

Tesis defendida por  
Roberto Alejandro Martínez Velázquez  
y aprobada por el siguiente Comité

---

Dr. Jesús Favela Vara  
Codirector del Comité

---

Dr. Luis Adrián Castro Quiroa  
Codirector del Comité

---

Dr. Israel Marck Martínez Pérez  
Miembro del Comité

---

Dr. Miguel Ángel Alonso Arévalo  
Miembro del Comité

---

Dr. José Antonio García Macías  
Coordinador del programa de posgrado en  
Ciencias de la Computación

---

Dr. David Hilario Covarrubias Rosales  
Director de la Dirección de Estudios de  
Posgrado

14 de febrero de 2013

**CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR  
DE ENSENADA**



---

Programa de Posgrado en Ciencias  
en Ciencias de la Computación

---

Diseño de componentes para el análisis del comportamiento y contexto de poblaciones de  
usuarios de teléfonos móviles

Tesis

que para cubrir parcialmente los requisitos necesarios para obtener el grado de  
Maestro en Ciencias

Presenta:

Roberto Alejandro Martínez Velázquez

Ensenada, Baja California, México  
2013

Resumen de la tesis de Roberto Alejandro Martínez Velázquez, presentada como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Diseño de componentes para el análisis del comportamiento y contexto de poblaciones de usuarios de teléfonos móviles

Resumen aprobado por:

---

Dr. Jesús Favela Vara

Codirector de Tesis

---

Dr. Luis Adrián Castro Quiroa

Codirector de Tesis

La ubicuidad de los teléfonos inteligentes así como la incorporación de sensores como el acelerómetro, brújula digital, GPS, micrófono, entre otros, ha hecho posible el identificar actividad humana a diversos niveles. Esta área de sensado con teléfonos inteligentes abre la posibilidad de que investigadores de ciencias sociales y del comportamiento puedan recolectar datos precisos sobre las personas, su comportamiento y su contexto con el fin de tener un mejor entendimiento de ciertos grupos. Con este fin han sido implementadas diversas herramientas de software para teléfonos inteligentes. Una de estas herramientas es InCense, el cual es una plataforma de sensado con teléfonos móviles que permite la adición de componentes que permiten el procesamiento de los datos recolectados.

Esta tesis presenta una modificación a la arquitectura de InCense con el objetivo de facilitar la adición de componentes y el despliegue de campañas de sensado. Una de las características de esta nueva arquitectura es que facilita el proceso de adición de componentes de manera que no se requiere de un conocimiento profundo de cómo funciona InCense ni de programación de bajo nivel con los sensores.

Como parte de las modificaciones a la arquitectura de InCense se propone la adición de un *componente genérico* como conceptualización de un componente. Esto permitió la implementación de herramientas para asistir el proceso de creación de componentes. Se realizó también un estudio de la literatura para detectar componentes relevantes para los propósitos de este proyecto de tesis e implementar algunos de ellos. Además, se presenta *OntoInCense*, ontología que modela el dominio de conocimiento de InCense que en conjunto con *InCense Manager* asisten al usuario en la creación de nuevos componentes y el despliegue de campañas de sensado. Finalmente se seleccionó uno de los componentes detectados en literatura para ilustrar las ventajas de la nueva arquitectura propuesta por medio de su implementación.

Palabras clave: **Sensado con teléfonos móviles, sensado oportunista, sensado participativo, InCense, android, diseño de componentes, clasificador.**

Abstract of the thesis presented by Roberto Alejandro Martínez Velázquez as partial requirement to obtain the Master of Science degree in Computer Science.

Design of components for the analysis of behavior and context of mobile phone users

Abstract approved by:

---

Dr. Jesús Favela Vara

Co-advisor

---

Dr. Luis Adrián Castro Quiroa

Co-advisor

The ubiquity of smartphones and the incorporation of sensors such as the accelerometer, digital compass, GPS and microphone among others have made possible the identification of human activity at various degrees. Using smartphones as sensing devices opens the possibility for researchers from social and behavioral sciences to collect precise data about humans, their behavior and context with the aim of gaining a better understanding of certain groups. To this aim, many tools for smartphones have been implemented. One such a tool is InCense, which is a mobile phone sensing platform that allows the addition of components to process the collected data.

This thesis presents a modification to the InCense architecture, carried out with the purpose of facilitating the process of adding new components and deploying sensing campaigns. One of the advantages of the new architecture is that the user no longer requires having a deep understanding of InCense or low-level, sensor-related programming skills.

A *generic component* is proposed as a conceptualization of a component, which is one of the modifications made to the architecture. At the same time the generic component enabled the implementation of tools that assist the user when adding new components to InCense. A literature review was conducted to detect components of relevance for this thesis project, some of which were designed and implemented. *OntoInCense* and *InCense Manager* are introduced as tools to assist the user in the creation of new components and the deployment of sensing campaigns. The first is an ontology that models the knowledge domain of InCense, whereas the latter is a software tool that assists the user in the implementation of new components added to the platform. Finally, one of the components identified in the literature review was selected to illustrate the advantages of the new architecture through its implementation.

**Keywords: Mobile phone sensing, opportunistic sensing, participatory sensing, InCense, android, component design, classifier.**

## Dedicatoria

*A mi madre, “Edy”. Que con esfuerzo y cuidados me formó como persona de bien.*

*A mis abuelos, Rosa y Roberto. Por concederme el honor de ser su nieto.*

## Agradecimientos

A mis directores de tesis, Dr. Jesús Favela Vara y Dr. Luis A. Castro Quiroa, por su guía, por su interminable paciencia y por brindarme la oportunidad de aprender de ellos.

A los integrantes del Comité de tesis, Dr. Israel M. Martínez Pérez, Dr. Miguel A. Alonso Arévalo, por el tiempo y esfuerzo dedicado a este proyecto, especialmente al final del mismo.

A los investigadores que forman parte del departamento de Ciencias de la Computación que durante los cursos contribuyeron a mi formación académica.

A la Dra. Ana I. Martínez García por ser guía y amiga de muchas generaciones de estudiantes.

A la Dra. Marcela D. Rodríguez Urrea que me brindó la oportunidad de trabajar y aprender de ella.

A Caro, Lydia y Jorge por su apoyo durante mi estadía en el posgrado.

A mis amigos y hermanos del posgrado, por brindarme su tiempo, su apoyo y su amistad. Gracias también por las experiencias y anécdotas que ahora tengo para contar: Gioconda Tarqui, Jessica Beltrán, Maythe Zúñiga, Susana Garduño, Valeria Mendoza, Alonso Aragón, Anuar Lezama, Ariel Quezada, Aritz Barrondo, Cesar López, Daniel Miramontes, Eduardo Martínez, Eduardo Quintana, Euler Ponce, Héctor Ramírez, Iván Ulloa, Jorge Álvarez, René Navarro y Víctor Cervantes.

A mi familia. En especial a mis tías de quien he recibido mucho apoyo y afecto.

A Alex y Lorenia, por el apoyo y amistad que me han brindado estos años, sin quienes este proyecto personal habría sido imposible.

Al Centro de Investigación Científica y de Educación Superior de Ensenada por otorgarme el honor de formar parte de la comunidad CICESE.

Al Consejo Nacional de Ciencia y Tecnología por su apoyo económico.

## Contenido

Resumen.....	i
Abstract.....	ii
Dedicatoria.....	iii
Agradecimientos.....	iv
Lista de Figuras.....	vii
Lista de Tablas.....	ix
1. Introducción.....	1
1.1. Planteamiento del problema.....	3
1.2. Preguntas de Investigación.....	5
1.3. Objetivo General.....	5
1.3.1. Específicos.....	5
1.4. Metodología.....	6
1.4.1. Revisión de literatura.....	6
1.4.2. Familiarización con InCense.....	6
1.4.3. Diseño e implementación de componentes.....	7
1.4.4. Análisis y modificación de la arquitectura de InCense.....	7
1.4.5. Caso de estudio.....	7
1.5. Organización de la tesis.....	8
2. Teléfonos móviles como plataformas de sensado.....	9
2.1. Sensado centrado en las personas.....	9
2.2. Sensado con teléfonos móviles.....	10
2.2.1. Aplicaciones de sensado por dominio de aplicación.....	12
2.2.2. Aplicaciones de sensado en función del número de participantes.....	14
2.2.3. Aplicaciones de sensado por paradigmas de sensado.....	16
2.3. Procesamiento y análisis de los datos.....	17
2.3.1. Clasificador.....	17
2.4. Redes comunitarias de similitud.....	19
2.5. Plataformas de sensado.....	20
2.5.1. On{X}.....	20
2.5.2. PRISM.....	22
2.5.3. BeWell.....	24
2.6. Resumen del capítulo.....	25
3. La plataforma de sensado InCense.....	26
3.1. Arquitectura.....	27
3.1.1. Sensores.....	29
3.1.2. Disparador de evento (Trigger).....	30
3.1.3. Cuestionario (Survey).....	30
3.1.4. Componente (Component o Filter).....	30
3.1.5. Almacenamiento (Sink).....	31
3.2. Análisis de literatura para detección de componentes candidatos a implementar en InCense.....	31
3.3. Creación de componentes en InCense.....	34
3.3.1. Podómetro.....	35

