividad, los desarrolladores deben considerar el mejor uso de los recursos, ya que de esta manera se puede explotar el potencial de un ambiente cuya plataforma puede razonar sobre los hechos del ambiente.

4.8 Reconocimiento de Entidades

El reconocimiento de las entidades dentro de la plataforma es independiente de la tecnología de comunicación, y permite establecer perfiles para cada entidad. Al momento en que un dispositivo se detecta (e.g. se encuentra en el rango de alcance de Bluetooth), será posible consultar su información a través de una API REST. En caso que se requiera, se debe incorporar una antena junto a un objeto que sirva de puente entre la tecnología inalámbrica a emplearse con la red IP. Un caso concreto es para representar una ubicación física dentro de un edificio, y obtener la cercanía respecto a ésta entidad.

La forma en cómo las entidades son identificadas depende del tipo de dispositivo, y existen varios métodos. En el caso de un smartphone, por ejemplo, es posible utilizar la MAC de la antena Bluetooth o WiFi para identificarlo de forma única. En el caso de este trabajo, se predefine un código que le permite identificarse por medio del nombre del dispositivo.

En esta propuesta se utiliza este identificador para realizar la búsqueda en la Base de Conocimientos para descubrir su descripción semántica.

4.9 Ontología

La ontología desarrollada en el presente trabajo, Context Ontology, permite representar las dimensiones requeridas por desarrolladores, para brindar un comportamiento proactivo a ambientes compuestos por objetos del Internet de las Cosas. Se puede encontrar las relaciones entre sus clases en el apéndice 9.2.

Context Ontology está compuesta por tres ontologías, dedicadas a modelar su propio dominio y separadas con fines de reutilización. Éstas se encuentran disponibles como ontologías separadas, por lo que es posible actualizarse cada una por separado y un manejo ordenado de las versiones. Las ontologías son las siguientes:

- Privacidad.
- Lugares.
- Proxémica.

En el resto de esta sección se hablará de cada una de las clases que le conforman para detallar en sus propiedades y detalles importantes de cada ontología.

4.9.1 Privacidad

La ontología Privacy, propuesta para este trabajo con base al trabajo de (Ahmed *et al.*, 2008), representa preferencias de privacidad de usuarios, lugares, objetos y eventos que surgen en el tiempo mencionadas en la sección Preferencia y privacidad de los usuarios. Consta de tres clases principales:

i. **Private Privacy Preference**. Define una preferencia de privacidad que indica que todo es considerado como privado y nada asociado a la entidad debe ser considerado como público.

   Es una representación básica para lugares muy privados, i.e. una habitación, un lugar de trabajo, etc. Cada objeto que tiene relación con el lugar indicado con esta

privacidad hereda su privacidad, y por lo tanto, no existen permisos para las personas ajenas, a menos que sea explícitamente expresado en la descripción de las entidades.

ii. **Public Privacy Preference**. Define una preferencia de privacidad que un usuario especifica para cierta actividad como pública.

Esta clase define algo como público. En el caso de lugares de acceso público no puede existir ninguna preferencia de privacidad, a menos que sea expresada por el administrador del lugar. Todo lo que está contenido dentro de éste lugar debe ser considerado como público, a menos que sea expresado de forma contraria.

iii. **Semi Public Privacy Preference**. Describe que algo debe ser representado con una regla semi-pública.

Se refiere a lugares en donde pueden existir eventos tanto privados como públicos. Ésta preferencia no hereda nada, pues es ambigüa, y requiere que las entidades se expresen en términos de esta preferencia.

4.9.2 Lugares

La ontología Places define una serie de clases que representan lugares físicos, de acuerdo con la percepción que tenemos de ellos, y que son delimitados por puertas o algún umbral definido por la arquitectura.

Dicha ontología fue realizada considerando el ambiente de ejemplo que se tiene dentro del departamento de Ciencias de la Computación en CICESE. Sin embargo, ofrece el soporte suficiente para representar otras clases de ambientes con las clases generales que se han propuesto dentro de la taxonomía.

En la ontología se definen las siguientes clases:

i. **Building (Edificio)**. Una edificación que puede contener lugares (Places).

Se define como un contenedor principal de lugares. Puede representar edificios como un departamento entero, un edificio administrativo, un complejo departamental, etc.

ii. **Corridor (Corredor, pasillo)**. Un espacio que define algo que lleva a lugares (Places) dentro de un edificio (Building).

Representa un pasillo que lleva a diferentes lugares. Es la vía que se tiene para llegar a un lugar. Por lo tanto, se requiere que exista un corredor dentro de un edificio, uno al menos para cada lugar que contiene el edificio (Building). Puede obviarse dentro de las aplicaciones, sin embargo, al añadirse a la descripción semántica de un lugar es posible establecer un nivel de granularidad en las aplicaciones que requieran establecer lo que sucede cuando una persona esta cerca de cierto lugar, para proveer un nivel de proactividad mayor.

iii. **Entrance (Entrada)**. Algo que representa una entrada o salida de un lugar (Places).

Representa un lugar físico o imaginario que define la separación entre dos lugares. Sirve para establecer relaciones topológicas entre entidades respecto a las entradas de los lugares. Pueden existir tanto en edificios, pasillos, o lugares.

iv. **Room (Cuarto)**. Es un lugar que define algo que puede contener entidades y objetos. Es usado para actividades específicas o grupos de actividades dentro de edificios (Building).

Dependiendo de las actividades, o el rol para que se usen, son caracterizadas dentro de los siguientes apartados.

iv.i. Auditorium (Auditorio). Un lugar usado para pláticas y presentaciones dentro de un edificio (Building).

iv.ii. Bathroom (Baño). Un lugar que contiene inodoros y/o mingitorios.

iv.iii. Caffeteria (Cafetería). Lugar que es usado para servir y/o comer comida.

iv.iv. Classroom (Aula de clases). Lugar usado para impartir clases dentro de un edificio (Building).

iv.v. JanitorRoom (Cuarto de intendencia). Un lugar usado para almacenar herramientas y químicos de limpieza.

iv.vi. Laboratory (Laboratorio). Lugar usado para llevar a cabo experimentos con herramientas especializadas.

iv.vii. MeetingRoom (Sala de juntas). Lugar en el que juntas, videoconferencias y/o presentaciones con poco público son llevadas a cabo.

iv.viii. Office (Oficina). Lugar usado para llevar a cabo actividades administrativas dentro de un edificio (Building).

iv.ix. TestingRoom (Cuarto de pruebas). Lugar donde se prueban ciertas actividades, usualmente involucrando a personas para experimentos.

iv.x. Workspace (Espacio de trabajo). Lugar usado para actividades de trabajo. Usualmente usado por ciertas personas, ofreciéndole prioridad a alguna de ellas.

v. **Stairway (Escaleras)**. Usado para moverse entre los pisos de un edificio (Building).

4.9.3 Proxémica

Parte de las dimensiones proxémicas, vistas en el capítulo Marco Teórico, en la sección 2.3, se encuentran modeladas dentro de esta ontología (Proxemic Dimension). En estas se incluyen en dos grandes categorías:

i. **Distance (Distancia)**. Clase que describe un concepto de Distancia para interacciones proxémicas.

   Cada distancia contiene dos fases que serán descritas más adelante en forma de reglas.

   i.i. Intimate Distance (Distancia Íntima). Clase que caracteriza un valor de distancia aproximado que va desde los 0 a los 46 cm. Identifica una interacción íntima entre entidades, es decir, que guardan una relación muy estrecha entre ellas.

   i.ii. Personal Distance (Distancia Personal). Clase que caracteriza un valor de distancia desde los 47 a los 122 cm. Identifica una interacción en la que se guarda una relación, sin embargo no tan estrecha como la Distancia Íntima.

i.iii. Social Distance (Distancia Social). Clase que caracteriza un valor de distancia desde los 123 hasta los 370 cm. Identifica una interacción menos intrínseca que una Distancia Personal. Usada para establecer relaciones en las que no existe una interacción directa, pero se puede llevar a cabo.

i.iv. Public Distance (Distancia Pública). Clase que caracteriza un valor de distancia a partir de los 371 cm. Identifica una interacción indirecta con cualquier entidad.

ii. **Identity (Identidad)**. Una clase que describe un identificador único para una entidad en particular.

iii. **Feature (Característica)**. Establece el aspecto de movilidad de una entidad.

iii.i. Fixed Feature (Característica Fija). Describe algo que es parte del amueblado de un ambiente y que no puede ser fácilmente movido de su lugar.

iii.ii. Semi Fixed Feature (Característica Semifija). Describe algo que puede ser movido fácilmente. Aplica a objetos normales como objetos, libros, lápices, etc.

4.10 Reglas

Como se mencionó anteriormente, existen reglas que corren sobre el razonador semántico para poder categorizar cada una de las distancias proxémicas que existen, en cada una de sus dos fases (cercana y lejana).

Para lograr esta categorización se emplearon 8 reglas, dos por cada distancia proxémica. Dichas reglas fueron elaboradas empleando el lenguaje para reglas de Jena para utilizarse con su razonador genérico basado en reglas (GenericRuleReasoner). La tabla 1 muestra las zonas proxémicas, con sus respectivas fases, y su cota superior e inferior para cada una de ellas. Las distancias proxémicas pueden observarse en forma de reglas en el apéndice 9.3.

Las tecnologías semánticas que se emplean en el presente trabajo se mencionan a continuación.

Tabla 1**: 8 reglas para las zonas proxémicas con sus fases.**

Proxemic Zone	Low Boundary	High Boundary
IntimateClosePhase	0 cm	15 cm
IntimateFarPhase	16 cm	46 cm
PersonalClosePhase	47 cm	76 cm
PersonalFarPhase	77 cm	122 cm
SocialClosePhase	1.23 m	2.10 m
SocialFarPhase	2.11 m	3.70 m
PublicClosePhase	3.71 cm	7.60 m
PublicFarPhase	7.61 m	...

4.10.1 Razonador Semántico

El razonador semántico está basado en la tecnología de **Apache Jena** (The Apache Foundation, 2014). Para los propósitos de este trabajo, el razonador **GenericRuleReasoner**, con soporte para reglas, permite realizar las inferencias necesarias para razonar sobre la información ofrecida dentro de la *Base de Conocimientos*.

4.10.2 OWL/RDFS

Las tecnologías empleadas para las ontologías y el razonador son OWL y RDFS. OWL se utiliza para el diseño y la representación de ciertos elementos, tales como los Individuos, dentro de **Protégé 5.0 beta**. Una vez empotrada la Base de Conocimientos dentro del razonador, se utiliza RDF y RDFS para representar la información que contiene el modelo semántico.

4.11 Conclusión

En este capítulo se describió a grandes rasgos el funcionamiento de la propuesta de tesis a ser desarrollada, así como sus características principales para llevar a cabo el objetivo principal del trabajo. Se describieron las dimensiones proxémicas y se abarcó la forma en que se abarcarían en el trabajo, las preferencias de los usuarios y un funcionamiento general para integrarlo dentro de la plataforma OpenThings (Delgadillo-Rodríguez, 2015).

Se describió además la ontología completa con las correspondientes descripciones de sus clases, la función que cumple cada clase, y sobre el razonador semántico y el

uso de reglas para categorizar las distancias proxémicas de acuerdo a su distancia en centímetros.

Capítulo 5. Arquitectura

Para poder comprender el funcionamiento de la arquitectura que se explicará más adelante, es importante definir algunos elementos que se utilizan a lo largo de este trabajo, para que el lector pueda ser visualizar con mayor facilidad la forma en que los diferentes elementos del sistema interactúan. Estos elementos son los siguientes:

- **HubServer**. Esta entidad de software implementa el protocolo ligero de Internet llamado PubSubHubbub (Network Working Group, 2014). Permite una comunicación asíncrona entre las entidades que conforman el sistema con base en un modelo de publicador - subscriptor. Más adelante, en la sección Registro de Ambiente Virtual se habla más detalladamente su funcionamiento y su rol dentro de la arquitectura.

- **VirtualEnvironment**. Es una pieza de software tomada de OpenThings (Delgadillo-Rodríguez, 2015) para representar un ambiente físico, en este caso una sala inteligente en donde coexisten humanos y smart objects. Se utiliza como el punto focal de este trabajo debido que aquí es donde se encuentra la pieza de software que contiene el razonamiento y la ontología.

- **ContextApplication**. Esta es la aplicación dentro de la plataforma que contiene una base de conocimientos, una instancia del razonador semántico basado en reglas, y es integrado dentro de un ambiente virtual de OpenThings para comunicar eventos como cambios de dimensiones proxémicas.

- **Beacon**. Es un dispositivo que sirve como punto de referencia para ubicar una persona respecto a un ambiente virtual. Es de ayuda para conocer si una persona se encuentra fuera o dentro de un lugar y cerca o lejos de un smart object.

- **TripleStoreServer**. Es un servicio del sistema que contiene una base de conocimientos, y su función reside básicamente en ofrecer un servicio REST que regrese el nombre de un ambiente virtual dado el identificador único de un Beacon.

- **Device**. Este dispositivo es la parte que representa un dispositivo móvil que representa a un usuario. Se encarga de la búsqueda y cálculo de la distancia entre éste y un Beacon, ya sea parte el que representa a un ambiente o de un smart object.

- **Smart Object**. Son los objetos que conforman la parte que interactúa con el ambiente físico, pero que tienen su representación virtual y son capaces de recibir órdenes de HubServer para adecuar los aspectos físicos del ambiente (iluminación, control de cortinas, apagado y encendido de dispositivos, etc.).

Dentro de este capítulo se hablará acerca de la arquitectura y el funcionamiento de la propuesta del trabajo, con los detalles técnicos y los diagramas necesarios para explicar cómo se lleva a cabo la comunicación entre los diferentes elementos de la plataforma, una vez introducido la pieza de software que contiene el razonamiento y la ontología.

También se hace referencia a cómo se llevó a cabo el proceso de cohesión de acuerdo a los lineamientos propuestos por Delgadillo-Rodríguez (2015), integrando así el trabajo dentro de las características necesarias para cumplir con sus requerimientos. Asímismo, se hablará sobre plantillas desarrolladas para facilitar la creación de nuevas aplicaciones para ambientes inteligentes, tomando como base los ejemplos y clases de la plataforma OpenThings.

5.1 Conexión con ambiente virtual

Para que suceda una interacción entre un usuario y a un ambiente, se deben considerar los siguientes aspectos fundamentales para recabar información esencial para reconocer el tipo de actividad y la identidad de la persona:

- Identidad del usuario.
- Distancia y ubicación respecto al ambiente y a los objetos cercanos.
- Orientación respecto a los objetos cercanos.

En la figura 12 se muestra el proceso de ubicación de una persona. Interviene una persona que posee un device con sistema operativo Android, el cuál es utilizado como dispositivo de ubicación para propósitos de este trabajo.

Cada vez que un nuevo Beacon es descubierto por el device, es importante conocer el VirtualEnvironment que lo contiene, es decir, el identificador que representa el Beacon del

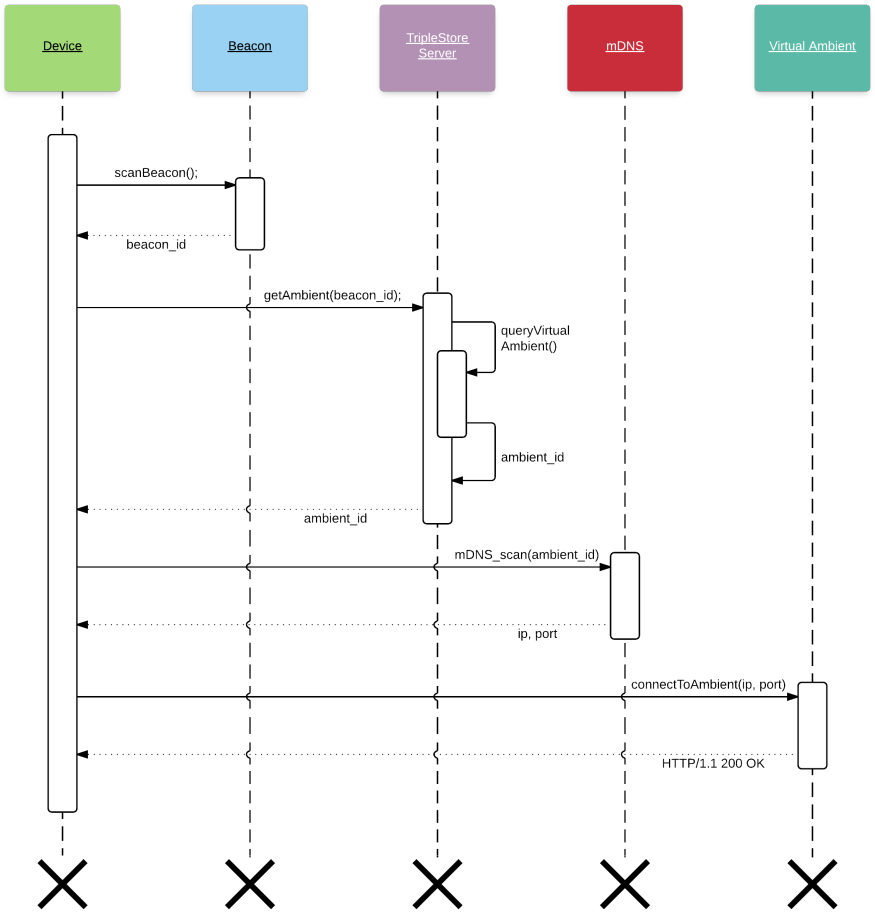

Figura 12**: Diagrama de secuencia para la conexión con un ambiente virtual.**

lugar físico. En este caso, los Beacon deben proveer un identificador único y asímismo el tipo de Beacon del que se trata, ya que puede existir un Beacon para representar un objeto físico o un lugar.

Para este proceso, el device debe obtener el identificador y enviarlo al servicio correspondiente para solicitar el nombre del ambiente que está representando. Como resultado, se obtendrá el nombre del ambiente el cuál le distingue de los demás en el proceso de búsqueda por mDNS. Una vez obtenido comienza el proceso de búsqueda por mDNS, y al ser encontrado, comienza el proceso de conexión con el Ambiente Virtual. El proceso llevado a cabo se describe a continuación:

1. **Buscar Beacons en la cercanía.** Cada cierto período de tiempo se buscan Beacons en el rango de alcance del dispositivo, y de esa manera calcular la distancia

aproximada respecto a estos Beacons. Se obtiene la distancia entre ambos usando la intensidad de la señal de lectura de los mensajes de ubicación que se envían entre los Beacon y el dispositivo (RSSI). A pesar de que la intensidad de la señal no es un buen indicador para aproximar la distancia debido a que puede ver afectada por obstáculos en el ambiente, es un método práctico para usarse dentro del laboratorio, en el que se usan condiciones ideales.

2. **Encontrar la ubicación que representa el Beacon.** Para este procedimiento se necesita previamente conocer los identificadores de los Beacons, así como la distancia relativa que existe entre ellos y el dispositivo. Éste debe hacerse cargo de tomar su identificador (provisto desde el escaneo realizado por Bluetooth Low Energy) y enviarlo por un mensaje GET del protocolo HTTP a un servicio que contiene información relacionada con él. Este servicio se conoce dentro de la plataforma como **TripleStoreServer**. Éste contiene un diccionario dentro de una Base de Conocimientos que asocia por medio de la propiedad *hasBeacon* la ubicación del edificio con el Beacon encontrado por el escaneo. Es enviada una petición a este servidor con el parámetro *beacon-id* para obtener el nombre del Ambiente Virtual como respuesta.

3. **Descubrir el Ambiente Virtual con base en el nombre obtenido de TripleStoreServer.** Una vez obtenido el nombre del Ambiente Virtual con la respuesta a la petición al servidor, se comienza el proceso de búsqueda por mDNS. Previamente, el Ambiente Virtual debe encontrarse en la red para poder comenzar con el proceso de conexión con el dispositivo, y publicando su existencia a través del mismo protocolo.

4. **Conectar con el Ambiente Virtual.** La conexión finalmente se lleva a cabo si la distancia entre los dos es suficientemente cercana para inferir que la persona se encuentra dentro del lugar. Al resolverse el nombre se conecta. Se envía un mensaje para garantizar que la conexión efectivamente existe y para actualizar el estado de conectado en ambas entidades de la plataforma.

5.2 Registro de Ambiente Virtual

Para enviar mensajes y seguir los lineamientos de OpenThings, todos los elementos que necesiten enviar y recibir información sobre eventos necesitan registrarse en un servicio virtual llamado **HubServer**. Está basado en un paradigma de intercambio de información *Publisher-Subscriber* (publicador-suscriptor), propuesto en un protocolo abierto para comunicación distribuida en Internet llamado Network Working Group (2014). Un ejemplo de una clase en Java puede observarse en las figuras 13, 14.

```
private static final PushInfo pushInfo = new PushInfo(
        TOPIC,
        "http://" + jmdns.getLocalAddressAsString("eth0") + ":" + SERVICE_PORT + "/callback",
        UUID.randomUUID().toString());
```

Figura 13**: Ejemplo de un objeto PushInfo que indica el tipo de tema al que un smart object se registrará en HubServer.**

```
@Override
public void serviceResolved(ServiceEvent serviceEvent) {
    String ip = serviceEvent.getInfo().getHostAddresses()[0];
    int port = serviceEvent.getInfo().getPort();

    if(serviceEvent.getName().equals(PSH_SERVER)){
        pushInfo.setHub(ip, port);
        router.sendSubscriptionRequest();
        System.out.println("Connected to Hub Server.");
    }
}
```

Figura 14**: Ejemplo de código en el que se envía una subscripción en cuanto se ha encontrado HubServer en la red.**

Para que el proceso de intercambio de mensajes ocurra con el HubServer se necesita registrar como *publicador*, y subscribirse para un evento que es de importancia. El Ambiente Virtual debe ser caracterizado por un puerto de IP definido por el desarrollador y un nombre según la plataforma. Asimismo se requiere establecer el comportamiento que se necesite para los diferentes eventos que puedan ocurrir en el ambiente de acuerdo a cambios de distancias y eventos en el tiempo.

Con la información provista, el Ambiente Virtual se registra dentro de la plataforma. El desarrollador no tiene por qué preocuparse de los detalles de programación. Se ha fa-

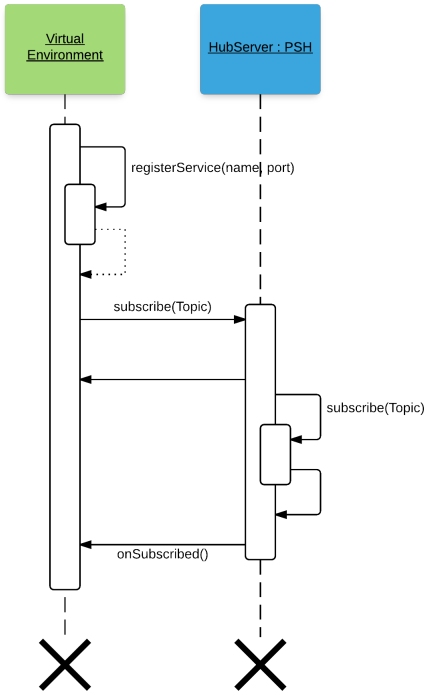

Figura 15**: Registro de un Ambiente Virtual con HubServer.**

cilitado con este propósito la clase *VirtualEnvironment* de manera que una persona cree su propio ambiente heredando de ésta y solo encargándose de proveer la información necesaria para la integración con la plataforma. En la imagen 16 puede observarse un ejemplo básico de la información y los métodos que deben llamarse cuando se registra la aplicación. La clase *VirtualEnvironment* maneja de manera interna la conexión y el registro con HubServer, de manera que los desarrolladores no necesitan conocer su funcionamiento.

El desarrollador debe encargarse de capturar los mensajes que provengan del HubServer, así como el envío de mensajes cuando suceda algún evento. Esto permitirá a la plataforma comunicar mensajes a los smart objects suscritos al evento publicado.

En la figura 17 se puede observar un diagrama de secuencia que muestra cómo es que el evento es disparado por el device y continúa el intercambio de mensajes hasta finalmente llegar a los Smart Objects. Internamente lo que sucede es un proceso de razonamiento debido a un cambio en la ontología. Este proceso se puede separar en cuatro partes importantes:

```
/**
 * This is a template for Virtual Environments, that manages everything to connect to OpenThings platform.
 *
 * Created by mc on 20/07/15.
 */
public class VirtualEnvironmentTemplate extends VirtualEnvironment
        implements VirtualEnvironmentInterface, VirtualEnvironmentEvents {

    private static final String TEMPLATE_ENVIRONMENT = "UbiquitousComputingLaboratory";
    private static final int TEMPLATE_ENVIRONMENT_PORT = 3000;

    private boolean isInsideEnvironment = false;

    public VirtualEnvironmentTemplate () {
        super("p3p2");

        // If we want notifications on this Environment, we must register it in parent class.
        registerForCallback(this);

        //We must name this Virtual Environment
        setEnvironmentInfo(TEMPLATE_ENVIRONMENT, TEMPLATE_ENVIRONMENT_PORT);

        // Starting contextual Application
        getApplication().setContextApplication(ContextApplication.getInstance());

        run();
    }
```

Figura 16**: Ejemplo de uso de una clase en la plataforma OpenThings.**

1. Envío de evento (device).
2. Recepción de evento (VirtualEnvironment).
3. Actualización y razonamiento (ContextApplication).
4. Procesado de información y envío de órdenes a HubServer (HubServer).

Cada una de estas partes conforman la operación del proceso de ubicación, y además del cambio de distancias y orientación, así como otras dimensiones proxémicas. Una aplicación dentro del device permite intercambiar información con *VirtualEnvironment* asociado al Beacon de la cercanía (que indica el lugar físico en donde se encuentra la persona). El *VirtualEnvironment* recibe la información del *device* y la envía a *ContextApplication*, encargada del manejo y actualización de la *ontología*. Éste elemento es el que contiene las capacidades de razonamiento dentro de la plataforma. Su funcionamiento se ilustra en la figura 18.

5.3 Smart Objects

Los smart objects se deben registrar directamente en el HubServer, suscribiéndose al tema de interés que les corresponde. Cada uno de ellos debe estar asociado a uno o más

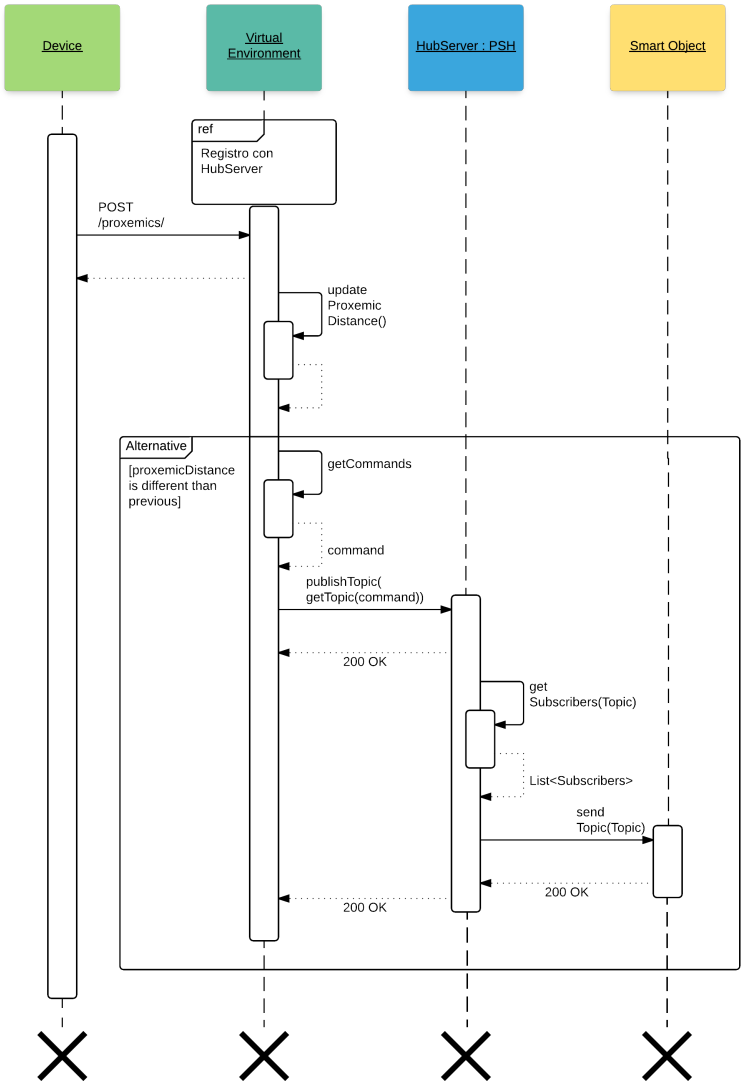

Figura 17**: Diagrama de secuencia para enviar un Topic a HubServer, y su correspondiente manejo y notificación a Smart Objects.**

temas (Topics) para que sean notificados de algún cambio dentro del sistema. Ellos son la parte final del proceso de razonamiento, en el que se ve reflejado el comportamiento proactivo de la plataforma. Su actuación depende intrínsecamente del Ambiente Virtual y las preferencias del usuario, las cuales definen el estado y el control de operación de acuerdo a la propiedad física que tengan, como la iluminación y la intensidad de opacidad de los cristales inteligentes.