3.3.2. Distancia de recorrido .....	43
3.3.3. Detector de actividad de voz.....	45
3.3.4. Proceso de creación de un nuevo componente .....	52
3.3.5. Retos en el diseño y codificación de un nuevo componente .....	56
3.4. Creación de campañas de sensado.....	57
3.4.1. Editor InCense .....	57
3.4.2. Archivo de configuración .....	58
3.5. Resumen del capítulo.....	60
4. InCense Extendido .....	61
4.1. Proceso de creación de componentes con la arquitectura original.....	61
4.2. Componente genérico .....	64
4.3. Arquitectura extendida.....	68
4.3.1. OntoInCense .....	72
4.3.2. Generar código fuente con Java Emitter Templates (JET).....	78
4.3.3. InCense Manager .....	83
4.3.4. Diagrama de la Arquitectura Extendida .....	90
4.4. Resumen de capítulo.....	91
5. Creación de nuevos componentes con InCense extendido.....	93
5.1. Nuevo proceso de creación de componentes .....	93
5.2. Caso de estudio: Estimador de nivel de actividad.....	96
5.2.1. Estudio de OntoInCense .....	96
5.2.2. Diseño del nuevo componente.....	97
5.2.3. Registro de componente en OntoInCense .....	101
5.2.4. Edición e implementación del componente.....	107
5.2.5. Actualizar instancia de Aplicación Móvil .....	109
5.3. Resumen del capítulo.....	110
6. Conclusiones y trabajo futuro.....	111
6.1. Conclusiones .....	112
6.1.1. Aportaciones .....	113
6.2. Trabajo futuro.....	114
Referencias bibliográficas .....	116

## Lista de Figuras

Figura 1. Ejemplos de clasificación con a) máquina de soporte de vectores y b) $k$ vecinos más cercanos.....	19
Figura 2. Regla para detectar el modo de transporte y mostrar un mensaje en pantalla.....	21
Figura 3. Arquitectura original de InCense (Pérez Gamboa, 2012). .....	27
Figura 4. Proyecto de sensado creado con el Editor InCense. ....	28
Figura 5. Lectura de un acelerómetro de tres ejes en el tiempo. Valores en metros sobre segundo al cuadrado.....	37
Figura 6. Pseudocódigo para remover la fuerza de gravedad de una lectura del acelerómetro. Fuente: (“Android Developer: Sensor Event,” 2012). .....	37
Figura 7. Serie de lecturas del acelerómetro sin el componente de la gravedad. ....	39
Figura 8. Vector magnitud de una serie de datos del acelerómetro y la misma serie con la técnica de suavizado. ....	39
Figura 9. Método de suavizado de la señal (Loredo Medina, 2011). ....	40
Figura 10. Señal suavizada del acelerómetro con los eventos de pasos enmarcados en círculos. ....	42
Figura 11. Podómetro con la notación gráfica de InCense. ....	42
Figura 12. Recorrido pedestre del punto A al punto B alrededor de la cuadra. ....	43
Figura 13. Señal de audio con los periodos de silencio marcados. ....	46
Figura 14. Marcos con 50% de traslape sobre señal de audio.....	47
Figura 15. Configuración del entrenamiento del algoritmo SMO en Weka. ....	50
Figura 16. Captura de pantalla de la validación cruzada del modelo construido con Weka.....	52
Figura 17. Componente DAV expresado con la terminología de InCense. ....	52
Figura 18. Componente distribuido en 5 clases de código fuente. ....	54
Figura 19. Imagen de la pantalla principal del Editor InCense .....	58
Figura 20. Ejemplo de un proyecto simple de sensado.....	59
Figura 21. Archivo de configuración de un nuevo proyecto.....	59
Figura 22. Gráfica rica del proceso de creación de un nuevo componente.....	63
Figura 23. Diagrama de componente genérico.....	64
Figura 24. Diseño conceptual de un componente.....	67
Figura 25. Proceso de creación de proyectos de sensado.....	69
Figura 26. Proceso alternativo de creación de campaña de sensado. ....	71
Figura 27. Representación gráfica de OntoInCense, con algunos componentes (C) de ejemplo.....	73
Figura 28. Captura de pantalla de OntoInCense en Protégé. ....	75
Figura 29. Instancia de OntoInCense en el editor gráfico.....	77
Figura 30. Segmento de una plantilla. ....	79
Figura 31. Template en su forma de clase de Java (después de JET Builder). ....	80
Figura 32. Implementación del componente genérico.....	82
Figura 33. Arquitectura extendida integrada al ambiente de desarrollo Eclipse.....	85
Figura 34. Diagrama Entidad-Relación de la base de datos de componentes.....	86
Figura 35. Explorador de componentes (Explorer) - ventana principal.....	87
Figura 36. Explorador de componentes - Modificar componente.....	88
Figura 37. InCense Manager – Exporter.....	89

Figura 38. Arquitectura extendida. ....	91
Figura 39. Proceso de creación de componentes y proceso de creación de proyectos de sensado combinados.....	95
Figura 40. Documentación del componente VMag. ....	97
Figura 41. Código de procesamiento de datos en el componente MET. ....	100
Figura 42. Registro de variable de configuración en OntoInCense.....	102
Figura 43. Creación de la clase MET.....	103
Figura 44. Edición del campo configuration_var. ....	104
Figura 45. Edición del campo node_relations. ....	105
Figura 46. Ventana de creación de instancias en Protégé. ....	105
Figura 47. Ventana Forms para editar la forma del componente. ....	106
Figura 48. Edición de la forma del componente y conector utilizado en el editor gráfico. ....	107
Figura 49. Explorer con el nuevo componente MET en la lista.....	108
Figura 50. Edición de MET.....	109

## Lista de Tablas

Tabla 1. Metodología para el proyecto. ....	6
Tabla 2. Figuras de los elementos existentes en InCense, necesarios para construir proyectos. ....	29
Tabla 3. Código de colores para los sensores de acuerdo al Editor InCense. ....	30
Tabla 4. Listado de categoría o tipo de componente y su frecuencia de aparición en literatura. ....	33
Tabla 5. Velocidad de desplazamiento cada 2 pasos (Zhao, 2012).....	45
Tabla 6. Listado de los archivos audios y participantes utilizados para generar el modelo de clasificación.....	49
Tabla 7. Matriz de confusión para el clasificador DAV. Líneas son etiquetas asignadas manualmente, columnas son clasificación otorgada con el modelo. ....	51

## Capítulo 1

---

### Introducción

El uso de teléfonos móviles se ha extendido ampliamente. De acuerdo a la Unión Internacional de Telecomunicaciones (ITU, por sus siglas en inglés), en el 2010 existían cerca de 5 mil millones de suscriptores de telefonía celular en el mundo, lo que representaba alrededor del 85% de la población mundial (Kay et al., 2011).

A diferencia de un teléfono celular convencional, los teléfonos inteligentes o smartphones cuentan con sensores integrados, a los cuales se le están agregando constantemente sensores nuevos. Entre las adiciones más recientes a los modelos más nuevos se encuentra la pantalla táctil y el lector de tarjetas de frecuencia cercana (NFC, por sus siglas en inglés). Actualmente, varios de estos dispositivos cuentan con acelerómetro, giroscopio, brújula digital, Sistema de Posicionamiento Global (GPS, por sus siglas en inglés), micrófono, cámara de video, sensor de luz ambiental y de proximidad (N. Lane et al., 2010). De ahí que un Smartphone tiene la capacidad implícita de capturar y transmitir imagen, audio, localización y otros datos capturados con los sensores del dispositivo.

El propósito inicial de estos sensores fue el de enriquecer la experiencia de usuario. Por ejemplo, el acelerómetro fue incluido para cambiar la orientación de la pantalla a medida que esta gira. Sin embargo, en los últimos años, debido a un aumento considerable en sus capacidades de cómputo y sensado así como en popularidad entre los usuarios, el sentido de utilidad de los Smartphones se ha transformado paulatinamente. Además del uso convencional, los teléfonos inteligentes comienzan a percibirse como sensores inmersos en la sociedad. Algunas de las posibles aplicaciones para este nuevo tipo de sensores son descubrir las relaciones sociales en una comunidad (grafo social) (Chronis et al., 2009; Eagle & Pentland, 2005) o detectar patrones de comportamiento asociados con individuos que presentan alguna otra condición médica (Chronis et al., 2009; Madan et al., 2010). También existe la posibilidad de conocer aspectos relevantes sobre las ciudades, por ejemplo hábitos de transporte de las personas (Thiagarajan et al., 2009), estimación de carga de tráfico en vialidades (Herrera et al., 2010) o contaminación auditiva en ciertas

zonas (Tanveer et al., 2010). Estas nuevas aplicaciones para los teléfonos inteligentes abrió camino para un nuevo campo de investigación: el sensado con teléfonos móviles.