El proceso de registro de los Smart Objects es similar. Ellos deben ser capaces de comunicarse con HubServer de la misma manera en que lo hacen los Ambientes Virtuales. Mediante este proceso se caracterizan las funciones que son capaces de proveer

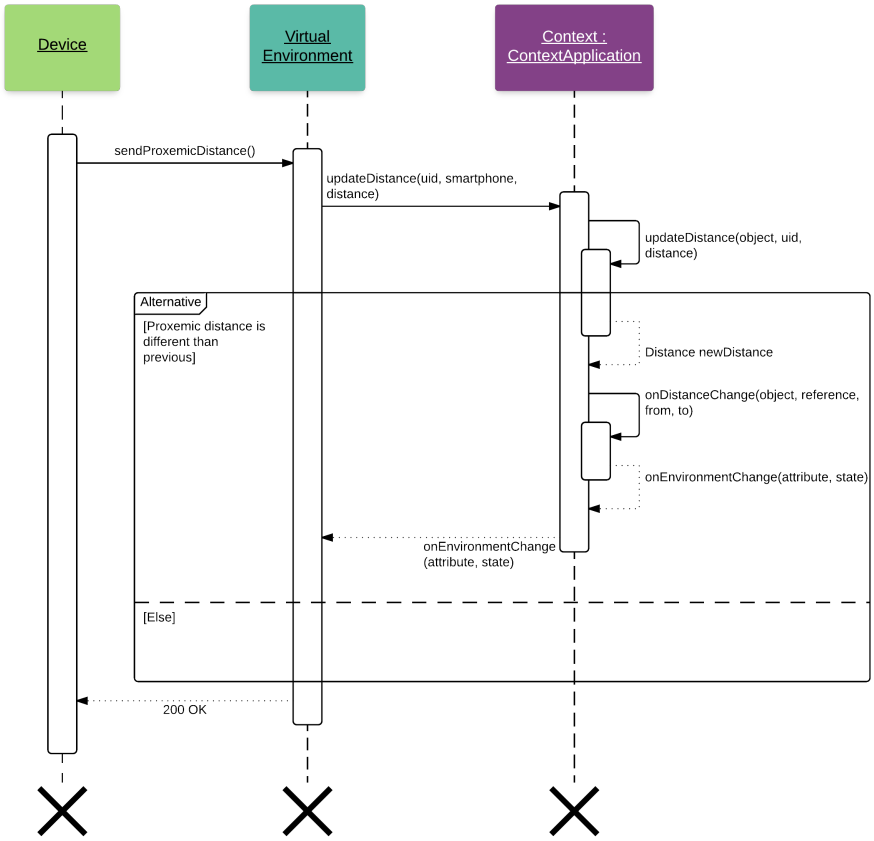

Figura 18**: Diagrama de secuencia cuando el Ambiente Virtual recibe un evento y es procesado por la aplicación contextual.**

al ambiente, tal como el control de la iluminación, su encendido o apagado, etc.

Para adherirse a los lineamentos de la plataforma, es necesario cumplir con los requerimientos de la misma. Para registrar el servicio por Multicast DNS se requiere colocar el dominio *"_openthings._tcp.local."* y proveer un puerto y un nombre, los cuales son libres de usarse. Se necesita implementar la clase Application del marco de trabajo Restlet e implementar la clase PushApplication de OpenThings, para que se suscriba de manera automática en cuanto sea encontrado HubServer a través de mDNS.

Una clase ejemplo desarrollada dentro del trabajo realizado para que sirva de ejemplo para el desarrollo de otro tipo de smart objects es *ComputerServer*. Existen otros ejemplos que demuestran el registro de smart objects pero suscritos bajo un tema o evento diferente.

Este elemento debe interpretar los comandos provenientes del HubServer con los

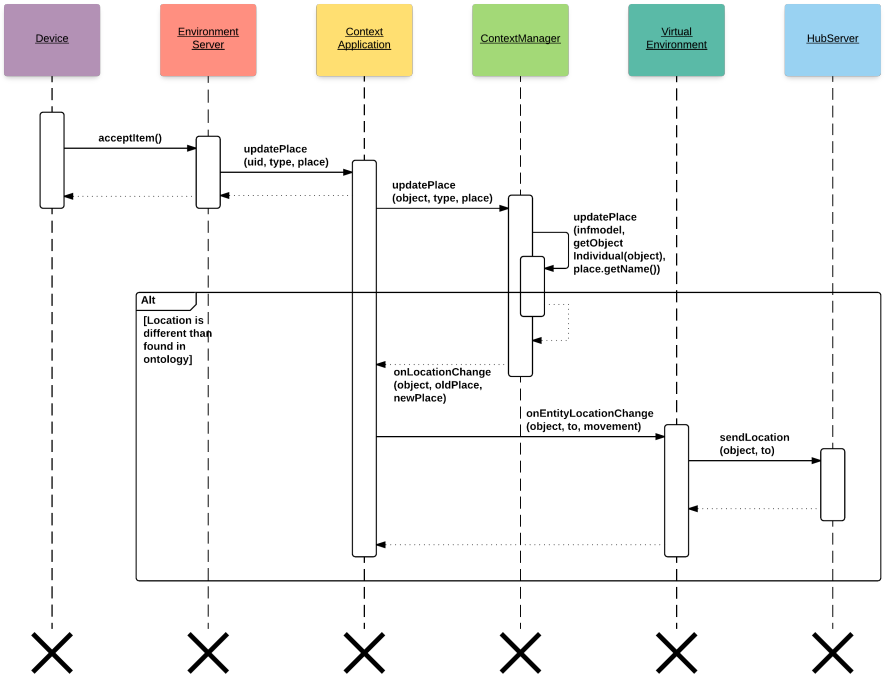

Figura 19**: Diagrama de secuencia que ejemplifica el flujo para la actualización de una ubicación.**

campos **title** y **content** del mensaje, de acuerdo a su función. Un ejemplo de un mensaje se puede observar en la figura 21, el cuál muestra la iluminación artificial debe estar encendida.

Este mensaje llega como un objeto JSON del cual el desarrollador toma la información que necesita el Smart Object para actuar de acuerdo a su función, de la forma en que la plataforma, y específicamente las personas, lo necesiten.

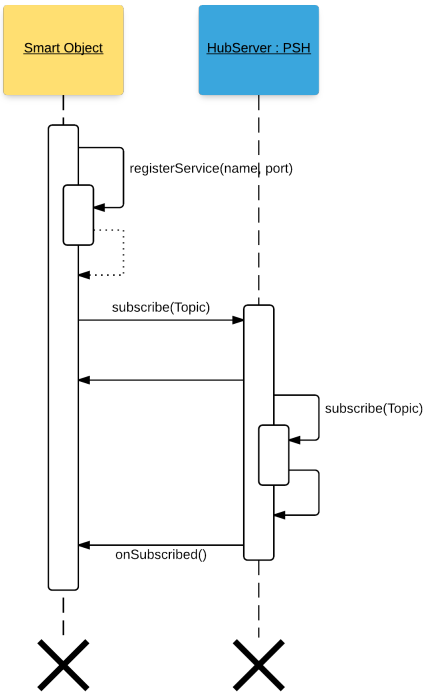

Figura 20**: Diagrama de secuencia para la suscripción de un Smart Object dentro de la plataforma OpenThings.**

```
{
    "title": "lightControl",
    "content": "on"
}
```

Figura 21**: Ejemplo de un mensaje para controlar la iluminación.**

Capítulo 6. Aplicación contextual y sus componentes

En este capítulo se describirán los detalles técnicos, así como describir los módulos de software desarrollados para el trabajo en cuestión. Se describirán detalles importantes sobre sus características, así como la prueba de concepto que se realizó como la forma de corroborar el funcionamiento de la integración de las dimensiones espaciales y temporales de acuerdo al contexto del ambiente, así como la sucesión de eventos que surgen en el tiempo y en el espacio.

Este capítulo se compone de las siguientes secciones:

- El componente TripleStoreServer. Aquí se describe a detalle el servicio virtual que se relaciona con el funcionamiento de los Beacons.
- El componente “Device”. Dentro de esta sección se detallará el rol que tiene un device dentro de la plataforma.
- Smart Objects. Se describe el funcionamiento y el registro de los smart objects, así como la clase desarrollada como plantilla para que los desarrolladores la empleen dentro de ellos.
- Beacons. Los beacons son aquellos componentes que sirven para representar objetos así como lugares físicos dentro de la plataforma. En esta sección se habla acerca de su implementación y su rol respecto al device.
- Ambiente Virtual. Es uno de los componentes de OpenThings usado para integrar el módulo de software que contiene el razonador semántico y la ontología. En esta sección se habla cómo los desarrolladores de aplicaciones para ambientes inteligentes deben utilizarlo.
- Mensajes. En la plataforma existe una comunicación de eventos, ya sea de cambios de distancias entre objetos, cambios de ubicación, así como otras dimensiones proxémicas. En esta sección se habla acerca de ellos y de los campos que contienen.

6.1 El componente TripleStoreServer

El servicio TripleStoreServer tiene la función de consultar información acerca de los Beacons en una Base de Conocimientos, dado su identificador único, con base en los principios de comunicación RESTful. Se expone el servicio por mDNS con el nombre de “TripleStoreServer”. Las aplicaciones deben ser capaces de comunicarse a éste para conocer la ubicación y el tipo de Beacon que se encuentra en el rango de alcance.

```
{
    "virtual-ambient-id": "UbiquitousComputingLaboratory",
    "place": "UbiquitousComputingLaboratory"
}
```

Figura 22**: Ejemplo del resultado de realizar una petición GET al servicio TripleStoreServer.**

Se trata de un servicio que cumple con esta función. Sin embargo, en futuras versiones de este trabajo se recomienda la implementación de mecanismos para guardar información semántica y actualizarla automáticamente, dejando la configuración de las entidades a la plataforma.

Cuando se hace una petición al servicio (El componente “Device”), éste se encarga de consultar una ontología de acuerdo al Beacon que tiene asociado el lugar.

En la figura 23 se observan las dos clases de la ontología: una que corresponde a la descripción del Beacon y cuyo nombre es representativo del lugar (UbiquitousComputingBeacon), y otra que corresponde al lugar que lo contiene (UbiquitousComputingLaboratory). Al realizar la consulta en la ontología, la aplicación busca información de los lugares que contienen el Beacon con el identificador “UCB5739”. El resultado de esta operación regresa a la entidad que hizo la petición se muestra en la figura 22.

También se espera que se tenga una ontología distribuida de acuerdo a los Beacons que puedan describirse a sí mismos por la plataforma. De esta manera, no toda la información estará concentrada dentro de su Base de Conocimientos, sino que tendrá que ser accedida usando los principios de Linked Data, tal como la plataforma OpenThings sugiere.

```
<!-- http://www.semanticweb.org/mc/ontologies/2014/9/locations#UbiquitousComputingBeacon -->

<owl:NamedIndividual rdf:about="&locations;UbiquitousComputingBeacon">
    <rdf:type rdf:resource="&locations;Beacon"/>
    <locations:hasFloor rdf:datatype="&rdfs;Literal">2</locations:hasFloor>
    <locations:hasIdentifier rdf:datatype="&xsd;string">B108</locations:hasIdentifier>
    <locations:containedIn rdf:resource="&locations;UbiquitousComputingLaboratory"/>
</owl:NamedIndividual>


<!-- http://www.semanticweb.org/mc/ontologies/2014/9/locations#UbiquitousComputingLaboratory -->

<owl:NamedIndividual rdf:about="&locations;UbiquitousComputingLaboratory">
    <rdf:type rdf:resource="&locations;Laboratory"/>
    <locations:hasIdentifier rdf:datatype="&rdfs;Literal">107</locations:hasIdentifier>
    <locations:hasFloor rdf:datatype="&rdfs;Literal">2</locations:hasFloor>
    <core:index rdf:datatype="&xsd;positiveInteger">3</core:index>
    <locations:containedIn rdf:resource="&locations;ComputerScienceDepartment"/>
    <locations:isContiguous rdf:resource="&locations;HCITestingRoom"/>
    <locations:nearBy rdf:resource="&locations;ImageProcessingComputingLaboratory"/>
    <locations:hasBeacon rdf:resource="&locations;UbiquitousComputingBeacon"/>
    <locations:frontOf rdf:resource="&locations;Workspace112"/>
    <locations:hasCorridor rdf:resource="&locations;Workspaces100Corridor"/>
</owl:NamedIndividual>
```

Figura 23**: Dos individuos en la ontología. Uno representa el Beacon con sus propiedades (como identificador único B108) y el otro el lugar (UbiquitousComputingLaboratory).**

6.2 El componente “Device”

Para este trabajo se pensó en un objeto inteligente que represente a un individuo y que interactúe virtualmente con la plataforma (llamado “device”) cumpliendo las siguientes tareas:

- Comunicación con plataforma (Ambiente Virtual).
- Ubicación en interiores.
- Identificación de usuario.
- Cercanía (diferencia de distancia) respecto a otros objetos.
- Proxy entre otros dispositivos para la medición de dimensiones proxémicas.

Para cumplir con ésto, se desarrolló una aplicación para dispositivos Android para que ofreciera esta funcionalidad y que además buscara congruencia con la plataforma OpenThings:

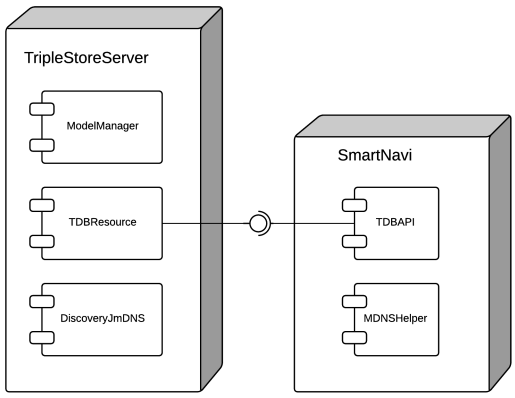

Figura 24**: Diagrama UML de componentes del servicio TripleStoreServer y la aplicación para Android SmartNavi.**

- Publicación del servicio por Multicast DNS.
- Conexión con Ambiente Virtual.
- Aviso de eventos importantes.
- Captura de mensajes provenientes de HubServer.

Para registrarse en *mDNS* es necesario utilizar las herramientas que ofrece Android para el descubrimiento y la publicación de servicios en la red. El SDK de Android ofrece la clase *NsdManager* con estas características. Se encontró que en el dispositivo ésta es la forma para su integración en la plataforma, ya que la librería utilizada en Java requiere dependencias nativas que no se encuentran en el dispositivo.

Para este caso se desarrolló una clase (MDNSHelper) que facilita la interacción dentro de la plataforma. Los desarrolladores pueden ajustarla a sus necesidades. Un ejemplo de uso de esta clase puede encontrarse ilustrado en la figura 26. Su función cumple con dos propósitos:

- Encontrar la entidad TripleStoreServer y consultar el nombre del Ambiente Virtual a partir del ID del Beacon asociado.
- Buscar el nombre de la aplicación contextual relacionada.

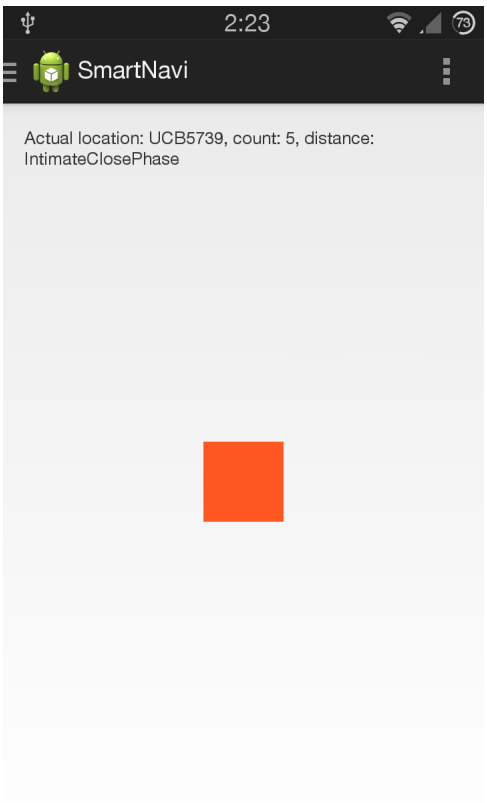

Figura 25**: Pantalla tomada de la aplicación SmartNavi.**

```
mdnsHelper = new MDNSHelper(this);
mdnsHelper.searchTripleStoreService();
```

Figura 26**: Ejemplo de uso de la clase MDNSHelper para buscar el servicio TripleStoreServer por mDNS.**

Cuando se ha encontrado el TripleStoreServer, se debe realizar una petición a través de un HTTP GET con el parámetro *beacon-id* con el ID del Beacon encontrado. Este proceso regresa el nombre del Ambiente Virtual en un objeto JSON en el campo *virtual-ambient*. Y con este nombre se puede proceder a conectarse con el Ambiente Virtual en la red, y hasta no haber establecido la conexión no se procederá a enviar ningún evento importante capturado por el device.

6.2.1 Servicio de Ubicación usando Bluetooth Low Energy

Para el proceso de ubicación y medición de distancias, se utiliza un servicio que usa las capacidades de Bluetooth Low Energy (BLE) con la API de Android. Ésta debe capturar los dispositivos BLE en la cercanía, y mediante un cálculo especializado para el dis-

positivo se establece la distancia aproximada entre ambos. Por tanto, sirve como método para obtener la ubicación de la persona en un edificio. También es utilizado para conocer la distancia entre la persona y un objeto inteligente.

El cálculo se hace mediante la captura de la intensidad de la señal (RSSI) del Beacon. Se utiliza una fórmula con una constante que varía con el smartphone, y es enviado al Ambiente Virtual una vez haya atravesado cierto umbral de distancia, de acuerdo a las distancias proxémicas de Hall (1990). En la figura 27 se muestra el método utilizado para calcular la distancia en Android.

```
/**
 * Obtains an approximation of a distance between this device with other.
 * This method uses the formula contained inside Android Beacon Library
 * (http://altbeacon.github.io/android-beacon-library/distance-calculations.html).
 *
 * @param device is the device to be calculated the distance with.
 * @return an approximation of the distance.
 */
public double getDeviceDistance(BLEDevice device){
    double avgDistanceValue;

    double A = 0.42093;
    double B = 6.9476;
    double C = BASE_DISTANCE;
    double t = -72;
    // Formula obtained from Android Beacon Library
    double actualDistanceValue = A * Math.pow(device.getRssi() / t, B) + C;
    avgDistanceValue = (actualDistanceValue + previousDistanceValue) / 2;
    previousDistanceValue = actualDistanceValue;

    return Double.parseDouble(decimalFormat.format(avgDistanceValue));
}
```

Figura 27**: Método para calcular la distancia entre dos objetos utilizando unos factores preestablecidos, propios de la antena Bluetooth con el que el device cueta.**

Sin embargo, se conoce que la intensidad de la señal y el cálculo de una distancia por este parámetro es inexacto y propenso a errores debido a obstáculos entre el emisor y el receptor. Para el propósito de este trabajo se probó en un ambiente controlado ya que sólo se pretende comprobar el funcionamiento del razonamiento, y no de las cuestiones de ubicación de forma precisa. De la misma manera, la búsqueda continua de Beacons por BLE consume energía y el tiempo de autonomía del celular disminuye. Para evitar este problema es necesario establecer nuevos mecanismos en los que el dispositivo se quede esperando mensajes de Beacons, de manera que sólo se active el proceso de búsqueda en momentos clave para maximizar el ahorro de energía.

6.3 Smart Objects

Los Smart Objects sirven como entidades con capacidades de procesamiento y almacenamiento de información que facilitan el control de aspectos del ambiente utilizando el control de pines digitales y analógicos. En el caso de este trabajo se utilizaron únicamente como controladores, sin embargo, es posible conectar una serie de sensores que permitan conocer factores del ambiente.

```
public class LightControl {

    private static LightControl instance;

    private final static int LIGHT_ON = 1;
    private final static int LIGHT_OFF = 2;

    private Mraa4j mraa;
    private GPIO gpio;

    public LightControl() {
        mraa = Mraa4j.getInstance();
        try {
            gpio = new GPIO(13);
            gpio.setDirection(GPIODirT.MRAA_GPIO_OUT);
        } catch (MraaException e) {
            e.printStackTrace();
        }
    }

    public void turnLightOn(){
        try {
            gpio.write(1);
        } catch (MraaException e) {
            e.printStackTrace();
        }
    }
```

Figura 28**: Ejemplo del control de pines digitales en Java con la librería Mraa4j para Intel Edison.**

Su rol de control depende de la plataforma en la que están desarrollados. Intel Edison ofrece herramientas para controlar sus pines digitales y analógicos con Node.js, una librería de JavaScript que ofrece una interfaz con librerías nativas de su sistema operativo. Para el caso de Java se debe utilizar un *wrapper* que permita utilizar estas librerías pero a un alto nivel.

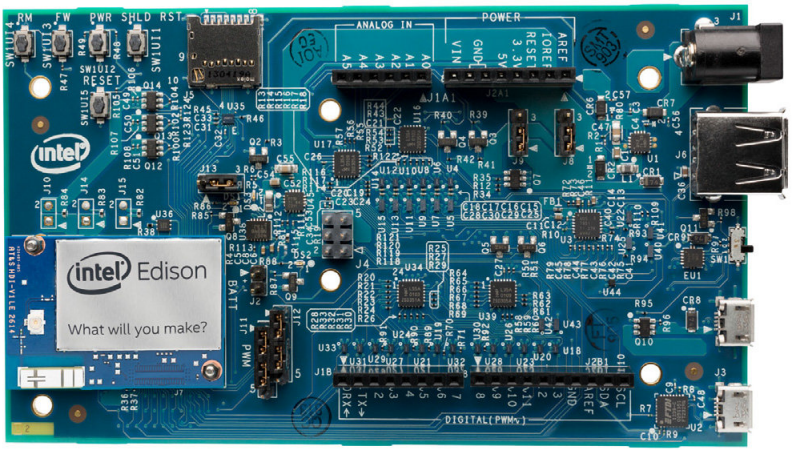

Figura 29**: Placa de desarrollo Intel Edison (2014).**

Intel Edison, como se muestra en la figura 29 es compatible con una gama amplia de electrónicos que van desde sensores hasta actuadores con motores y relevadores, que son controlados utilizando protocolos de comunicación serial o con señales digitales.

Para el caso demostrativo de este trabajo se controló el encendido y apagado del LED de la placa. Con esto demuestra el estado en el que se encuentra el control. Sin embargo, es importante denotar que se pueden conectar otro tipo de electrónicos, sensores y actuadores, para interactuar con el ambiente de acuerdo al rol que desempeñan.

Es importante que el dispositivo se registre en la red para una interacción instantánea y libre de configuración atendida para su instalación sencilla y una comunicación semántica. Su registro debe establecerse con un nombre arbitrario, pero exponiendo el tipo de Smart Object del cual se trata. En la figura 20 se muestra un diagrama que describe el flujo del registro.

De forma contraria a como lo hace SmartNavi, los Smart Objects deben suscribirse a los eventos de acuerdo al tema de interés para ellos. En el caso del control de la iluminación, se ha establecido que el nombre del Topic “Lights” sirva para la comunicación a los elementos cuyo interés reside en la iluminación. Ahora, si se publica un Topic con este título, solamente van a enterarse quienes estén registrados bajo este nombre. La comunicación se vuelve asíncrona, por lo que hay una mejora en el rendimiento energético y de carga en la red. La publicación del mismo se representa en la figura 20.

Como se mencionó anteriormente, cuando se envía una orden a HubServer con el

tópico “Lights” entonces esta entidad distribuye el mensaje a los suscriptores asociados. Si tomamos por ejemplo la figura 21, el título del mensaje es “lightControl” con el contenido “on”, entonces el Smart Object se debe encargar de interpretar el comando e interactuar acorde a su plataforma. El ejemplo de este caso se puede ver ilustrado en el código de la figura 28.

En el caso de este trabajo, fue utilizada la placa de desarrollo Intel Edison (29), y tuvo que emplearse una librería (Mraa4j), la cuál es una envoltura de una librería nativa para controlar los pines de entrad y salida digitales. En la figura 28 se muestra un ejemplo de una clase en Java para controlar los pines digitales.

6.4 Beacons

Figura 30**: Fotografía de un Beacon utilizando BLE Shield y la plataforma Arduino. Recuperado el 2 de noviembre del 2015 de http://redbearlab.com/getting-started-bleshield/.**

Los Beacons son dispositivos con procesamiento limitado, cuya función es emitir mensajes en la red que son capturados por otro dispositivo, con el motivo de ubicarse respecto a éste. En el sistema, éstos son empleados con un propósito: representar una entidad física. Sin embargo, su rol cambia de acuerdo a lo que están representando:

- Una ubicación.
- Un objeto o entidad física.

La diferencia entre ambos es que uno es empleado para disparar eventos de cambios de localización siendo que el otro se utiliza para conocer la posición en el ambiente, disparando cambios en la distancia y por lo tanto representar un cambio en la distancia proxémica. Para que exista un correcto funcionamiento, debe establecerse el tipo de Beacon del que se trata. Diferenciándolos es posible saber si los eventos que se dispararán con cambios en la distancia entre el device y el Beacon se tratan de cambios en la ubicación o en la distancia respecto a un dispositivo. Para demostrar su uso en la plataforma se implementó el primer tipo, para ubicar una persona respecto a la una sala.

Para el propósito especial de los Beacons, se ha implementado un servicio REST para obtener la información de éstos con facilidad, a través de una petición HTTP GET. Como se mencionó anteriormente, se envía dicha petición con el parámetro *beacon-id* y obtenemos como resultado un objeto JSON con la información del mismo. En la figura 22 se puede apreciar la respuesta de consultar este servicio con el ID del Beacon “UCB5739”.

En este caso, obtenemos como resultado el nombre del Ambiente Virtual asociado al Beacon, cuyo nombre es “UbiquitousComputingLaboratory”. El campo “virtual-ambient-id” contiene el identificador del lugar que es usado para publicar su servicio a través de mDNS. SmartNavi utiliza las librerías de Android para hacer la petición y obtener de esta manera el lugar del Beacon. A través de este proceso es como cualquier aplicación puede localizarse y saber a quién debe enviar los eventos de cambio de distancias y ubicación.

6.5 Ambiente Virtual

El Ambiente Virtual es una instancia especial de un servicio virtual, que representa un sitio físico o un grupo de servicios y aplicaciones que comparten un espacio físico de uso. Éste se encarga de recibir los eventos que surgen relacionados con el lugar físico que representan y de canalizarlos de acuerdo a la aplicación que tiene asociada.

Dicho Ambiente Virtual es representado dentro de la plataforma por la clase *VirtualEnvironment*. Ésta se encarga de establecer los parámetros para establecer la conexión en OpenThings y ofrecer una comunicación en una sola vía con HubServer. Para su correcto funcionamiento se debe proveer una serie de datos que identificarán al Ambiente

Virtual de los otros elementos de la plataforma. A continuación se enlistan los campos que deben ser enviados al Ambiente Virtual.

- Interfaz de red.
- Registro de eventos.
- Identificación.
- Aplicación contextual.
- Implementación de interfaces para la captura de eventos.
- Manejo de ontologías.

6.5.1 Interfaz de red

La librería para publicar el servicio por mDNS requiere que se provea una interfaz de comunicación, de manera que se utilice el protocolo IPv4 (en el caso del trabajo así se eligió debido a que no existe un soporte adecuado todavía por la librería para direcciones IPv6). Por tanto, si se trabaja con Linux, ya sea en el servidor o con la plataforma Intel Edison, es importante ajustar este parámetro.

Se ha comprobado el funcionamiento de éste dentro de una computadora con Linux, usando una distribución basada en Ubuntu 14.04, elementary OS 0.3.2, y en OpenSUSE 13.2. En el caso de OpenSUSE se necesitó hacer unos cambios internos a la configuración de los servicios para poder arrancar el servicio de Avahi, el cual implementa el protocolo propuesto inicialmente por Apple (Bonjour), y posteriormente hecho libre y delegado a Zeroconf. Se configuraron las interfaces dentro del Ambiente Virtual y finalmente se pudo encontrar el servicio a través de mDNS.

En el caso particular del ambiente de desarrollo en OpenSUSE se configuró la interfaz como *p3p2*, no se utilizó la tradicional *eth0* configurada automáticamente por Linux.

Se configura mediante el uso de la llamada a la clase padre *super("eth0")*, con el nombre de la interfaz correspondiente al que tiene acceso a la red.

6.5.2 Registro de eventos

En primer lugar se debe registrar el Ambiente Virtual para que se notifique sobre los eventos importantes en el ambiente, ya sea el registro con HubServer, el cambio de ubicación de un usuario, así como algún otro cambio dentro del mismo. Esto permitirá al desarrollador ser avisado cuando el ambiente se encuentra disponible en la red y registrado de forma correcta con HubServer, para que posteriormente se envíen las órdenes necesarias cuando suceda algún evento de importancia.

Se utiliza llamando el método *registerForCallback(this)*, encontrado dentro de VirtualEnvironment.

6.5.3 Identificación

El nombre del ambiente virtual o su identificador único puede ser dado por el desarrollador de forma libre, no existe ninguna limitante en cuanto a su formato. Sin embargo, para ser legible y comprensible por otras personas se recomienda tener un nombre que sea fácil de reconocer a las personas y que a su vez tenga relación con la información contenida dentro de la ontología de TripleStoreServer. El puerto puede ser configurado, pero no es obligatorio a menos que el desarrollador así lo requiera.

Hay que recordar que el nombre debe representar un lugar físico también en la ontología Places. Si no existe una correlación entre el lugar al que el Beacon tiene registrado en sus propiedades (dentro de TripleStoreServer), y tampoco existe la misma relación en la ontología no es posible crear un vínculo adecuado, y por ende tampoco la comunicación inmediata de acuerdo a las prácticas que se proponen en esta clase.

La idea es que con mecanismos más poderosos sea posible integrar su identificador desde un sólo punto (el desarrollador de la aplicación debe encargarse de administrar dicho nombre) y que actualice a los diferentes elementos que son parte del sistema. Un administrador no requiere cuidar el uso de los identificadores, ya que la plataforma misma se encargaría de su manutención.

Para registrar la información es necesario llamar al método *setEnvironmentInfo(name, port)* con una clase que hereda de *VirtualEnvironment*.

6.5.4 Aplicación Contextual

La aplicación contextual permite mantener comunicación con la ontología y el razonador semántico, para actualizar y consultar información contenida en la Base de Conocimientos. Es la clase mediante la cual se puede obtener un acceso a la ontología. Utiliza la librería de Jena (The Apache Foundation, 2014), su API para el manejo de ontologías y el uso de un razonador basado en reglas, *GenericRuleReasoner*.

La aplicación contextual debe ser capaz de actualizar, consultar y avisar de eventos que suceden en el plano espacio temporal. Los eventos son llamados por las interfaces *VirtualEnvironmentInterface* y *VirtualEnvironmentEvents*, las cuales deben ser implementadas en la clase del ambiente virtual para capturar dichos eventos.

Para registrar la aplicación con el ambiente virtual se debe llamar el método *getApplication().setContextApplication(ContextApplication.getInstance())*, en la cuál debe ser pasada como argumento una instancia de alguna aplicación contextual.

6.5.5 Implementación de interfaces para la captura eventos

Los eventos que ocurren de acuerdo a las dimensiones proxémicas, son:

- **onEnvironmentChange**. Evento que indica algun cambio en el ambiente.
- **onEntityLocationChange**. Evento que sucede cuando existe algún cambio en la ubicación de una entidad.
- **onFacing**. Evento disparado cuando hay un cambio en la orientación de una entidad respecto a su ambiente.

Y los eventos que se disparan de acuerdo a los eventos que suceden a nivel de la plataforma son:

- **onEnvironmentRegistration**. Cuando se registra de forma correcta a HubServer.
- **onMessageReceived**. Cuando se recibe un mensaje de HubServer.