Hasta hace poco, realizar investigación sobre el campo de sensado con teléfonos móviles requería de plataformas especializadas y poco flexibles. Estas plataformas son generalmente costosas y difíciles de implementar. Como resultado de esto, la investigación sobre sensado con teléfonos móviles se vio limitada a un grupo pequeño de investigadores. A pesar de ello, la idea de usar dispositivos móviles como plataforma de sensado ha sido explorada en distintas ocasiones. En la mayoría de los casos se implementa una Aplicación Móvil especialmente diseñada para la campaña de sensado que recolecta los datos obtenidos con los sensores para almacenarlos y luego realizar un análisis posterior. Sin embargo, diseñar e implementar una Aplicación Móvil diferente para cada campaña de sensado representa un esfuerzo considerable. Algunos ejemplos de plataformas de sensado se encuentran en (Choudhury et al., 2008; Eagle et al., 2005; Herrera et al., 2010; Madan et al., 2010).

Debido a que esta área de investigación está en ascenso, hay muchos retos y oportunidades por explotar. Uno de los grandes retos de esta área es poder inferir lo que está sucediendo con el usuario del teléfono móvil y su entorno utilizando los datos que se obtienen de los diferentes sensores. Por ejemplo en (Kwapisz et al., 2011) se usa el acelerómetro del teléfono inteligente para identificar la actividad que el usuario realiza: caminar, trotar, subir escaleras, sentarse, etc. Si además de identificar la actividad realizada por el usuario del teléfono móvil se combina con datos de ubicación del individuo (ejemplo, GPS), se logra obtener información de más alto nivel, como por ejemplo que está trotando en un determinado centro deportivo.

Preparar una campaña de sensado con teléfonos inteligentes es una tarea que requiere un esfuerzo considerable. Especialmente porque, como se mencionó anteriormente, es necesario implementar una Aplicación Móvil específicamente diseñada para la campaña de sensado ya que cada campaña generalmente requiere recolectar datos de diferentes sensores y en diferentes ocasiones. De ahí que una oportunidad en este campo de investigación es la

de crear plataformas que sean flexibles para ser reutilizadas en diferentes campañas de sensado.

Entre otros esfuerzos que se han estado haciendo en diversos grupos de investigación, en el CICESE, se ha venido trabajando en el diseño e implementación de una herramienta de propósito general que pueda ser utilizada para el estudio de poblaciones de usuarios de teléfonos móviles (Pérez Gamboa, 2012). Esta herramienta, denominada InCense, es una plataforma de sensado con teléfonos móviles que cuenta con la capacidad de ser configurada para desplegar campañas de sensado con propósitos diferentes (Pérez Gamboa, 2012; Pérez et al., 2011). También tiene la capacidad de realizar sensado participativo y oportunista. Lo mismo se puede utilizar InCense para observar los hábitos de transporte de una población o descubrir el grafo social entre los individuos de la misma población.

### **1.1. Planteamiento del problema**

Los esfuerzos que se han hecho en este tema, aunque exitosos, son muy focalizados en el sentido de que realizan una tarea muy específica. Por ejemplo, en (Chronis et al., 2009; Miluzzo et al., 2008; Pentland, 2005) se busca conocer aspectos sociales sobre una población de individuos. Entre otras cosas se registra el número de interacciones sociales que tiene el individuo en un periodo de tiempo determinado. Por otro lado están los proyectos que se enfocan en aspectos de la salud, como UbiFit Garden (Consolvo et al., 2008). UbiFit Garden es un jardín virtual que crece conforme el individuo realiza ejercicio de manera regular. En este tipo de proyectos se mide principalmente las distancias de caminatas o sesiones de entrenamiento de un individuo. Así es como existen otros proyectos cuyo propósito es conocer hábitos de transporte o medir contaminación auditiva en distintas zonas de una ciudad. Sin embargo solo cumplen con un propósito y no es posible reconfigurarlas para observar otros aspectos sobre el individuo y su entorno. Dicho de otra manera, la mayoría de los proyectos de sensado con teléfonos móviles no son extensibles o reconfigurables.

La arquitectura de InCense está basada en componentes. Los componentes son piezas de software que reciben datos de los sensores, los procesan y los envían a otro componente

hasta que llegan a un módulo de almacenamiento (*sink*). De manera conceptual, un componente recibe datos de bajo nivel (directamente provenientes de un sensor) para pre-procesarlos y genera datos de más alto nivel. Ejemplo de componentes son un podómetro o un detector de actividad de voz humana. Uno de los retos que todavía existe en InCense es encontrar un mecanismo para agregar nuevos componentes a la plataforma. Cabe mencionar que ya existe trabajo orientado hacia la adición de componentes a InCense. Sin embargo, InCense se diseñó para ser utilizado por usuarios con bajo nivel técnico y se requiere más trabajo para simplificar la tarea de agregar componentes, ya que actualmente es necesario realizar modificaciones sobre el código fuente de InCense para agregar un nuevo componente.

Adicionalmente, se considera que existe la necesidad de conocer cuáles son los componentes que resultan más relevantes y que una plataforma de sensado debe incluir para lograr cierto grado de versatilidad, es decir, que se puede configurar para recolectar datos de acuerdo a las necesidades específicas de un experimento.

En síntesis, si bien existe InCense como plataforma de propósito general para la recolección de datos de usuarios de teléfonos móviles, es necesario diseñar un mecanismo que facilite la adición de nuevos componentes. Y si se tiene en cuenta que InCense está orientado a usuarios con bajo nivel de habilidad técnica en programación (Pérez Gamboa, 2012), también es necesario modificar la arquitectura de InCense para que el proceso de adición de nuevos componentes sea más fácil. Esto con la finalidad de que la plataforma de sensado (InCense) sea más versátil, flexible y configurable.

## **1.2. Preguntas de Investigación**

Tomando como base la discusión previa así como oportunidades que hay en el área, se presentan las siguientes preguntas de investigación:

*¿Cuál es el conjunto de componentes que debería incluir InCense, dada la frecuencia en que son usados en la literatura previa en el área?*

*¿Qué abstracciones de programación son adecuadas para facilitar la implementación de nuevos componentes en InCense?*

*¿Qué mecanismos pueden facilitar la adición componentes a InCense si se toma en consideración que el usuario de InCense posee bajo nivel de habilidad técnica en programación?*

## **1.3. Objetivo General**

Extender la arquitectura de InCense para proveer abstracciones adecuadas que faciliten la implementación de nuevos componentes.

### **1.3.1. Específicos**

Como objetivos secundarios del presente trabajo están:

- Crear un mecanismo que facilite la adición de nuevos componentes a la plataforma.
- Identificar un conjunto de componentes que por su relevancia en la literatura conviene incorporar a InCense.
- Diseñar e implementar alguno de los componentes identificados como relevantes para InCense.

## 1.4. Metodología

En esta sección se describe la metodología propuesta para realizar el presente trabajo de investigación. La Tabla 1 muestra una gráfica que ilustra el orden en que se realizaron las actividades.

**Tabla 1. Metodología para el proyecto.**

Etapa	1	2	3	4	5
Revisión de literatura	■	■	■	■	■
Familiarización con InCense		■			
Diseño e implementación de componentes			■		
Análisis y modificación de la arquitectura de InCense				■	
Caso de estudio					■

El trabajo de investigación se dividió en 5 etapas ordenadas de forma secuencial y progresiva, de manera que cada una de ellas se fundamenta en la anterior. Enseguida se detallan las cinco actividades que se realizaron y el orden en que se realizaron de acuerdo a la etapa a la que corresponde.

### 1.4.1. Revisión de literatura

Se revisó la literatura más reciente en sensado con teléfonos móviles siguiendo tres objetivos: conocer el estado del arte en el tema, realizar un conteo de los componentes que se han utilizado en la literatura y detectar clasificadores que ya se diseñaron y evaluaron. Esta es la única actividad que se inició en la primera etapa del proyecto y continuó hasta la conclusión del proyecto.

### 1.4.2. Familiarización con InCense

Durante esta etapa del proceso (segunda etapa) se revisó detalladamente el código fuente de la plataforma. El objetivo en esta etapa consistió en ganar el conocimiento necesario para agregar componentes a InCense utilizando la arquitectura original así como la manera en que se despliegan campañas de sensado. A manera de práctica con la plataforma se

implementaron también algunos componentes sencillos como monitoreo de rutas de transporte urbano por medio del GPS en el teléfono móvil.