Cada uno de ellos deben sobrecargarse dentro del ambiente virtual. El desarrollador de la aplicación tiene que capturar y considerar los eventos que surjan para que tenga su comportamiento proactivo. Los últimos eventos sirven para conocer el momento cuándo es adecuado enviar comandos en la red, y qué hacer en caso de que suceda algún inconveniente. Es a consideración de los desarrolladores ejecutar acciones previas a la conexión con HubServer, así como manejar en qué momento es adecuado establecer la suscripción al servicio.

6.5.6 Manejo de ontologías

Dentro de la aplicación contextual debe considerarse el uso del manejador de ontologías, para actualizar y consultar los individuos que representan un objeto o entidad física en la plataforma. En la clase ejemplo, **ContextApplication**, hay un manejo interno de las dimensiones proxémicas (distancia, identidad, ubicación, orientación).

Cuando se recibe una nueva ubicación a través de la aplicación SmartNavi, el ambiente virtual, en conjunto con las clases que ayudan a modificar la ontología (*ContextApplication, ContextManager*), realizan el proceso de actualización. En caso de que haya surgido algún cambio en la ontología (es decir, un evento de cambio de ubicación), la clase *ContextManager* avisa a *ContextApplication* para indicarle qué tipo de evento se llevó a cabo, así como el objeto que cambió de lugar. Esto permite a los desarrolladores enviar los comandos pertinentes de acuerdo a la aplicación.

Una vez que el sistema ha terminado de avisar a *ContextApplication*, envía el evento *onEntityLocationChange* para indicar al ambiente virtual que ha sucedido algún cambio dentro del ambiente, que corresponde al cambio de ubicación de un individuo o entidad, como puede ser la entrada o la salida de una persona de la habitación que representa el ambiente virtual. Finalmente, se envían los comandos adecuados a HubServer para que se adapte a las preferencias del usuario o las que se tengan por omisión para dicho ambiente.

6.6 Mensajes

Los mensajes que son enviados dentro de la plataforma son parte de una API REST desarrollada para enviar los parámetros importantes de las dimensiones proxémicas que se miden en una aplicación.

Se muestran tres tipos de mensajes diferentes:

- Ubicación.
- Distancias proxémicas.
- Orientación.

6.6.1 Ubicación

```
{
    "device-name": "mc-galaxy-alpha",
    "uid": "mc-galaxy-alpha",
    "type": "smartphone",
    "beacon-id": "UCB5970"
}
```

Figura 31**: Ejemplo de un mensaje en formato JSON para actualizar la ubicación.**

Los mensajes de ubicación sirven para identificar y avisar al ambiente virtual que corresponde a cierto Beacon sobre los eventos importantes que suceden en éste, ya sea la entrada o la salida de un lugar. Es el disparador para el ambiente virtual que indica que se debe hacer algún cambio en el ambiente de acuerdo a las preferencias de la persona o las que tiene por omisión, en caso de esté ausente.

A continuación se indican los campos del ejemplo de la figura 31 que contiene el mensaje en JSON y su correspondiente descripción:

- **device_name**. Describe el nombre del dispositivo, útil para relacionarlo con la persona a la que le pertenece.

- **uid**. Es el identificador único del dispositivo.
- **type**. Es el tipo de dispositivo del que se trata.
- **beacon-id**. Es el identificador único del Beacon que se usa como referencia. Útil para conocer la ubicación física de la persona.

6.6.2 Distancias proxémicas

```
{
    "message-type": "proxemic",
    "device-name": "mc-galaxy-alpha",
    "uid": "mc-galaxy-alpha",
    "type": "smartphone",
    "beacon-id": "UCB5970",
    "proxemic-distance": "IntimateClosePhase",
    "proxemic-distance-value": 100
}
```

Figura 32**: Ejemplo de un mensaje para un cambio de distancia proxémica.**

Las distancias proxémicas, debido a que están bien establecidas por los trabajos de Hall (1990), pueden implementarse desde un dispositivo (en este caso, la aplicación SmartNavi), y son enviadas junto con la distancia medida entre el dispositivo respecto a un Beacon de referencia, el cuál es representante de un objeto de la cercanía. Un ejemplo del uso de este mensaje se puede observar en la figura 32.

A continuación se describen los campos que se utilizan para describir una distancia proxémica que se envía al ambiente virtual.

- **message-type**. Para identificar el tipo de mensaje del que se trata.
- **device_name**. Indica el nombre del dispositivo.
- **uid**. Es el identificador único del dispositivo.
- **type**. Identifica el tipo del dispositivo del que se trata.
- **beacon-id**. Es el identificador único del Beacon que se usa como referencia.

- **proxemic-distance**. Es la distancia proxémica con su fase, de acuerdo a la distancia medida del dispositivo.
- **proxemic-distance-value**. Es el valor medido del dispositivo respecto al Beacon.

6.6.3 Orientación

```
{
    "message-type": "device",
    "uid": "mc-galaxy-alpha",
    "beacon-id": "UCB5970",
    "x": -44,
    "y": 100,
    "z": -221
}
```

Figura 33**: Ejemplo de un mensaje para un cambio de orientación.**

Los cambios de orientación son útiles para conocer la postura de la persona respecto a un objeto de la cercanía. De esta manera se puede inferir la actividad que está realizando. Permite conocer el significado de la distancia. Un ejemplo de este mensaje se puede observar en la figura 33.

A continuación se indican los campos del mensaje de orientación que es enviado al ambiente virtual.

- **message-type**. Es el tipo de mensaje del que se trata.
- **uid**. Es el identificador único del dispositivo.
- **beacon-id**. Es el Beacon que se usa como referencia para la orientación.
- **x**. Representa el plano x medido por el magnetómetro del dispositivo.
- **y**. Representa el plano y.
- **z**. Representa el plano z.

Capítulo 7. Implementación y Resultados

Dentro de este capítulo se describirá la implementación que se ha efectuado para probar el trabajo desarrollado en la presente tesis y su posterior evaluación. Asimismo se revisará a detalle una prueba de concepto y los resultados correspondientes a ésta.

7.1 Demostración de uso de la Aplicación Contextual (ContextApplication) con OpenThings

Lo que se busca con el siguiente ejemplo es ilustrar cómo funcionaría una plataforma con tecnologías semánticas y conciencia temporal y espacial en un caso real. Todas las situaciones se llevaron a cabo en un ambiente controlado para probar el razonamiento y su comportamiento proactivo. Se pretende mostrar además que las dimensiones temporal y espacial son necesarias para obtener contexto de la actividad de las personas. Además, se busca mostrar que las tecnologías semánticas ofrecen los mecanismos para representar ambas dimensiones y obtener conclusiones a partir de la configuración física y el contexto temporal, lo cual facilita a los desarrolladores la creación de aplicaciones con comportamiento proactivo.

La prueba de concepto describe un día de trabajo de un arquitecto al entrar en un edificio donde se encuentra su despacho. A continuación se describe cómo va sucediendo:

Un arquitecto tiene un despacho el cual cuenta con un sistema con objetos inteligentes que utilizan una plataforma del Internet de las Cosas, y está revestido por semántica. Para situaciones particulares tiene preferencias de uso en su ambiente de trabajo: cuando entra en su despacho, él desea que se adecue lo mejor posible para utilizar la iluminación natural, por lo que requiere que se abran las cortinas, se apague la iluminación artificial, y se encienda su computadora, pero para ahorrar energía, desea que la pantalla permanezca apagada mientras no se encuentre mirándola.

El arquitecto entra a su oficina al comienzo de un día laboral, y el sistema reconoce su identidad en cuanto pasa por la puerta. De esta manera el sistema considera el evento que atravesó la puerta y entró en el lugar para hacer que el sistema se adecue de acuerdo

a sus necesidades y preferencias. La iluminación artificial se apaga, y permite la entrada de luz natural subiendo las cortinas.

Se acerca a su escritorio para revisar las tareas de hoy, pero en vez de permanecer cerca del monitor de su computadora, dirige su atención hacia una pantalla gigante en la que tiene puesto los planos que está manejando de una obra. La pantalla de la computadora sigue sin encenderse, y el arquitecto comienza a interactuar con la pantalla gigante manteniéndose a una distancia cercana al monitor de su computadora.

Es hasta que se sienta en su silla y cambia su orientación hacia la pantalla de la computadora cuando se enciende, y abre una aplicación de correos, ya que tiene algunos que necesita revisar para comenzar con los pendientes del día de hoy.

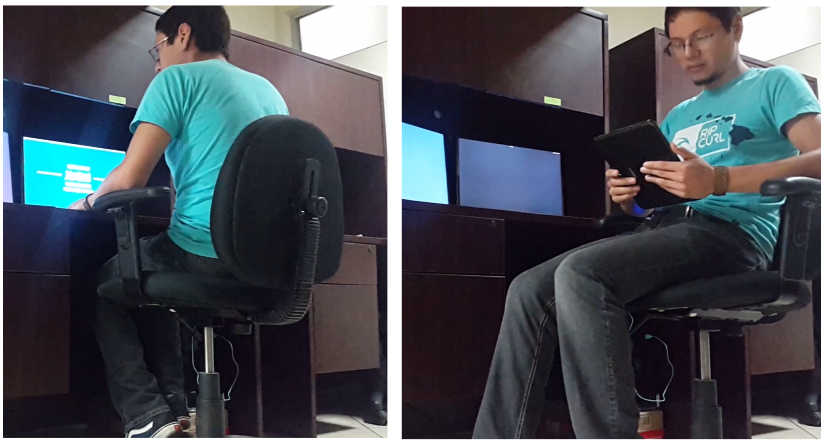

Figura 34**: Dispositivo interactuando encendiendo o apagando la pantalla ante la posición del usuario.**

7.2 Análisis de la prueba de concepto

Después de describir cómo es que el sistema actuó para adaptarse a las necesidades y requerimientos del usuario, es importante ver cuál fue el procedimiento que se llevó a cabo para razonar sobre el contexto del ambiente. Para ello se utilizan diagramas para mostrar gráficamente la implementación en la plataforma.

Para mantener organizada esta sección se mencionan los puntos importantes que ocurrieron en el momento en que cada evento sucedió. A continuación se muestran dichos aspectos de la prueba de concepto.

- **Comportamiento del sistema**. La pieza de software tiene la cualidad de comportarse de forma reactiva a los eventos que suceden en el ambiente, como el cambio de distancias entre los objetos que le componen así como el inicio y el final de ciertos eventos que son considerados importantes y cuyos requerimientos de programación piden explícitamente un cambio de factores del ambiente.

- **Aspectos temporales**. El tiempo es una dimensión importante del sistema para poder ofrecer un contexto mucho más sofisticado y completo que la cercanía entre los objetos inteligentes. Permite conocer la actividad que pueden estar realizando los usuarios dada la hora del día. En la prueba de concepto interviene un proceso que considera el inicio y el final de los eventos que pueden ocurrir dentro de un ambiente. El evento de importancia ha sido previamente creado por el mismo usuario en el momento en que se registró para usar su lugar de trabajo. El procedimiento que lleva a cabo consta del flujo que se demuestra en el diagrama 35. Lo que se realiza es básicamente una consulta en SPARQL al modelo que contiene la ontología *Time Events*, para conocer qué evento está sucediendo o está por suceder. En caso de que esté ocurriendo un evento, la aplicación debe tener consciencia de las características del mismo, por ejemplo el usuario al que se le debe considerar como prioritario, la hora del día, la clase de actividad que se está llevando a cabo, etc.

- **Distancia proxémica**. La conjunción del tiempo con la distancia de las entidades (objetos o personas en un ambiente) brinda un nivel contextual mucho mayor, ya que permite saber la ubicación de la persona con respecto a otros objetos, y cuya distancia tiene un significado según la proxémica (Hall, 1990). Por lo tanto, la cercanía del dispositivo de la persona con el escritorio donde está su computadora indica que se encuentra *a)* dentro del ambiente y *b)* que está en el lugar determinado para trabajar. Esto le permite a la aplicación saber que el usuario tiene la intención de utilizar su computadora, y envía órdenes para adecuarse a sus preferencias.

- **Identidad**. La identidad es un factor esencial para conocer información sobre la persona. En la ontología dentro de la aplicación se encuentra la descripción de la persona, así como sus preferencias para estos casos particulares y su rol dentro de la institución. Le permite tener prioridad de acuerdo con su perfil de usuario, asig-

nado previamente por un administrador. El smartphone es el dispositivo empleado para conocer su identidad. Al entrar dentro del rango de un dispositivo BLE que representa el espacio de trabajo, el dispositivo envía un mensaje para identificarse y avisar que ha llegado a cierto lugar. La conjunción entre identidad y distancia permite al sistema conocer la ubicación relativa del mismo.

- **Ubicación**. Sumado a lo anterior, la ubicación es muy importante para complementar el contexto que le rodea. De esta manera se puede conocer la actividad que está realizando, y por tanto, adecuarse a sus necesidades, dependiendo de su perfil y de los permisos que el usuario tenga. Así, se conoce dónde se encuentra dentro del ambiente inteligente y con ello actuar de forma acorde a sus necesidades.

- **Orientación**. Un aspecto importante para reconocer la actividad se obtiene de la orientación de su cuerpo respecto a los objetos de su ambiente. En el caso mencionado, si la persona se encuentra cerca de la pantalla, y además la está mirando, significa que muy probablemente desea trabajar. En versiones futuras se recomienda integrar un reconocimiento de actividad con otros dispositivos que puedan obtener esta información de forma más precisa y sencilla.

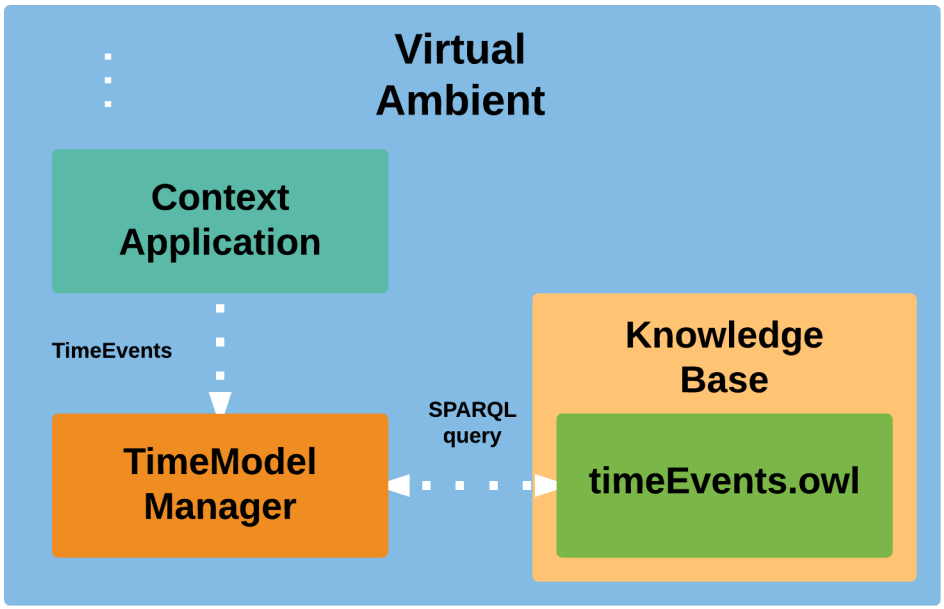

Figura 35**: Interacción de aplicación con Base de Conocimientos.**

7.3 Roles de la prueba de concepto

A continuación se mencionan los roles principales de los dispositivos dentro de la implementación de la prueba de concepto, junto con las tecnologías utilizadas.