#### **1.4.3. Diseño e implementación de componentes**

Se seleccionó un conjunto de componentes que, por la frecuencia con que son utilizados en la literatura o por el interés existente al interior del equipo de investigación, resultaban relevantes para implementar en InCense. El diseño e implementación se realizaron siguiendo la descripción que se hace en literatura de cada uno de ellos y en uno de los casos adaptar el componente a un dispositivo móvil. Esta etapa tiene doble propósito: Primero se aprovecha para implementar componentes que puedan ayudar en el trabajo de investigación al interior del grupo de trabajo en cómputo móvil y ubicuo del departamento de ciencias de la computación en el CICESE. La razón es que una vez implementados en InCense, existe la posibilidad de aprovecharlos en futuras campañas de sensado. Segundo, la experiencia ganada de la implementación de los componentes permite afianzar el conocimiento que se obtiene al implementarlos y marca la pauta para realizar las modificaciones a la arquitectura de InCense.

#### **1.4.4. Análisis y modificación de la arquitectura de InCense**

Se realizó un análisis de la arquitectura original de InCense. Tomando en cuenta las experiencias ganadas en las etapas anteriores de la investigación se propuso una modificación a la arquitectura de la plataforma; derivado de esta modificación propuesta, fue necesaria la creación de dos herramientas: OntoInCense e InCense Manager ambas se integran como parte ambiente de desarrollo Eclipse y que reúne todas las modificaciones hechas a la arquitectura de InCense.

#### **1.4.5. Caso de estudio**

Esta es la última etapa de la metodología. Se eligió uno de los componentes del conjunto identificado en la tercera etapa: un estimador de nivel de actividad. La elección se basó en el interés al interior del equipo de investigación por contar con un estimador de nivel de actividad que sería utilizado en distintas campañas de sensado. El objetivo que se persiguió en esta etapa fue el de mostrar las modificaciones hechas a la arquitectura en la cuarta etapa

y su funcionamiento. También se evidenció que las modificaciones hechas a la arquitectura resultan en una mejora en la creación de nuevos componentes para InCense.

### **1.5. Organización de la tesis**

En el capítulo 2 se aborda el sensado con teléfonos móviles. También se presentan los tipos de sensado con teléfonos móviles de acuerdo al nivel de participación del usuario (participativo y oportunista) o de acuerdo a su dominio de aplicación (urbano, cuidado de la salud, ambiental y otros). Finalmente se presentan un par de plataformas de sensado con teléfonos móviles y una aplicación orientada al cuidado de la salud.

En el capítulo 3 se introduce InCense como plataforma de sensado así como los elementos que conforman su arquitectura (original). También se presenta una exploración de literatura para obtener una lista de componentes que son académicamente relevantes de acuerdo a la literatura revisada. Por último se presenta el diseño de tres componentes seleccionados de esta lista.

El capítulo 4 presenta la propuesta de extensión de la arquitectura de InCense así como la descripción detallada de los nuevos elementos de la arquitectura extendida. Como parte de la propuesta de extensión para la arquitectura de InCense se encuentra OntoInCense e InCense Manager.

El capítulo 5 describe detalladamente el proceso de creación e componentes con la arquitectura extendida con el objetivo de mostrar las ventajas con respecto de la arquitectura original.

El capítulo 6 muestra un resumen del trabajo realizado durante el proyecto. También presenta una sección con las conclusiones y el trabajo que resta por realizar.

## Capítulo 2

---

### **Teléfonos móviles como plataformas de sensado**

En este capítulo se presenta la manera en que los teléfonos inteligentes se han estado usando como plataformas de sensado. En esta área se encuentra enmarcado el presente trabajo de tesis. El capítulo está dividido en tres secciones. En la primera sección se aborda el tema de sensado centrado en las personas. En la segunda sección se discute sobre sensado con teléfonos móviles, que a su vez se divide en dos paradigmas de sensado: oportunista y participativo. Finalmente, en la tercera sección se analizan algunas plataformas de sensado con teléfonos móviles que se han desarrollado en años recientes.

#### **2.1. Sensado centrado en las personas**

Gracias a la proliferación de los teléfonos inteligentes y la amplia variedad de sensores integrados en ellos, estos dispositivos tienen el potencial de sensar ciertas acciones del usuario o eventos que suceden a su alrededor. De esta manera, se convierten en una puerta de acceso al individuo y a su contexto. Tal fenómeno ha despertado interés entre la comunidad científica por estudiar la información que se captura con estos sensores, centrando su atención en el individuo y dando paso al sensado centrado en las personas (Blom et al., 2011; Khan et al., 2012).

Anteriormente, una manera de llevar a cabo sensado sobre un área geográfica era por medio de las redes de sensores. Las redes de sensores estuvieron dominadas por el uso de nodos inalámbricos de sensado<sup>1</sup>, conocidos también como motes. Ejemplos muy populares de estos dispositivos son MICAz<sup>2</sup> y TELOSb<sup>3</sup>. La mayoría de las aplicaciones de estas redes de sensores eran a pequeña escala, esto en parte se debió a que el alcance de la señal inalámbrica de un mote es limitado (hasta 100 metros con línea de vista directa). Esto hace a los motes ideales para su uso en la industria ya que suelen colocarse para monitorear

---

<sup>1</sup> Un nodo inalámbrico de sensado es un dispositivo electrónico que cuenta con capacidad reducida de procesamiento puesto que su diseño se basa en el uso de microcontroladores. Además del procesamiento tienen la capacidad de recolectar datos de sensores que se agregan al nodo y transmitirlos inalámbricamente.

<sup>2</sup> [http://www.openautomation.net/uploadsproductos/micaz\\_datasheet.pdf](http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf)

<sup>3</sup> <http://www.cs.gsu.edu/yli/teaching/Fall12/8223/slides/telosb.pdf>

maquinaria y/o el espacio de trabajo. Con la aparición de teléfonos inteligentes que poseen capacidades de sensado se abrieron otras posibilidades mucho más interesantes para generar redes de sensores de escala urbana o incluso de mayor alcance. En estas redes, el foco de interés pasa de la maquinaria y los elementos en el ambiente a la persona y su contexto. La movilidad de los dispositivos es un factor clave en estas plataformas de sensado puesto que la red de sensores tendría potencialmente la misma escala que la red de telefonía celular. Aunque la movilidad es una ventaja, también presenta otros retos, como el identificar el contexto de la persona que está siendo sensada.

Formalmente, contexto se puede definir como “cualquier información que sea de utilidad para caracterizar la situación de una entidad. Una entidad entidad es una persona, lugar u objeto que se considere relevante para la interacción entre el usuario y una aplicación, inclusive el mismo usuario y las aplicaciones” (Dey, 2001, p.4).

## **2.2. Sensado con teléfonos móviles**

Un teléfono móvil con capacidad de sensado es aquel que cuenta con sensores integrados en el dispositivo mismo; o en su defecto aquellos dispositivos que aprovechan la tecnología Bluetooth o cualquier otra interfaz de comunicación disponible en el teléfono para extender su capacidad de sensado conectándose con sensores externos (ejemplo, sensor de temperatura).

Una gran cantidad de personas cuentan con un teléfono inteligente y lo traen consigo la mayor parte del día. Esto representa una oportunidad para obtener datos que permitan modelar ciertos aspectos de su vida, de su contexto y por supuesto del usuario mismo y sus patrones de conducta. Si se analizan correctamente los datos obtenidos, se pueden detectar patrones que de otra forma sería muy difícil observar. Por ejemplo, en (Madan et al., 2010) se observa el comportamiento social de los individuos haciendo uso del GPS, Wi-Fi e interacciones sociales con el teléfono (SMS y llamadas) para encontrar una relación entre los patrones de conducta de las personas y su estado de salud.

Hasta antes de la llegada de los teléfonos inteligentes, sensar a las personas y su contexto (sensado centrado en las personas) implicaba serios retos técnicos, lo que limitaba el trabajo

hecho en el campo a un grupo reducido de investigadores. Anteriormente, era necesario implementar plataformas como Mobile Sensing Platform (Choudhury et al., 2008), que requieren de personal técnico capacitado en su programación y mantenimiento. Sin embargo, la combinación de ciertos avances como la integración de sensores baratos en teléfonos móviles y la relativa facilidad con que pueden ser programados ha dado lugar a un nuevo tipo de investigación: sensado con teléfonos móviles.

Nokia y Apple fueron los primeros fabricantes en integrar sensores a un teléfono inteligente con el Nokia N95 y el iPhone en 2007. En el caso del Nokia N95, el acelerómetro tenía la función de estabilizar la imagen en la grabación de video y cambiar la orientación de las fotos (Campbell & Choudhury, 2012). Sin embargo, los teléfonos inteligentes más modernos como el iPhone 5 (2012), Motorola RAZR HD (2012) o el Samsung Galaxy S III (2012) cuentan con una gran diversidad de sensores integrados en el dispositivo como lo son acelerómetro, sensor de luminosidad, cámara frontal, cámara posterior y GPS entre otros. El propósito original de incluir sensores en los teléfonos móviles ha sido siempre el de enriquecer la experiencia del usuario. Por ejemplo, el acelerómetro permite cambiar la orientación de la imagen en la pantalla del dispositivo cuando este es posicionado de manera horizontal o vertical. De la misma manera, la cámara frontal hizo posible realizar video-llamadas con el teléfono celular. La tendencia es que se irán integrando más sensores a los distintos modelos de teléfonos móviles conforme vayan saliendo al mercado. Por supuesto que esto beneficiará al sensado con teléfonos móviles ya que permitirá enriquecer el modelo que se pueda crear del contexto del usuario y el usuario mismo en cuanto a sus patrones de comportamiento.

Las posibilidades de aplicación del sensado con teléfonos móviles son muy variadas puesto que no solo en ciencias de la computación existe un interés en el campo, sino que en áreas pertenecientes a las ciencias médicas, sociales, ambientales y otras como el urbanismo también existe interés por estudiar los patrones de conducta de los individuos además de su contexto (Kanhare, 2011; N. Lane et al., 2011; Miluzzo et al., 2008; Zhou et al., 2012). Como se mencionó al inicio del capítulo, los teléfonos móviles han proliferado y esto convierte al sensado con teléfonos móviles en el vehículo adecuado para conocer más

acerca de una población de usuarios de teléfonos móviles o incluso, un individuo en particular.

Las aplicaciones de sensado con teléfonos móviles se pueden agrupar desde tres puntos de vista: por dominio de aplicación, por paradigma y por escala<sup>4</sup> (Khan et al., 2012; N. Lane et al., 2010). A continuación se abordan los tres.

### **2.2.1. Aplicaciones de sensado por dominio de aplicación**

A continuación se discuten algunos de los principales dominios de aplicación.

**Transporte.** La movilidad (infraestructura disponible para el transporte público) en las grandes ciudades es un problema cuya solución requiere conocer las vialidades con mayor tráfico y las horas pico entre otras características derivadas de los patrones de desplazamiento de la población. Por ejemplo, si los usuarios del sistema de transporte público proporcionan información sobre su trayecto mientras van en ruta, es posible estimar la hora de llegada de las unidades de transporte a sus respectivas paradas. Con esa información el resto de los usuarios del sistema de transporte que esperan en las paradas pueden tomar decisiones sobre alternativas para moverse en la ciudad (Zhou et al., 2012).

**Social.** Muchos usuarios participan en redes sociales en línea con frecuencia y cada vez lo hacen más con dispositivos móviles, es por ello que gran parte del trabajo hecho en el sensado con teléfonos móviles esté dedicado a entender y modelar estas conexiones entre las personas. Mucha de la información sobre las aristas del grafo social de una comunidad se puede encontrar en los Registros Detallados de Llamadas (CDRs por su nombre en inglés), si se desea conocer algo más específico como los encuentros cara a cara que tienen los nodos del grafo se puede obtener sensando la proximidad de los individuos por medio del Bluetooth o el Wi-Fi. Utilizando el micrófono del dispositivo se obtiene un mayor detalle sobre la cercanía de dos nodos en el grafo social si se registran los encuentros cara a cara de estos dos individuos. Así, al incluir más sensores, datos obtenidos de la actividad que cada individuo tiene en las redes sociales y los registros del teléfono, se enriquece

---

<sup>4</sup> Cabe mencionar que existen proyectos que resultan híbridos ya que esta clasificación no los hace mutuamente excluyentes..

todavía más el modelo del grafo social de una comunidad (Chronis et al., 2009; N. D. Lane, 2012).

Ambiental. Para conocer las condiciones ambientales de un espacio exterior o interior, es necesario valerse de instrumentos para medir parámetros como temperatura, presión atmosférica, concentración de dióxido de carbono ( $\text{CO}_2$ ) en el ambiente, humedad, nivel de ruido ambiental, entre otros. Los teléfonos móviles más modernos no cuentan con instrumentos para medir directamente todos estos parámetros sin tomar en cuenta los servicios meteorológicos disponibles por internet. Por ello, el sensado del ambiente por medio de teléfonos móviles se ha hecho de dos maneras.

La primera es midiendo indirectamente la huella de carbono que produce un individuo. La idea es que al estimar la huella de carbono que deja un colectivo de individuos en una zona, se puede proyectar las condiciones ambientales en cuanto a las concentraciones de  $\text{CO}_2$ . Esta manera de sensado indirecta requiere de software sofisticado corriendo en el teléfono que permita determinar el medio de transporte de un individuo, su nivel de actividad física para estimar la cantidad de  $\text{CO}_2$  que libera a través de la respiración y otras variables (Mun et al., 2009).

La segunda manera de sensar el ambiente con un teléfono móvil es más directa. En esta segunda aproximación, el teléfono se conecta a un sensor externo aprovechando alguna de las interfaces de comunicación disponibles en un teléfono inteligente. Por ejemplo, cuando el teléfono obtiene las mediciones de temperatura y humedad provenientes de un dispositivo hecho a la medida colocado en algún sitio de la infraestructura urbana. Lo que se plantea es que se pueden distribuir muchos sensores inalámbricos en una ciudad y estos enviarían las mediciones a los teléfonos móviles que estén en rango de distancia para recibirlos por ejemplo, a través de un enlace Bluetooth. Los datos recolectados por un teléfono luego son enviados a un repositorio común para su análisis. Como resultado se obtiene una red de sensores con cobertura urbana. De igual forma, se pueden cambiar los sensores de humedad y temperatura por otros instrumentos para medir otros parámetros (Aram et al., 2012).

Salud y bienestar. La información acerca del estado de salud que guarda un individuo, se origina principalmente cuando el paciente visita a su médico personal o en algunos casos los pacientes son internados en una institución clínica para mantenerlos en observación. Con un teléfono móvil existe la posibilidad de obtener información sobre la persona de manera continua, lo que representa una gran ventaja en el tratamiento de algunos padecimientos como obesidad, hipertensión o Alzheimer.

Existe mucho trabajo relacionado con la estimación del nivel de actividad física y el gasto energético de un individuo utilizando acelerómetros. Por ello es natural que se aproveche la incorporación del acelerómetro en los teléfonos móviles con el objetivo de realizar este tipo de estimaciones (Crouter, Churilla, et al., 2006; Härtel et al., 2010; Lee et al., 2011; Yamada et al., 2009). Igualmente se ha logrado asociar la interacción social de un individuo con su estado de salud mental. Aprovechando el micrófono incluido en el teléfono móvil se puede procesar la señal de audio y detectar si un individuo tiene una interacción con otra persona. También es posible monitorear la actividad física de un individuo así como sus interacciones sociales durante el transcurso del día, utilizando los registros diarios, el sistema hace recomendaciones puntuales al individuo para mejorar su estado de salud mental y física (N. Lane et al., 2011).

### **2.2.2. Aplicaciones de sensado en función del número de participantes**

Otra manera de clasificar las plataformas de sensado con teléfonos móviles es de acuerdo al número de participantes. Existen tres categorías de acuerdo a la cantidad de participantes: personal, grupal, y de comunidad (N. Lane et al., 2010). Cabe mencionar que aunque por ejemplo en el sensado personal podrían participar varios individuos, los datos sensados se observan y analizan a nivel individual. De la misma manera, en el sensado grupal y de comunidad se obtienen datos de varios o muchos individuos, sin embargo, en estas escalas los datos se observan y analizan desde una perspectiva de conjunto. Enseguida se discuten las tres escalas mencionadas.

Personal. Este tipo de sensado corresponde a aplicaciones donde interesa observar al individuo de forma independiente de otros. Están mayormente orientadas a la recolección y análisis de los datos. Mismos que en la mayoría de los casos están disponibles solo para el usuario y no para compartir con otros, de manera que la privacidad de los datos tiende a ser un factor de relevancia. Un ejemplo de este tipo de aplicaciones es Nike + iPod (“Nike + iPod,” 2012). Un sensor (acelerómetro) instalado en el calzado deportivo del usuario registra la aceleración del pie y transmite esos datos a un iPhone o iPod de Apple, donde se calculan parámetros de velocidad y distancia de recorrido para llevar un registro de las sesiones de ejercicio de la persona.

Grupal. Un grupo está formado por individuos que comparten una meta, interés o preocupación de forma colectiva. Este tipo de aplicación tiende a ser popular y denota un interés por las redes sociales ya que los individuos que participan suelen tener un vínculo social entre ellos (amigos, vecinos, compañeros de trabajo) además de que están dispuestos a compartir información entre los participantes, tal vez, con algunas restricciones (N. Lane et al., 2010). CenceMe (Miluzzo et al., 2008) es una instancia de este tipo de aplicaciones. Por medio de los sensores integrados al teléfono móvil se determina la actividad que el usuario está realizando así como su ubicación. Luego se comparte esta información con un grupo de contactos a través de Facebook.