- **Dispositivo de identificación inalámbrico**. Un dispositivo con sistema operativo *Android* fue empleado como identificador y, como se mencionó anteriormente, para aproximar la cercanía de una persona dentro de un ambiente inteligente.
- **Representación de ubicación**. Para la ubicación fue empleada la placa de prototipado rápido *Arduino* con un añadido para comunicación *Bluetooth Low Energy* (RedBearLab, 2015). Fueron utilizados para representar un objeto, además de utilizarse como los representantes de un ambiente inteligente.
- **Pieza de software con razonamiento**. Esta pieza de software es la entidad que contiene una *Base de Conocimientos*, en donde está almacenada la información sobre la plataforma, como el dispositivo Android, el ambiente virtual, los Beacons Bluetooth. Es la encargada de hacer las inferencias para tomar como entrada la ubicación de una persona y la distancia con respecto a los objetos de la cercanía.
- **Objetos inteligentes**. Los smart objects permiten representar algún objeto del ambiente, requiera o no de procesamiento computacional. En la implementación fueron utilizados la placa de prototipado *Arduino* e *Intel Edison*, los cuales representan tanto un ambiente inteligente, como un objeto, así como un servicio que puede interactuar con el ambiente para adaptarlo de acuerdo a las necesidades de la persona.

7.4 Módulos empleados de la plataforma OpenThings

En esta sección se describen las piezas de software utilizadas para concordar con los lineamientos de OpenThings e integrar nuestro trabajo en una aplicación, con el fin de alcanzar los objetivos indicados.

- **HubServer**. Este elemento implementa el mecanismo *PubSubHubbub* basado en un modelo de suscriptor - publicador (Network Working Group, 2014). En éste servicio, publicado a través de Multicast DNS, se lleva a cabo todo el proceso de registro de una suscripción, el manejo y enrutamiento de los mensajes provenientes del publicador.

- **PushRouter**. Esta clase que hereda de *Router* del marco de trabajo *Restlet* se encuentran los métodos para facilitar la conexión con HubServer. También utiliza la interfaz *PushApplication* que requiere cualquier aplicación que utiliza *PushRouter* como el Router que necesita la aplicación RESTful para utilizar recursos personalizados. Ofrece la conexión con HubServer, y por tanto, las aplicaciones que se conectan con un Topic son avisados una vez que llega un mensaje publicado con ese Topic.

- **Multicast DNS**. Uno de los requerimientos de la plataforma OpenThings es la publicación de los dispositivos a través de mDNS. Se utiliza la clase *DiscoveryJmDNS*, tanto para registrar el servicio como para habilitar la búsqueda de servicios en la red a través de este mecanismo.

- **Restlet Framework**. Este framework permite la comunicación bidireccional usando el protocolo HTTP de la suite de Internet. Es un componente básico para la plataforma OpenThings para que pueda mantener una interacción con los demás elementos de software del sistema en un ambiente puramente distribuido.

- **Modelo de Ambiente Virtual**. El modelo propuesto por Delgadillo-Rodríguez (2015) contiene cuatro elementos principales: servicios, servicios virtuales, ambientes virtuales y aplicaciones. Dichas categorías componen las entidades de la plataforma, las cuales se encargan de encapsular servicios a más alto nivel para que desarrolladores y usuarios puedan utilizarlos sin inconvenientes, y de una forma estándar adecuada a la Web Semántica. Por esto mismo, es importante representar un lugar y asociarlo a servicios que le ayuden a distribuir la información acorde a los requerimientos propuestos, con sus prácticas recomendadas.

7.5 Dispositivo proxémico

Para que se demostrara la facilidad de integrar un dispositivo nuevo dentro de la plataforma, se optó por desarrollar un proyecto en colaboración con Angello Hoyos para medir la dimensión de la orientación e interpretar el significado de los datos provenientes de dicho dispositivo. Gracias a esta colaboración se obtuvo el dispositivo proxémico (figura 36).

El dispositivo proxémico es una banda que se usa a la altura de la cintura y contiene: un magnetómetro y acelerómetro para medir la orientación que tiene la persona respecto a su ambiente; una antena Bluetooth Low Energy, basada en tecnología para cómputo vestible Flora, y un microcontrolador que se encarga del procesamiento de los datos y del control de información para ser enviado a un dispositivo Android (la aplicación Smart-Navi), el cuál se encarga de enviar la información semántica a la aplicación del ambiente inteligente y posteriormente a la Base de Conocimientos.

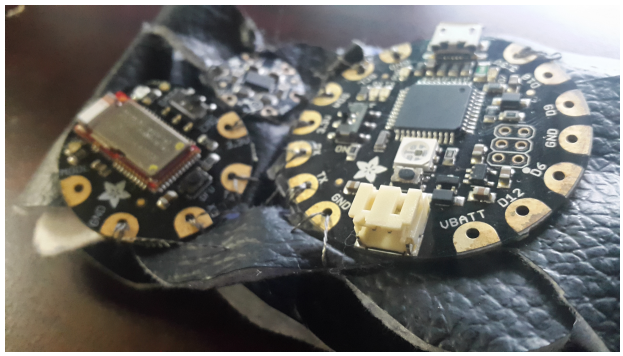

Figura 36**: Dispositivo proxémico con acelerómetro, magnetómetro, comunicación por BLE y usando la plataforma FLORA (https://www.adafruit.com/products/659).**

La función de este dispositivo es determinar la posición de la persona con respecto a su ambiente, utilizando un acelerómetro y un magnetómetro. El dispositivo envía la información cuando es requerida (la aplicación debe estar enviando mensajes pidiendo la información), de forma que es posible configurar el tiempo entre cada dato, y por lo tanto puede establecerse un balance en la precisión y en su tiempo de vida.

Para categorizar los datos dentro de una ontología, se procedió a modificar el código que captura la información dentro del Ambiente Virtual.

Previamente se ha establecido la regla proxémica para que el razonador semántico

clasifique tres parámetros, que establece que una persona se encuentra volteando hacia un lugar en particular. Sin embargo, no se puede categorizar únicamente con su posición. Ya que las dimensiones proxémicas propuestas por Greenberg *et al.* (2011) no pueden separarse las unas de las otras, ya que cada una ofrece un pedazo de información importante para conocer el contexto en el que una persona se encuentra, se tiene que considerar la orientación del individuo y además la cercanía que existe con otros objetos del ambiente.

En la prueba de concepto descrita anteriormente, el proceso que sucede en el Ambiente Virtual y en la aplicación SmartNavi se describe a continuación:

1. El individuo entra en su oficina. Se obtiene la cercanía con el Beacon BLE que representa el lugar físico.
2. Se acerca a su escritorio. La distancia proxémica es captada y categorizada en la zona íntima.
3. La orientación que tiene el usuario desde su silla se identifica como que no está volteando hacia él.
4. La orientación vuelve a cambiar hacia el escritorio. El evento de que ha regresado su vista hacia éste se dispara. El sistema envía los comandos para encender el monitor de la computadora.

7.6 Resultados obtenidos a partir de la prueba de concepto

En esta sección se muestran los resultados obtenidos a partir de la prueba de concepto, su desarrollo e implementación en la plataforma elegida del Internet de las Cosas.

Con la evaluación del sistema con la prueba de concepto se demostró que es factible utilizar la plataforma con consciencia temporal y espacial para responder a los eventos generados por cambios de distancias y del tiempo. La creación de aplicaciones con consciencia de ambas dimensiones se torna más sencilla para los nuevos desarrolladores. Se facilita al desarrollador el uso de la plataforma, así como de la característica del razonamiento con plantillas previamente hechas a las que solamente tiene qué configurar

para distinguirla de otras aplicaciones y enfocarse en el comportamiento. Sin embargo, es necesario un conocimiento previo del lenguaje de programación Java, servicios Web RESTful y nociones básicas de OpenThings.

En el desarrollo del trabajo, fue necesario construir un conjunto de entregables. A continuación se listan las contribuciones a partir de los objetivos planteados en la propuesta de la tesis.

- **Tabla comparativa con trabajos relacionados**.
- **Ontologías**.
- **Manejador de Base de Conocimientos**.
- **Implementación de un razonador semántico**.
- **Fundamentos para crear nuevas aplicaciones**.
- **Técnicas propuestas para implementar razonadores semánticos**.

7.6.1 Tabla comparativa con trabajos relacionados

A continuación se muestra una tabla comparativa respecto a otros trabajos encontrados en una investigación profunda en la literatura relacionada. Se toman en cuenta algunas características, con las que cuentan o no otros trabajos. Los puntos a comparar son:

- **Uso de tecnologías semánticas.** Indica si se manejan estándares de tecnologías semánticas en su trabajo.
- **Conciencia temporal.** Indica si fue utilizada la dimensión temporal como un punto importante para conocer la actividad de un individuo.
- **Conciencia espacial.** Indica si se utilizó la dimensión espacial para conocer el contexto de un individuo con su ambiente.

- **Razonamiento semántico.** Si se utilizó o no algún tipo de razonamiento semántico (y no alguna otra herramienta fuera de las tecnologías semánticas).
- **Proactividad.** Si ofrece un comportamiento reactivo a la presencia de las personas para ayudarlos en sus actividades.
- **Interacción con los humanos.** Si permite algún tipo de interacción con los humanos.
- **Modelo semántico.** Si utiliza ontologías para describir y relacionar conceptos.
- **Heterogeneidad.** Si permite que distintos componentes, de diferentes tecnologías, interactúen entre sí.
- **Interoperabilidad.** Si permite una comunicación comprensible para los elementos de la plataforma.
- **Escalabilidad.** Si se permite un crecimiento para la plataforma.
- **Seguridad.** Si ofrece aspectos de la seguridad en la información.

Tabla 2**: Tabla que muestra las características para los trabajos relacionados revisados.**

Características	Genio	DomoML	MOCOCO	Proximity Toolkit	AwareMedia	SENSORD	ContextApplication
Uso de tecnologías semánticas	X	O	O	X	X	X	X
Conciencia temporal	X	X	X	O	O	O	X
Conciencia espacial	O	X	X	X	X	O	X
Razonamiento	X	O	O	X	X	X	X
Proactividad	O	O	O	O	O	O	X
Interacción	O	O	O	O	O	X	X
Modelo semántico	X	O	O	X	X	X	X
Heterogeneidad	O	-	X	O	O	X	O
Interoperabilidad	X	O	O	X	O	O	O
Escalabilidad	X	O	-	X	O	-	X
Seguridad	X	-	X	X	X	X	O

X = Cumple; O = No cumple; - = Cumple parcialmente

En esta comparativa se encontraron las siguientes ventajas respecto a los trabajos encontrados en la literatura:

- Se encontró que soporta, además de conciencia temporal y espacial, las tecnologías semánticas para representar ambas dimensiones.
- Tiene un razonador semántico que le otorga proactividad e interacción con los individuos, ya que OpenThings ofrece estas características.
- Describe la información en términos de un modelo semántico, el cual utiliza para su funcionamiento. Esto lo hace independiente de la tecnología de razonamiento, del estándar elegido para representar la ontología, y de los objetos inteligentes que estén conectados usando la plataforma.

Por otro lado, las desventajas que se hallaron en comparación con los trabajos de la literatura fueron:

- No se contempló la heterogeneidad entre razonadores y entidades de otra clase de

tecnologías semánticas. Permite solamente el uso exclusivo de un razonador en conjunto con una ontología. Actualmente no soporta ontologías fuera del Ambiente Virtual en el que se encuentran.

- No hay interoperabilidad con otras plataformas. Es compatible únicamente con OpenThings.
- No ofrece seguridad en la información, debido a que el enfoque de este trabajo fue exclusivamente la implementación de las dimensiones en la plataforma, y el soporte del razonamiento semántico.

7.6.2 Ontologías

La ontología espacial propuesta, con base en las ubicaciones presentes en el edificio del departamento de Ciencias de la Computación en CICESE, y junto con las dimensiones proxémicas y un modelo general de preferencias de usuarios, conforman parte del trabajo necesario para poder representar el espacio en términos adecuados para representar su significado.

Se desarrolló la ontología **Places** utilizando como base el dominio general que se pudo observar en el edificio. La descripción de dicha ontología se encuentra dentro del capítulo 4. Asimismo se creó la ontología **Proxemics** que contiene la descripción de las preferencias de usuarios y su privacidad. Todo esto se engloba dentro de la ontología "context-ontology" que viene incluida con el código fuente del proyecto. Se puede encontrar un diagrama en el apéndice 9.2, así como su descripción en el capítulo Ontología.

Dicha ontología fue elaborada debido a que no se encontró alguna otra que tuviera una definición similar para representar el espacio. De esta manera, el uso de la ontología facilitó la creación posterior de elementos y dispositivos dentro de la Base de Conocimientos para poder representar el espacio, los lugares, y los objetos. En el caso del tiempo se empleó la ontología timeEvents, basada en la ontologías del tiempo propuesta por la W3C (Jerry H. Hobs, 2006). Con ella se representaron los eventos importantes que se debieron considerar en la prueba de concepto propuesta y en la aplicación implementada.

7.6.3 Manejador de Base de Conocimientos

Esta pieza de software programada en Java funciona como la interfaz para interactuar con una ontología hecha usando la API de Jena (The Apache Foundation, 2014). Permite la actualización, consulta y eliminación de propiedades e individuos dentro de la misma, sin emplear *SPARQL*. Esto es debido a que la implementación de *SPARQL 1.1* en la API permite únicamente realizar consultas a una ontología, por lo que se encuentra limitada para los propósitos de este trabajo.

La documentación pertinente a los métodos que se usan se encuentra en el mismo código, para que los desarrolladores que deseen realizar operaciones más detalladas sobre los individuos y sus propiedades, con previos conocimientos de la web semántica, puedan hacerlo fácilmente.

7.6.4 Implementación de un razonador semántico

Además del desarrollo de la ontología espacial “context-ontology” y el manejador de la Base de Conocimientos, se introdujo un razonador semántico de la API que ofrece Jena, *GenericRuleReasoner*, de forma que se toman en consideración algunas reglas básicas para categorizar las distancias proxémicas y otros conceptos que tienen que ver con la actualización de los individuos en la Base de Conocimientos.

Este objetivo fue alcanzado gracias a que previamente se desarrollaron los dos puntos anteriores. El razonador semántico también puede crecer y ser usado en otros dominios. Se ha provisto de las clases y el ejemplo que demuestra que esto se puede llevar a cabo.

Fue elegido sobre otros debido a que ofrece la capacidad de introducir reglas personalizadas en un formato especial de Jena (The Apache Foundation, 2014), y además que su capacidad de deducción transitiva en su razonamiento es suficiente para cumplir con las tareas de inferencias que se requieren.

7.6.5 Fundamentos para crear nuevas aplicaciones

Se han desarrollado ejemplos para que los desarrolladores tomen como base para crear sus propias aplicaciones. Asimismo, en el contenido de este documento se en-

cuentran los aspectos técnicos que permitirán utilizar y configurar sus propios servicios de acuerdo a los requerimientos de la plataforma OpenThings. Se facilita además la creación de nuevas aplicaciones, ya que los conocimientos y la configuración que se requieren para la plataforma son menores que crearlo sin nociones básicas.

7.6.6 Técnicas propuestas para implementar razonadores semánticos

En el transcurso de las pruebas y de la implementación de la Base de Conocimientos se encontraron técnicas que pueden facilitar futuras integraciones con otros modelos semánticos, así como versiones actualizadas de los razonadores de Jena y de otras tecnologías.