Comunidad. Este tipo de aplicaciones involucran la participación de una gran cantidad de personas que no necesariamente tienen otra relación más que la de pertenecer a una comunidad (pueblo, ciudad, estado). Se obtiene una cantidad masiva de datos que se analizan y comparten por el bien de la comunidad, lo que incrementa la necesidad de contar con mecanismos para asegurar la privacidad de los datos. Por ejemplo, por medio de información proporcionada por los usuarios del transporte público en una ciudad se estima el tiempo de trayectos y hora de llegada a las diferentes paradas de autobuses de la ciudad (Zhou et al., 2012).

### **2.2.3. Aplicaciones de sensado por paradigmas de sensado**

Otra forma de clasificar las aplicaciones de sensado con teléfonos móviles es de acuerdo al rol que tiene el usuario en la aplicación. A esto se le llama paradigma de sensado y existen dos categorías: sensado participativo y sensado oportunista (N. Lane et al., 2010). En el primero es el usuario quien decide qué, cómo y cuándo sensar mientras que en el segundo se hace de manera automática. Enseguida se describen ambos paradigmas.

**Sensado Participativo.** El sensado participativo requiere un mayor involucramiento por parte del individuo. Como se mencionó anteriormente es el individuo quien decide qué, cómo y cuándo sensar. Estas decisiones representan un incremento en la carga de trabajo para el usuario. El mayor inconveniente del sensado participativo es que la calidad de los datos recolectados depende completamente del criterio del usuario puesto que es el usuario quien decide qué, cómo y cuándo tomar las muestras.

Por otro lado, el problema de determinar el contexto queda para que el usuario lo resuelva. Por ejemplo, la tarea de identificar el sitio donde se recolectan los datos (casa, trabajo, cine) o identificar la actividad que realiza el usuario (reposo, dormir, correr, etc.) es responsabilidad del usuario. Gracias a que los seres humanos realizamos estas tareas de forma cotidiana, identificar el contexto resulta muy natural para la mayoría de las personas.

**Sensado Oportunista.** Por otro lado, en el sensado oportunista, la recolección de datos esta automatizada, de manera que la participación del usuario se limita a ser portador del dispositivo.

La ventaja principal de utilizar sensado oportunista es que reduce la carga de trabajo del usuario.

El mayor reto técnico que presenta este paradigma es determinar el contexto del usuario cuando las variables contextuales son fáciles de medir. Por ejemplo, si se desea realizar una recolección de datos a cierta hora de día en un día no laboral, se puede revisar la fecha del calendario para determinar el contexto. Sin embargo, existen casos más complejos, como grabar unos minutos de audio cuando el usuario se dispone a realizar una actividad específica como conducir al trabajo. Una manera de averiguar cuando se está llevando a

cabo esta actividad podría implicar el uso del acelerómetro y el GPS. Es por esto que en muchos casos, las aplicaciones de este tipo presentan mayor dificultad técnica y requieren de un mayor esfuerzo en su implementación.

### **2.3. Procesamiento y análisis de los datos**

Una vez que se han recolectado los datos de algún proyecto de sensado, estos deben ser procesados y analizados. La razón es que existe mucha información que se encuentra contenida de manera implícita en los datos que se obtienen de los sensores. Dependiendo del interés del investigador y el objetivo de la campaña de sensado, será necesario determinar el tipo de procesamiento que hay que realizar. Por ejemplo, al observar una serie de números representando muestras obtenidas con el acelerómetro (movimiento) del teléfono como datos en crudo, ayudaría a determinar si se registró movimiento (sacudir el dispositivo). Pero si se desea ir más lejos en el análisis de los datos, en esa misma serie de números se encuentra registrada otra información de nivel más alto. Solo que para verla claramente sería necesario procesar la serie de lecturas del acelerómetro para determinar si el usuario se encontraba en reposo, caminando, corriendo, saltando o alguna otra actividad. Haciendo un análisis más sofisticado, en el análisis de los datos, se podría estimar el gasto calórico que le genera al usuario realizar una actividad como las mencionadas. Esto también es válido para otro tipo de datos, como el audio recuperado del micrófono, los datos que se obtienen del GPS o de cualquier otro sensor. Las herramientas estadísticas y de minería de datos son las más adecuadas para realizar inferencias como las mencionadas anteriormente sobre los datos recolectados (Eagle et al., 2005). Entre estas herramientas se encuentran los algoritmos de clasificación. Estos permiten segregar los datos en clases.

#### **2.3.1. Clasificador**

Un clasificador (que clasifica) es aquel que determina a que clase pertenece una persona u objeto. Por ejemplo, si se tienen varias pistas de audio y se desea determinar si existe o no voz humana en la pista de audio, un clasificador entrenado para detectar voz humana clasificaría dichas pistas de audio tomando como clases: VOZ y NO VOZ.

Los clasificadores se pueden implementar por medio de software y entrenarse utilizando técnicas de minería de datos y aprendizaje máquina. Por mencionar un par, entre las técnicas de entrenamiento asistido se encuentran los algoritmos de clasificación  $k$  vecinos más cercanos ( $k$ -NN, por su nombre en inglés) y máquina de soporte de vectores (SVM, por su nombre en inglés). Este es un tema muy extenso e interesante, motivo de muchos más proyectos de investigación por lo que se aborda de manera breve para explicar de manera conceptual su funcionamiento. Para un tratamiento más completo del tema se sugiere consultar los siguientes trabajos: (Aha et al., 1991; Platt, 1999).

Para ilustrar el propósito y utilización de los clasificadores, considere lo siguiente. Si se pretende clasificar cosas, por ejemplo un conjunto de archivos de audio, es necesario observar características como la amplitud de la señal de audio en cierto momento o la frecuencia fundamental (pitch, en inglés). En la Figura 1 se pueden observar dos planos cartesianos con dos clases de círculos (con relleno y sin relleno). Se asume que los ejes de cada uno están mapeados a las características de los círculos (característica  $X$  y característica  $Y$ ), los cuales representan archivos de audio. Algunos que contienen solamente VOZ y otros no contienen voz (NO VOZ). Los círculos rellenos representan audios de tipo VOZ y los no rellenos NO VOZ.

Para ilustrar el funcionamiento del clasificador SVM, observe la Figura 1a, en la que se observa una línea  $h$  que separa las dos clases de círculos. A esta línea en SVM se le conoce como hiperplano y es equidistante a los círculos más cercanos de cada clase (VOZ y NO VOZ). De tal forma que si se desea mapear un círculo nuevo al plano cartesiano, este caerá de un lado o el otro de la línea y entonces quedará clasificado en base al entrenamiento previo con los círculos iniciales.

Por otro lado, siguiendo la Figura 1b, observamos el funcionamiento del clasificador  $k$ -NN, en la que también se tiene dos clases de círculos. Se asume que los ejes del plano en la Figura 1b también están mapeados a las características  $X$  y  $Y$  de los audios representados por los círculos. La posición de los círculos se modificó con el propósito de mostrar más claramente el funcionamiento de  $k$ -NN. La teoría del algoritmo de clasificación  $k$ -NN dice

que si se desea clasificar un nuevo objeto se deberá calcular la distancia de ese objeto con el resto y determinar los  $k$  vecinos más cercanos. En este caso, con un  $k = 5$  se puede observar que el nuevo objeto (el círculo más grande) pertenece a la clase de los círculos rellenos.

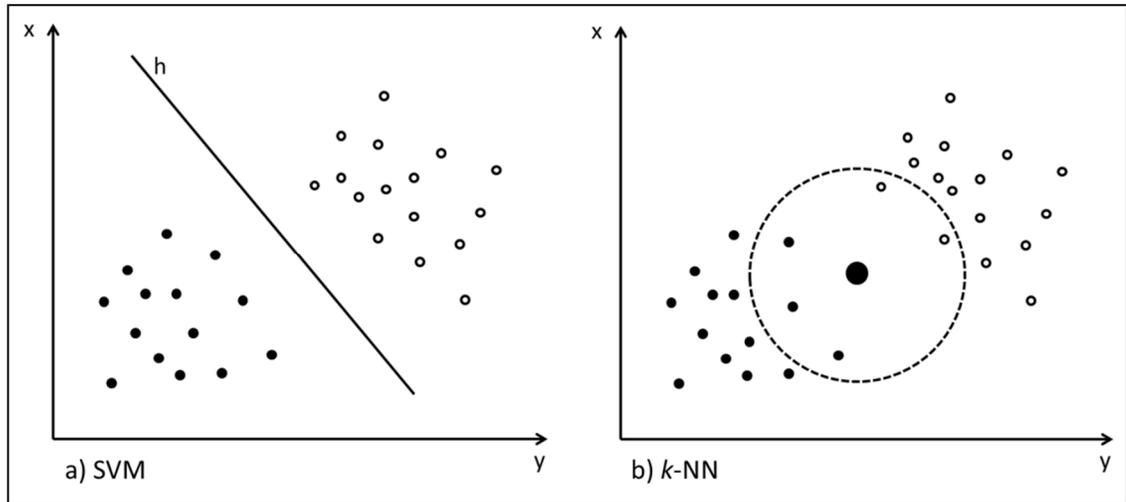


Figura 1. Ejemplos de clasificación con a) máquina de soporte de vectores y b)  $k$  vecinos más cercanos.

Existen otros clasificadores mucho más sofisticados que pueden aprender durante la misma ejecución de la aplicación, sin embargo podrían consumir demasiados recursos si se ejecutan en un teléfono móvil lo cual comprometería el rendimiento de la aplicación. Por ello, es una buena estrategia utilizar aquellos que son más ligeros como SVM que una vez que ya se realizó el entrenamiento (obtener el hiperplano) solamente requiere evaluar las características de un objeto en una función.

#### 2.4. Redes comunitarias de similitud

Diferentes aplicaciones para dispositivos móviles producen grandes cantidades de información provenientes de los sensores de los dispositivos mismos. Esta información luego es almacenada para su análisis o hacer minería de datos con ella. En general esto representa un reto técnico ya que si se utilizan clasificadores para realizar el análisis de los datos debe considerarse que la precisión de los mismos decaerá por la diversidad de los datos que se usan para el entrenamiento. Por ejemplo, un niño camina a diferente velocidad que un adulto o un adulto mayor por lo que si se clasifica la actividad puede determinarse

que el usuario está corriendo cuando en realidad está caminando. Para atender este reto se ha propuesto el uso de Redes Comunitarias de Similitud (CSN por su nombre en inglés). Las CSNs se basan en el hecho de que existen similitudes entre los usuarios que producen los datos y aprovechando las similitudes usan estos datos para entrenar clasificadores distintos para cada segmento de la comunidad. Luego los clasificadores especializados en cada segmento de la comunidad de usuarios pueden ser utilizados para realizar minería manteniendo la precisión de los clasificadores (Campbell et al., 2012).

## **2.5. Plataformas de sensado.**

El objetivo principal del presente trabajo de tesis es facilitar la creación de componentes para el sensado con teléfonos móviles y el despliegue de campañas de sensado. Por lo anterior fue necesario hacer una selección de los proyectos que son de especial relevancia para el presente proyecto de investigación. A continuación, se discuten aquellos que fueron seleccionados.

### **2.5.1. On{X}**

On{X} (“on{X} - automate your life,” 2012). Es una aplicación para automatizar algunas acciones en un teléfono celular. Entre las acciones que se pueden automatizar está “notificar cuando el usuario se encuentra cerca de un museo” o “silenciar el tono de llamada entrante cuando el usuario coloca el teléfono con la pantalla hacia abajo”. Estas acciones se denominan como reglas. Una característica muy interesante de esta aplicación es que el usuario es quien genera las reglas que desea que se ejecuten en su teléfono móvil. Dado que on{X} está dirigido al público en general es de interés para el presente proyecto de tesis estudiar cómo se aborda el problema de los usuarios con pocas habilidades técnicas.

On{X} cuenta con una plataforma web a la que se conectan los usuarios para escribir sus reglas; las reglas se programan con una API<sup>5</sup> hecha en JavaScript. Para permitir que usuarios con pocas habilidades de programación puedan usar on{X} también existen las

---

<sup>5</sup> API es el acrónimo de *Application Programming Interface*. Es un conjunto de objetos de software, funciones y métodos cuyo propósito es servir como interfaz entre las aplicaciones. Por ejemplo Facebook cuenta con una API que permite a los programadores integrar sus propias aplicaciones web con Facebook y entre otras cosas mostrar información del grafo social.

recetas, que son reglas ya hechas por otros usuarios y publicadas en la plataforma, disponibles para descargar gratuitamente. Usualmente las recetas están diseñadas para situaciones comunes a muchos usuarios.

Las reglas tienen la estructura on{X} do{Y}, es decir, cuando suceda X, realiza Y. En la Figura 2 se muestra un ejemplo de una regla creada para mostrar en la pantalla del dispositivo “conduzca con cuidado” cuando el usuario comience a conducir.

```

1 device.modeOfTransport.on('changed', function(signal) {
2   if (signal.current === 'driving') {
3     device.notifications.createNotification('Conduzca con cuidado!').show();
4   }
5 });
```

**Figura 2. Regla para detectar el modo de transporte y mostrar un mensaje en pantalla.**

Todas las reglas en la plataforma web son luego enviadas a la Aplicación Móvil para que sean ejecutadas.

Como se puede observar de la Figura 2, una regla que hace algo tan sofisticado como identificar el modo de transporte se puede implementar con relativa sencillez. Esto gracias a que la API disponible cuenta con una serie de clasificadores ya implementados y que están disponibles para el usuario. Por ejemplo, se puede reemplazar `driving` en la segunda línea de código por `running` y entonces, la regla mostraría la notificación en pantalla cuando el usuario comience a correr. Sin embargo, como se menciona con anterioridad, cuando se quiere clasificar la actividad del usuario, conviene conocer las características de un usuario puesto que la precisión del clasificador puede decaer drásticamente.

En la API de on{X}, existen los triggers estos son condiciones que deben ocurrir para que el teléfono realice una acción. Por ejemplo, de la Figura 2 podemos observar la primera línea de código y darnos cuenta de que ese es un *trigger* creado con el objetivo de realizar una acción en caso de que cambie el modo de transporte del usuario. El objeto `device` es el objeto principal de la API, de ahí derivan el resto de los objetos como `modeOfTransport`,

location, calendar, nfc, screen, y muchos otros, cada uno con sus condiciones y acciones definidas.

El mayor inconveniente de on{X}, es precisamente que los usuarios no pueden personalizar ni las condiciones ni los clasificadores que están definidas en la API.

### **2.5.2. PRISM**

PRISM (Das et al., 2010) es una plataforma de sensado con teléfonos móviles implementada en teléfonos inteligentes que usan la plataforma Windows Mobile. De acuerdo a los desarrolladores de esta plataforma, para alcanzar el potencial del sensado con teléfonos inteligentes, es necesario asegurar la facilidad de desarrollo y despliegue de estas aplicaciones de sensado. Con la propuesta de esta plataforma, sus desarrolladores buscan alcanzar este potencial. Para ello aseguran que es necesario balancear tres características que debe tener una plataforma de sensado con teléfonos inteligentes: generalidad, seguridad y escalabilidad. La seguridad tiene que ver con el acceso a información confidencial del usuario del teléfono móvil.

La generalidad resulta de particular interés puesto que consiste en soportar un amplio rango de aplicaciones con la flexibilidad de reusar código. Al igual que con InCense, los desarrolladores de PRISM buscan realizar una plataforma de sensado con teléfonos móviles que sea de propósito general. Además se pretende que sean los mismos investigadores quienes puedan personalizar y desplegar nuevas aplicaciones de sensado. De ahí que la arquitectura de PRISM y las estrategias tomadas por sus desarrolladores para lograr la generalidad en la plataforma se consideran relevantes para el presente proyecto de tesis. Misma estrategias que fueron tomadas en cuenta para realizar la propuesta de modificación a la arquitectura de InCense.

La escalabilidad se refiere a que el sistema soporte una gran cantidad de dispositivos registrados para el sensado.

PRISM tiene cuatro elementos clave en su arquitectura: el servidor de aplicaciones, los trabajos, el servidor PRISM y el cliente PRISM.

Los trabajos son archivos ejecutables que contienen el código necesario para acceder a los sensores del dispositivo, capturar los datos y procesarlos. Para crear los trabajos en PRISM también (como en on{X}) se propone una API que facilita la reutilización de código.

Los servidores de aplicación son aquellos encargados de enviar los trabajos hacia los servidores PRISM, que a su vez, distribuyen estos trabajos a los dispositivos registrados para participar en el proyecto de sensado. Por último tenemos el cliente PRISM que se encuentra en todos los dispositivos móviles y se encarga de ejecutar los trabajos en una caja de arena<sup>6</sup> (*Sandbox* en inglés) sirviendo como una capa de acceso entre los trabajos y los recursos disponibles en el teléfono.

Algunas de las características más interesantes y positivas de PRISM es que se propone una plataforma que facilita el desarrollo y despliegue de aplicaciones de sensado que sean generales, seguras y escalables. Empero, parece tener dos desventajas a considerar. La primera es que al contar con una API para crear los trabajos, se reduce el dominio de las aplicaciones que se pueden desarrollar, además de que no es muy claro hasta qué punto hay libertad con esta API para realizar el procesamiento de los datos recolectados. Segundo, si se desea desplegar una campaña de sensado con PRISM se debe tener en cuenta que esta plataforma está implementada en teléfonos que corren con Windows Mobile, lo que implica que se debe esperar una audiencia menor con respecto de otros teléfonos inteligentes que corren otros sistemas operativos más populares, por ejemplo, aquellos que corren android.