- **Reutilizar conceptos e individuos en la Base de Conocimientos**. Una vez que se ha creado un concepto y un individuo existe dentro de la ontología y requiere tiempo de procesamiento para encontrarse, por lo tanto, se recomienda ampliamente que se verifique si existe el individuo dentro de la ontología y así evitar redundancia en la información.

  Para atacar este problema, se agregó una propiedad genérica para todos los individuos un **identificador único del objeto** (UID), el cuál puede estar presente en los objetos inteligentes como su dirección *MAC* o algún identificador único provisto por el fabricante de dichos dispositivos.

- **Colocar Bases de Conocimiento distribuidas con interfaces RESTful**. El paradigma de comunicación REST permite ahorro de energía, ancho de banda, y comunicación entre los dispositivos de diferente índole, sean microcontroladores cuya capa de transporte está basada en WiFi, o IEEE 802.15.4. Aunque para algunos servicios sea más conveniente implementar *SPARQL endpoints*, en el manejo de dispositivos cuya limitante principal es la batería se recomienda usar protocolos de comunicación simples, que una persona pueda utilizar para maximizar la autonomía de los dispositivos conectados a la plataforma.

7.7 Conclusiones

Al finalizar este trabajo se obtuvo una plataforma del Internet de las Cosas (OpenThings (Delgadillo-Rodríguez, 2015)) que cuenta con capacidades semánticas, un razonador implementado dentro de un Ambiente Virtual, propio de la plataforma, y un modelo ontológico que es usado por el razonador.

Fue implementada una representación del espacio y del tiempo para conocer la actividad de una persona en un modelo ontológico. Gracias a esto, se desarrolló una ontología que representa el tiempo (en forma de eventos) y el espacio (con proxémica). Se logró obtener un enfoque centrado en el humano gracias al alcanzar este objetivo.

Capítulo 8. Conclusiones y trabajo futuro

Dentro de este capítulo se describe la conclusión del trabajo, así como un resumen de las aportaciones y las limitaciones que se encontraron a lo largo del desarrollo. Además se proponen temas que se pueden abarcar para trabajos posteriores.

8.1 Conclusiones

El trabajo facilita el desarrollo de aplicaciones que requieren el uso del plano espacial y temporal, centrado en los usuarios, tomando como base las tecnologías semánticas (razonadores semánticos y ontologías), y el uso en una plataforma que facilita una comunicación semántica. El trabajo refleja una representación comprensible de las relaciones del espacio y del tiempo dentro del sistema. Permite que los desarrolladores utilicen este modelo para representar una relación entre dos objetos en términos de dimensiones proxémicas y el tiempo. La conjunción de ambas dimensiones se representaron semánticamente, se obtiene un contexto más preciso sobre la actividad de una persona para de acuerdo a la situación.

Como fundamento de la tesis se utilizó la plataforma OpenThings para que el razonamiento se llevara a cabo en un ambiente semántico. Facilita la interacción entre diferentes elementos de la plataforma, como Smart objects que controlan diferentes aspectos de un ambiente. Dentro del Ambiente Virtual desarrollado están los mecanismos para reaccionar ante una situación o evento particular. Dicho modelo está relacionado con los modelos de tipo ECA (Evento - Condición - Acción) en el sentido que un evento sucede, y un conjunto de condiciones se cumplen para llevar a cabo una acción en particular. Pero sumado a este modelo, la representación de información en ontologías ayuda a facilitar la comprensión de un dominio.

Se concluye que el razonamiento semántico funciona a esta escala y además permite identificar prácticas y obstáculos que se encontraron para futuros trabajos. La importancia radica también en la integración dentro de un ambiente del IoT que representa el espacio y el tiempo, y lo usa para tomar información del contexto.

8.2 Comparativa entre sistemas

Este trabajo fue evaluado con las características que ofrece respecto a otros sistemas. Para ello, se distinguieron y se enfocó en las siguientes características:

- Uso de tecnologías semánticas
- Conciencia temporal
- Conciencia espacial
- Razonamiento
- Proactividad
- Interacción
- Modelo semántico
- Heterogeneidad
- Interoperabilidad
- Escalabilidad
- Seguridad

8.3 Aportaciones

Entre los aportes que este trabajo tiene hacia el conocimiento se encuentran:

- El desarrollo de una ontología espacial, basada en las dimensiones proxémicas de (Greenberg *et al.*, 2011) y el uso de una ontología temporal para modelar los eventos en el tiempo.
- La integración de razonadores semánticos basados en reglas en una plataforma semántica del Internet de las Cosas.

- Creación de un servicio dentro de OpenThings que contiene una Base de Conocimientos (ontología) y el razonador semántico que lo actualiza conforme a las reglas establecidas en su funcionamiento.

- Modificación del HubServer para que soporte otros métodos necesarios para vincular los eventos de conexión y desconexión de la plataforma. Facilita la comprensión para los desarrolladores y su uso se simplifica.

8.4 Limitaciones

Las limitaciones encontradas en el trabajo están asociadas con los aspectos que no pudieron ser abarcados por falta de tiempo y complejidad. Éstas se listan a continuación:

- La introducción de razonadores semánticos facilita la categorización de los elementos, pero **requiere un conjunto de reglas para categorizar de acuerdo a parámetros requeridos.**

- **Debido al crecimiento de la ontología, el tiempo de razonamiento se prolonga.** No existen mecanismos que controlen la creación y la actualización de los individuos en la ontología y mantenerla lo más simple posible.

- **No hay un manejo adecuado de la privacidad de la información.** Ya que el trabajo se enfocó únicamente en la introducción de las ontologías y del razonamiento semántico, no fue posible introducir requerimientos para proteger la información de los usuarios.

- **No existen mecanismos para separar una Base de Conocimientos del razonador semántico.** Ambos elementos deben estar presentes en una sola entidad por la simplicidad con la que funciona de esta forma. Así se puede asegurar un tiempo de respuesta más adecuado al comportamiento que se requiere, pero pone una limitante en cuanto a su escalabilidad.

8.5 Trabajo futuro

Gracias a la evaluación de este trabajo se recomiendan varios puntos a considerar para dirigir trabajos posteriores.

- **Incorporación de tecnologías precisas de posicionamiento en interiores y medidas para las dimensiones proxémicas.** Una tecnología más precisa puede atacar varios problemas que surgieron con la localización en interiores dentro del trabajo. Pese a no ser necesario, se recomienda para verificar una interacción más granular y precisa.

- **Integrar otros tipos de razonamiento en el servicio desarrollado.** En este trabajo se hizo uso de un razonador basado en reglas, por lo que las inferencias se realizan con base en un conjunto de ellas. Una ontología con reglas, pero además con razonadores semánticos que utilicen otros métodos para razonar sobre la ontología, permitiría una integración y robustez más amplia para mejorar el comportamiento del sistema. Además se vuelve importante detallar los tiempos de razonamiento en cuanto al uso de razonadores.

- **Añadir el soporte de la dimensión movimiento.** En este trabajo se obvió la dimensión de movimiento (la combinación de las dimensiones *orientación* y *distancia* y sus cambios en el tiempo). Es importante introducirla en versiones futuras para el trabajo, de forma que se represente un cambio de distancias o alguna acción en particular por parte del usuario que permita conocer más información del contexto en el que se encuentra.

- **Definir un esquema de acciones o recetas basados en un modelo de Evento - Condición - Acción.** Para rutinas simples es posible definir esta clase de oraciones que puedan ser interpretados por razonadores y elementos de software. Esto permitirá hacer el sistema más modular y comprensible para personas que no tienen conocimiento de tecnologías de la web semántica.

- **Compartir información semántica a través de otros protocolos.** Ya sea si se utiliza CoAP para compartir información de dispositivos de baja capacidad de proce-

samiento (como motes), un sistema debe contar con comunicación semántica entre todas las entidades, y de esta forma permitir que el razonamiento se lleve a cabo con base en parámetros del ambiente.

Capítulo 9. Apéndice

En esta sección se pueden encontrar los apéndices a los que se hacen referencia en el cuerpo de este trabajo.

9.1 API

A continuación se muestra la API desarrollada para actualizar los diferentes eventos, comunicando el smartphone su información a un ambiente virtual.

Nombre	Operación	URI	Campos	Descripción
Publicar dimensión proxémica	POST	Openthings.local:{PORT}/proxemics/	{"message-type": messageType, "device-name": deviceName, "uid": uid, "type": type, "beacon-id": beaconId, "proxemic-distance": proxemicDistance, "proxemic-distance-value": proxemicDistanceValue}	Publica una dimensión proxémica en un ambiente virtual. El campo "message-type" debe ser de valor "proxemic".
Actualizar ubicación a dispositivo	POST	Openthings.local:{PORT}/locations/	{"device-name": deviceName, "uid": uid, "type": type, "place": place }	Actualiza una ubicación a un dispositivo dado por su identificador único ("uid").
Actualizar Beacon de ubicación a dispositivo	POST	Openthings.local:{PORT}/locations/	{"device-name": deviceName, "uid": uid, "type": type, "beacon-id": beaconId }	Actualiza el Beacon de ubicación a un dispositivo dado por su identificador único ("uid").
Actualizar orientación de dispositivo respecto a un objeto	POST	Openthings.local:{PORT}/proxemics/	{"message-type": messageType, "device-name": deviceName, "uid": uid, “x”: x, “y”: y, “z”: x }	Actualiza la orientación de un dispositivo dados los parámetros X, Y, y Z, provenientes de información de un magnetómetro. El campo “message-type” debe ser “device”.

Figura 9.1**: API del ambiente virtual para actualizar las dimensiones proxémicas.**

Nombre	Operación	URI	Parámetros	Descripción
Obtener nombre de ubicación por Beacon	GET	TripleStoreServer/?&beacon-id={beacon-id}	Beacon-id	Obtiene el nombre de un ambiente virtual dado el identificador del Beacon (beacon-id).

Figura 9.2**: API de TripleStoreServer.**

9.2 Ontología

A continuación se muestran las ontologías visualizadas en Protégé 5.0 beta, en el visualizador OntoGraf de la misma herramienta. La descripción de la ontología completa puede encontrarse en el capítulo Componentes y descripción de la propuesta.

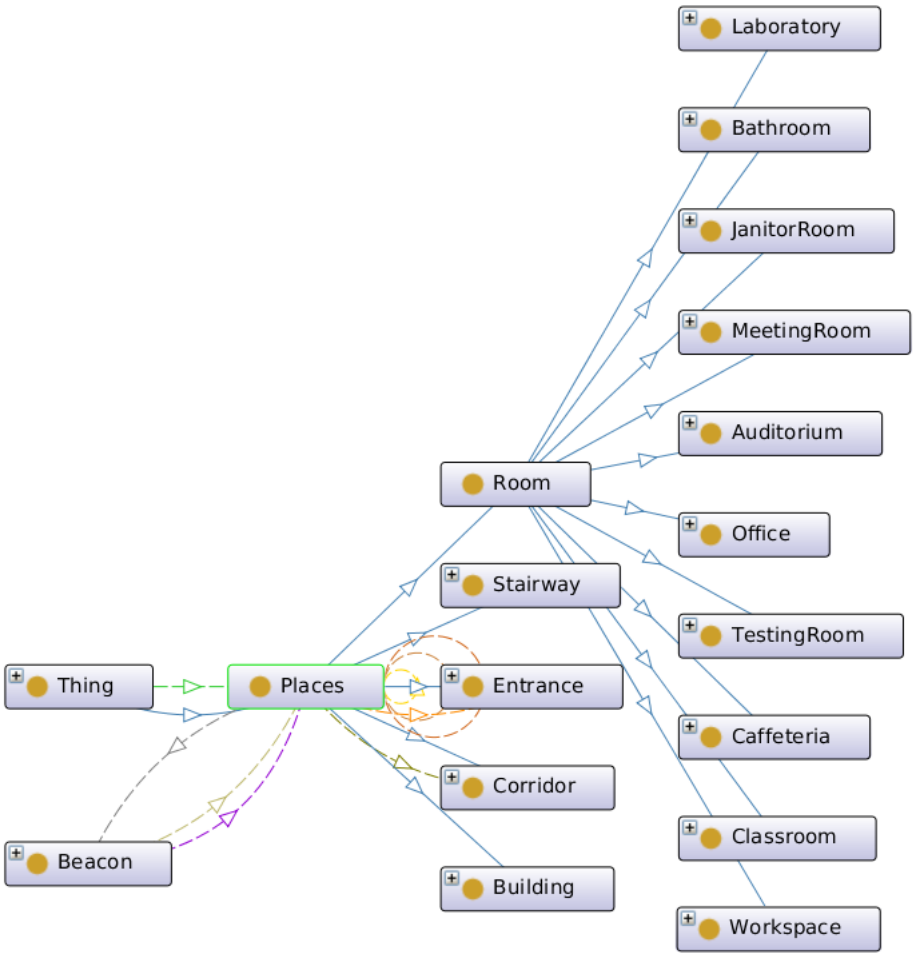

Figura 9.3**: Ontología de lugares (Places).**

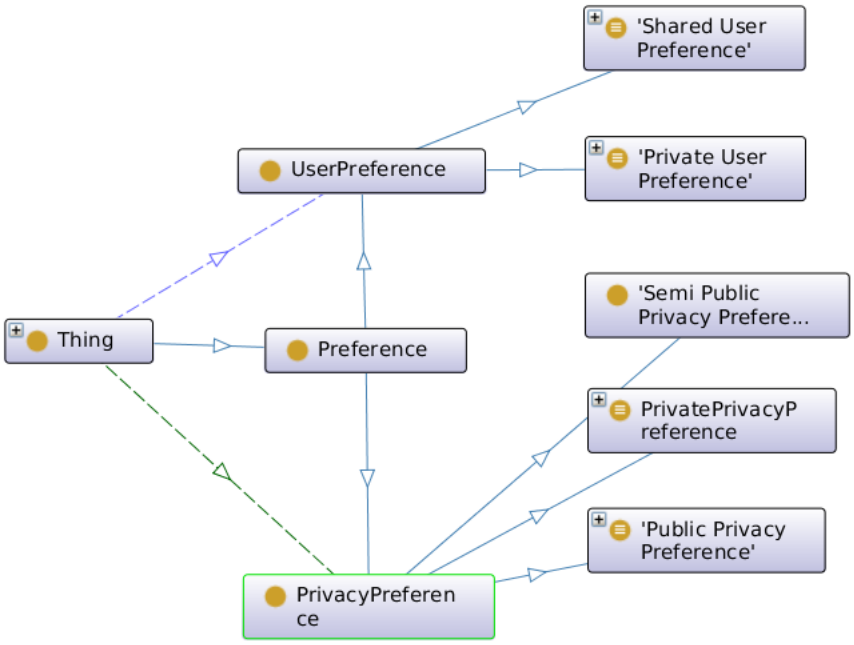

Figura 9.4**: Ontología de preferencias y privacidad de usuarios.**

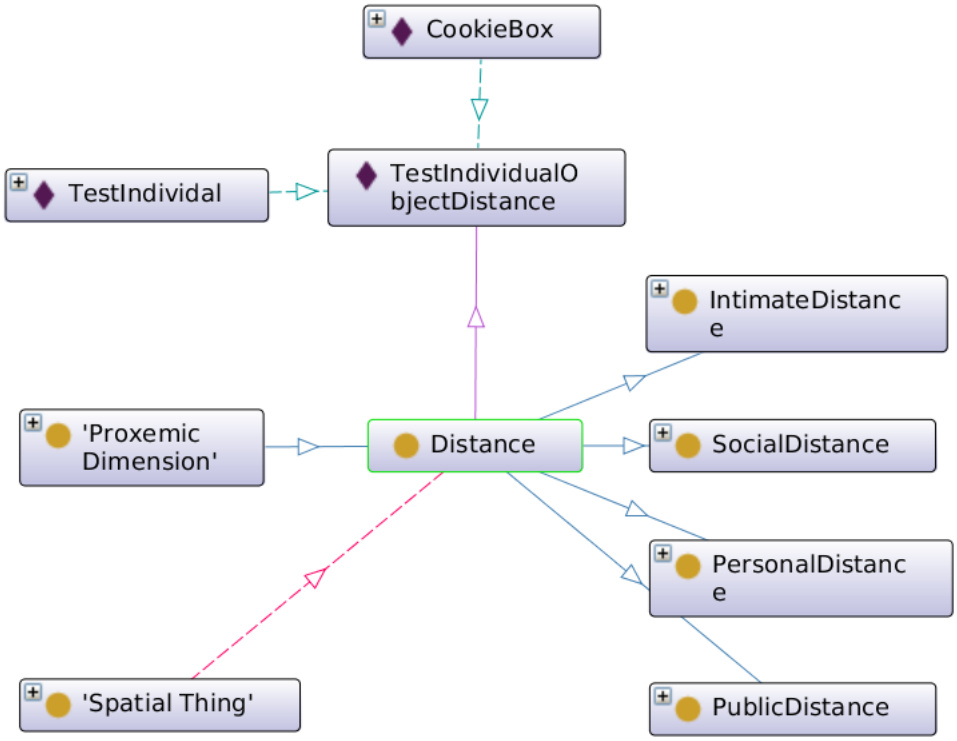

Figura 9.5**: Ontología de las distancias proxémicas (Proxemics).**

9.3 Reglas

Las reglas mencionadas de la sección Reglas se pueden encontrar a continuación:

- **Fixed rule**: [fixed_rule: (?u proxemics:canBeMoved ?v), equal(?v, ‘false’xsd:boolean) -¿ (?u rdf:type proxemics:FixedFeature)]
- **Semi fixed rule**: [semi_fixed_rule: (?u proxemics:canBeMoved ?v), equal(?v, ‘true’xsd:boolean) -¿ (?u rdf:type proxemics:SemiFixedFeature)]
- **Intimate close phase rule**: [intimate_close_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘0’xsd:int), le(?v, ‘15’xsd:int) -¿ (?u rdf:type proxemics:IntimateClosePhase)]
- **Intimate far phase rule**: [intimate_far_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘16’xsd:int), le(?v, ‘46’xsd:int) -¿ (?u rdf:type proxemics:IntimateFarPhase)]
- **Personal close phase rule**: [personal_close_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘47’xsd:int), le(?v, ‘76’xsd:int) -¿ (?u rdf:type proxemics:PersonalClosePhase)]

- **Personal far phase rule**: [personal_far_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘77’xsd:int), le(?v, ‘122’xsd:int) -¿ (?u rdf:type proxemics:PersonalFarPhase)]
- **Social close phase rule**: [social_close_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘123’xsd:int), le(?v, ‘210’xsd:int) -¿ (?u rdf:type proxemics:SocialClosePhase)]
- **Social far phase rule**: [social_far_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘211’xsd:int), le(?v, ‘370’xsd:int) -¿ (?u rdf:type proxemics:SocialFarPhase)]
- **Public close phase rule**: [public_close_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘371’xsd:int), le(?v, ‘760’xsd:int) -¿ (?u rdf:type proxemics:PublicClosePhase)]
- **Public far phase rule**: [public_far_phase: (?u proxemics:hasDistanceValue ?v), ge(?v, ‘761’xsd:int) -¿ (?u rdf:type proxemics: + PUBLICFarPhase)]
- **Is facing rule**: [is_facing: (?u proxemics:xValue ?v), le(?v, ‘-151’xsd:int) -¿ (?u proxemics:isFacing ‘true’xsd:boolean)]
- **Is not facing rule**: [is_not_facing: (?u proxemics:xValue ?v), ge(?v, ‘-150’xsd:int) -¿ (?u proxemics:isFacing ‘false’xsd:boolean)]

Lista de referencias bibliográficas

A. Dunkels, J. V. (2008). Ip for smart objects. *Personal and Ubiquitous Computing, White Paper 1*.

Adafruit Industries (2015). FLORA - Wearable electronic platform. Recuperado el 8 de octubre del 2015, de https://www.adafruit.com/products/659.

Aggarwal, R. y Das, M. L. (2012). RFID security in the context of ”internet of things”. *Proceedings of the First International Conference on Security of Internet of Things - SecurIT ’12*, pp. 51–56.

Ahmed, M., Anjomshoaa, A., y Tjoa, A. M. (2008). Context-based privacy management of personal information using semantic desktop: Semanticlife case study. En: *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, New York, NY, USA. ACM, iiWAS ’08, pp. 214–221.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, **26**(11): 832–843.

Arduino (2015). Arduino - Introduction. Recuperado el 8 de febrero del 2016, de https://www.arduino.cc/en/Guide/Introduction.

Atzori, L., Iera, A., y Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, **54**(15): 2787–2805.

Augusto, J. C. y Nugent, C. D. (2004). The use of temporal reasoning and management of complex events in smart homes. En: *ECAI*. Vol. 16, p. 778.

Augusto, J. C., Liu, J., McCullagh, P., Wang, H., y Yang, J.-B. (2008). Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home. *International Journal of Computational Intelligence Systems*, **1**(4): 361–378.

Autor desconocido (2011). OWL Reasoner. Recuperado el 27 de mayo del 2015, de https://www.w3.org/2001/sw/wiki/Category:OWL_Reasoner.

Autor desconocido (2012). Bringing reasoning to the Web. Recuperado el 27 de mayo del 2015, de http://reasoning.restdesc.org/.

Autor desconocido (2015). Auto-ID Labs. Recuperado el 20 de octubre del 2015, de http://autoidlabs.org/.

Avilés-López, E. (2012). *Creación de aplicaciones para ambientes inteligentes: un marco de trabajo basado en fuentes de datos y servicios en la Internet de las Cosas*. Tesis de doctorado, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., y Patel-Schneider, P. F., (eds.) (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. New York, NY, USA.

Barbero, C., Zovo, P. D., y Gobbi, B. (2011). A flexible context aware reasoning approach for iot applications. En: *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*. IEEE, Vol. 1, pp. 266–275.

Bardram, J. E., Hansen, T. R., y Soegaard, M. (2006). Awaremedia: a shared interactive display supporting social, temporal, and spatial awareness in surgery. En: *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. ACM, pp. 109–118.

Barnaghi, P., Wang, W., Henson, C., y Taylor, K. (2012). Semantics for the internet of things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS)*, **8**(1): 1–21.

Baziz, M., Boughanem, M., y Traboulsi, S. (2005). A concept-based approach for indexing documents in ir. En: *INFORSID*. Vol. 2005, pp. 489–504.

Berners-Lee, T. (2006). Linked data. Recuperado el 29 de mayo del 2015, de http://www.w3.org/DesignIssues/LinkedData.html.

Berners-Lee, T., Fielding, R., y Masinter, L. (1998). Uniform resource identifiers (URI): generic syntax. Recuperado el 8 de febrero del 2016, de http://www.hjp.at/doc/rfc/rfc3986.html.

Boley, H., Tabet, S., y Wagner, G. (2001). Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In proceedings of the first Semantic Web Working Symposium. Stanford University, California, USA, pp. 1–24.

Cook, D. J., Augusto, J. C., y Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, **5**(4): 277–298.

De Ruyter, B. y Aarts, E. (2004). Ambient intelligence: visualizing the future. En: *Proceedings of the working conference on Advanced visual interfaces*. ACM, pp. 203–208.

Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., y Horrocks, I. (2000). The semantic web: The roles of xml and rdf. *Internet Computing, IEEE*, **4**(5): 63–73.

Delgadillo-Rodríguez, S. (2015). *Plataforma semántica para internet de las cosas*. Tesis de maestría, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California.

Diaz-Marino, R. y Greenberg, S. (2010). The proximity toolkit and viconface: The video. En: *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA. ACM, CHI EA '10, pp. 4793–4798.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., y Burgelman, J.-C. (2003). Ambient intelligence: From vision to reality. *IST Advisory Group Draft Report, European Commission*.

Eric Prud'hommeaux, A. S. (2008). SPARQL Query Language for RDF. Recuperado el 8 de febrero del 2016, de http://www.w3.org/TR/rdf-sparql-query/#introduction.

Gaggioli, A. (2005). Optimal experience in ambient intelligence. *Ambient intelligence*, pp. 35–43.

Gárate, A., Herrasti, N., y López, A. (2005). Genio: An ambient intelligence application in home automation and entertainment environment. En: *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, New York, NY, USA. ACM, sOc-EUSAI '05, pp. 241–245.

Giaretta, P. y Guarino, N. (1995). Ontologies and knowledge bases towards a terminological clarification. *Towards very large knowledge bases: knowledge building & knowledge sharing*, **25**: 32.

Greenberg, S. y Kuzuoka, H. (1999). Using digital but physical surrogates to mediate awareness, communication and privacy in media spaces. *Personal Technologies*, **3**(4): 182–198.

Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., y Wang, M. (2011). Proxemic interactions: The new ubicomp? *interactions*, **18**(1): 42–50.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, **43**(5): 907–928.

Grütter, R., Bauer-Messmer, B., y Hägeli, M. (2008). Extending an ontology-based search with a formalism for spatial reasoning. En: *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, pp. 2266–2270.

Gubbi, J., Buyya, R., Marusic, S., y Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, **29**(7): 1645–1660.

Guinard, D., Trifa, V., y Wilde, E. (2010). A resource oriented architecture for the web of things. En: *Internet of Things (IOT), 2010*. IEEE, pp. 1–8.

Hall, E. T. (1990). *The Hidden Dimension (Anchor Books a Doubleday Anchor Book)*. Anchor.

Hobbs, J. R. y Pan, F. (2004). An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, **3**(1): 66–85.

Horrocks, Ian and Patel-Schneider, Peter F. and Boley, Harold et al. (2004). Swrl: A semantic web rule language combining owl and ruleml. Vol. 21, p. 79.

Huang, Y. y Li, G. (2010). A semantic analysis for internet of things. En: *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on*. IEEE, Vol. 1, pp. 336–339.

Hurtienne, J. (2014). Primary metaphors describe standard meanings of topological arrangements. *NordiChi 2014*.

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Recuperado el 8 de febrero del 2016, de http://www.w3.org/Submission/SWRL/.

Jakkula, V. R., Crandall, A. S., y Cook, D. J. (2007). Knowledge discovery in entity based smart environment resident data using temporal relation based data mining. En: *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, pp. 625–630.

Jara, A., Lopez, P., Fernandez, D., Castillo, J., Zamora, M., y Skarmeta, A. (2014). Mobile digcovery: discovering and interacting with the world through the internet of things. *Personal and Ubiquitous Computing*, **18**(2): 323–338.

Jerry H. Hobs, F. P. (2006). Time Ontology in OWL. Recuperado el 6 de octubre del 2015, de http://www.w3.org/TR/2006/WD-owl-time-20060927/.

Kabir, M. A., Han, J., Yu, J., y Colman, A. (2014). User-centric social context information management: An ontology-based approach and platform. *Personal Ubiquitous Comput.*, **18**(5): 1061–1083.

Kifer, M. (2008). Rule interchange format: The framework. En: *Web reasoning and rule systems*. Springer, pp. 1–11.

Kim, B., Kim, T., Ko, H.-G., Lee, D., Hyun, S. J., y Ko, I.-Y. (2013). Personal genie: A distributed framework for spontaneous interaction support with smart objects in a place. En: *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, New York, NY, USA. ACM, ICUIMC '13, pp. 97:1–97:10.

Kortuem, G., Kawsar, F., Fitton, D., y Sundramoorthy, V. (2010). Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, **14**(1): 44–51.

M. Presser, A. G. (2009). The Internet of Things: Connecting the Real World with the Digital World. Recuperado el 8 de febrero del 2016, de http://www.eurescom.eu/message.

Marquardt, N., Diaz-Marino, R., Boring, S., y Greenberg, S. (2011). The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. En: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA. ACM, UIST '11, pp. 315–326.

Network Working Group (2014). PubSubHubbub. Recuperado el 8 de febrero del 2016, de http://pubsubhubbub.github.io/PubSubHubbub/pubsubhubbub-core-0.4.html.

OWL Working Group (2012). Web Ontology Language (OWL). Recuperado el 8 de febrero del 2016, de http://www.w3.org/2001/sw/wiki/OWL.

Paschke, A. y Boley, H. (2010). Rule markup languages and semantic web rule languages.

Paton, N. W. y Díaz, O. (1999). Active database systems. *ACM Comput. Surv.*, **31**(1): 63–103.

Pederson, T., Janlert, L.-E., y Surie, D. (2010). Towards a model for egocentric interaction with physical and virtual objects. En: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, New York, NY, USA. ACM, NordiCHI '10, pp. 755–758.

RDF Working Group (2014). Resource description framework (rdf). recuperado el 8 de febrero del 2016, de http://www.w3.org/rdf/.

RedBearLab (2015). BLEShield - RedBearLab. Recuperado el 8 de febrero del 2016, de http://redbearlab.com/bleshield/.

Sadri, F. (2011). Ambient intelligence: A survey. *ACM Computing Surveys (CSUR)*, **43**(4): 36.

Sashima, A., Inoue, Y., y Kurumatani, K. (2006). Spatio-temporal sensor data management for context-aware services: Designing sensor-event driven service coordination middleware. En: *Proceedings of the 1st International Workshop on Advanced Data Processing in Ubiquitous Computing (ADPUC 2006)*, New York, NY, USA. ACM, ADPUC '06, pp. 4–.

Singh, D., Tripathi, G., y Jara, A. J. (2014). A survey of internet-of-things: Future vision, architecture, challenges and services. En: *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, pp. 287–292.

Soheili, A., Kalogeraki, V., y Gunopulos, D. (2005). Spatial queries in sensor networks. En: *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, New York, NY, USA. ACM, GIS '05, pp. 61–70.

Sommaruga, L., Formilli, T., y Rizzo, N. (2011). Domoml: an integrating devices framework for ambient intelligence solutions. En: *Proceedings of the 6th International Workshop on Enhanced Web Service Technologies*. ACM, pp. 9–15.

Sterling, B. (2005). *Shaping Things (Mediaworks Pamphlets)*. The MIT Press.

Tekli, J., Rjeily, A. A., Chbeir, R., Tekli, G., Houngue, P., Yetongnon, K., y Abebe, M. A. (2013). Semantic to intelligent web era. *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems - MEDES '13*, pp. 159–168.

The Apache Foundation (2014). Apache Jena. Recuperado el 5 de agosto del 2015, de https://jena.apache.org/.

Tim Berners-Lee, J. H. y Lassila, O. (2001). The semantic web. pp. 29–37.

Weiser, M. (1993). Ubiquitous computing. *Computer*, **26**(10): 71–72.

Working Group I1 (2006). R2ML – The REWERSE I1 Rule Markup Language. Recuperado el 8 de febrero del 2016, de http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=R2ML.

Xu, T., David, B., Chalon, R., y Zhou, Y. (2011). A context-aware middleware for ambient intelligence. En: *Proceedings of the Workshop on Posters and Demos Track*. ACM, p. 10.