---

<sup>6</sup> En computación es el término que se utiliza para describir un entorno de ejecución de aplicaciones aislado del resto del sistema. En este se pueden ejecutar aplicaciones de forma segura con un conjunto de recursos bien delimitado (memoria, puertos, procesador). La máquina virtual de java es un ejemplo de caja de arena.

### **2.5.3. BeWell**

BeWell (N. Lane et al., 2011). Aplicación para el monitoreo de patrones de comportamiento asociados al estado de salud y bienestar de un individuo. Aunque es una plataforma de propósito específico fue relevante para el trabajo de investigación al inicio del presente proyecto de investigación ya que permitió al equipo entender mejor los alcances de una plataforma de sensado con teléfonos móviles. Sirvió también como referente para el diseño de algunos componentes que se detallan más adelante en este trabajo de investigación.

Esta aplicación está implementada en Android y hace uso de sensores como acelerómetro, GPS, micrófono y otros para medir indicadores asociados directamente a enfermedades como alta presión arterial, estrés, ansiedad o diabetes; entonces, hace recomendaciones muy puntuales para mejorar esos indicadores.

Patrones de sueño, interacción social y actividad física son los indicadores que monitorea BeWell y entre las características más relevantes de BeWell se encuentra el uso de clasificadores para detectar estas actividades. Más allá de solo detectar, mantiene un registro de estos indicadores en una plataforma web a la que el usuario puede conectarse y hacer correcciones en caso de existir una clasificación equivocada.

BeWell es una de las aplicaciones que pertenecen a la categoría Mobile Health (Kay et al., 2011) que por sus características muestran claramente el potencial que tienen este tipo de aplicaciones en el área médica y la relevancia que está cobrando el sensado con teléfonos móviles en la misma.

Desde su diseño, la aplicación se implementó con un propósito específico, lo que implica que los clasificadores utilizados para detectar los patrones de sueño o la interacción social del individuo no se pueden reutilizar o recombinar con otros, al menos no de manera práctica. También se debe pensar en lo mucho que podrían beneficiarse otros proyectos del área médica y por consiguiente la salud de los usuarios-consumidores de esas aplicaciones. Por lo tanto, queda evidenciada la necesidad de establecer mecanismos que faciliten el intercambio de estos modelos de clasificación; incluso la estandarización de estos modelos

facilitaría ampliamente la colaboración entre los integrantes de los diferentes proyectos de sensado con teléfonos móviles, puesto que existiría un marco de referencia común para el intercambio de modelos de clasificación.

## **2.6. Resumen del capítulo**

En la medida en que proliferó el uso de teléfonos inteligentes con sensores integrados, también creció el interés en la comunidad científica por utilizar estos dispositivos como un medio para estudiar a las personas y su contexto. A esto se le conoce como sensado con teléfonos móviles.

En el sensado con teléfonos móviles la atención se centra en las personas. Esto se debe a que los datos recolectados por los teléfonos inteligentes son fundamentalmente información acerca del usuario del dispositivo y su entorno. Es un campo emergente, por lo que existen muchas oportunidades para contribuir al mismo. Entre los retos aún existentes se encuentra el crear plataformas de sensado con teléfonos móviles que sean de propósito general, que sean fáciles de usar y que también provean los mecanismos necesarios para extender sus capacidades.

De acuerdo a la literatura revisada, actualmente existen ya algunas plataformas de sensado con teléfonos móviles, sin embargo son muy pocas que cumplen con los requisitos de generalidad, escalabilidad, facilidad para usar y extender sus capacidades. En el siguiente capítulo se presenta InCense como kit de investigación para el sensado con teléfonos móviles.

## Capítulo 3

---

### La plataforma de sensado InCense

Como se explicó en el capítulo anterior, existe un creciente interés en el sensado con teléfonos móviles y varios grupos de investigación han tenido que desarrollar su propia herramienta de recolección de datos. De ahí que existe una oportunidad implícita de desarrollar una herramienta de recolección que sea reutilizable y de propósito general. Para atender esta necesidad, el equipo de cómputo móvil y ubicuo del departamento de ciencias de la computación en el CICESE desarrolló una herramienta de propósito general para la recolección de datos de usuarios de teléfonos móviles denominada InCense (Pérez et al., 2011).

Cabe recordar que el objetivo principal del presente trabajo de investigación es proponer mecanismos para facilitar la creación de componentes para el sensado con teléfonos móviles (como se mencionó en el Capítulo 1). En este capítulo describimos InCense, la plataforma en la que se basa el trabajo de investigación realizado en esta tesis.

InCense es una plataforma de sensado con teléfonos móviles que permite desplegar campañas de sensado de cualquier escala y paradigma. Además, siendo de propósito general, sirve para cualquiera de los dominios de aplicación mencionados en el capítulo anterior. Una descripción más completa de InCense es: kit de investigación para facilitar la obtención de datos de comportamiento de poblaciones de usuarios de teléfonos móviles. Esta descripción implica que a diferencia de otras plataformas de sensado como on{X}, InCense está orientado a servir propósitos académicos. De ahí que para realizar el diseño se siguieron cuatro principios:

- Centrado en el usuario: InCense se diseñó pensando en investigadores con un nivel bajo de habilidad técnica (en la programación o desarrollo con teléfonos inteligentes) y con orientación hacia la investigación (Pérez et al., 2011).

- Móvil: Debido al dominio de aplicación de InCense, el kit se concibió para que fuera móvil (los teléfonos móviles son utilizados como plataforma de hardware) (Pérez et al., 2011).
- Flexible: El sistema soporta cualquier cantidad de dispositivos participando en una campaña de sensado, además de ser escalable en lo que respecta a la adición de componentes generados por el usuario (Pérez et al., 2011).
- Eventos programables: El sistema tiene la capacidad para iniciar una campaña de sensado al detectar un evento en particular por ejemplo el cambio de modo de transporte (Pérez et al., 2011).

En la siguiente sección se detalla la arquitectura de InCense (resultado del diseño basado en los cuatro principios mencionados).

### 3.1. Arquitectura

Retomando el objetivo principal del presente trabajo de investigación, se abordó el problema a nivel arquitectónico, es decir, la solución propuesta (que será detallada en el capítulo 4) requirió extender la arquitectura original de InCense. De ahí que debe hacerse una diferenciación clara entre *arquitectura original* y *arquitectura extendida*. La Figura 3 muestra el diagrama de la *arquitectura original* de InCense (Pérez Gamboa, 2012).

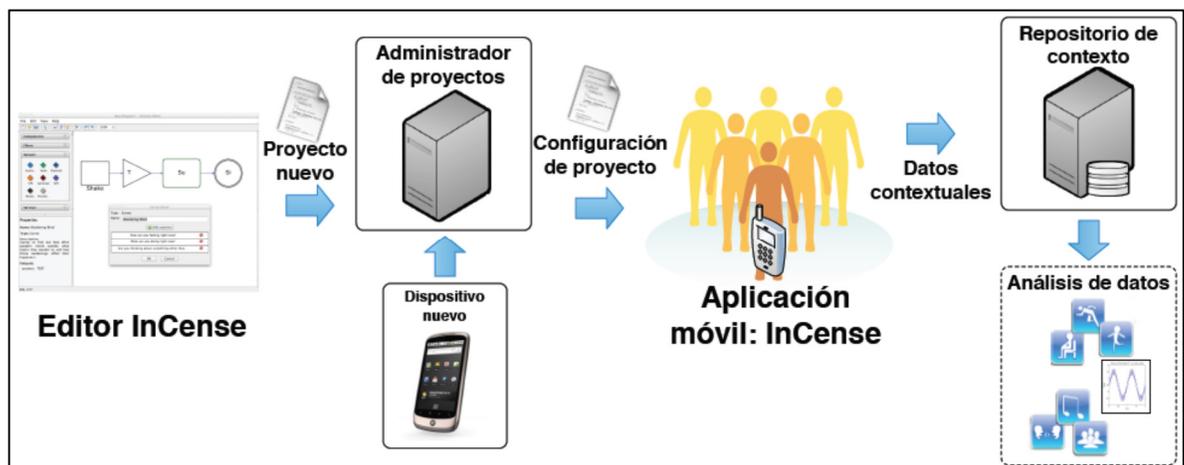


Figura 3. Arquitectura original de InCense (Pérez Gamboa, 2012, p. 43).

En la Figura 3 se observan seis elementos. Primero se tiene el Editor InCense que permite crear nuevos proyectos (representando la configuración de nuevas campañas de sensado), estos son enviados al Administrador de proyectos que selecciona los dispositivos que llevarán a cabo estas nuevas campañas de sensado. Dentro del dispositivo está corriendo la Aplicación Móvil que recibe el nuevo proyecto y se inicia la campaña de sensado, los datos recolectados y pre-procesados luego son enviados al Repositorio de contexto para su análisis.

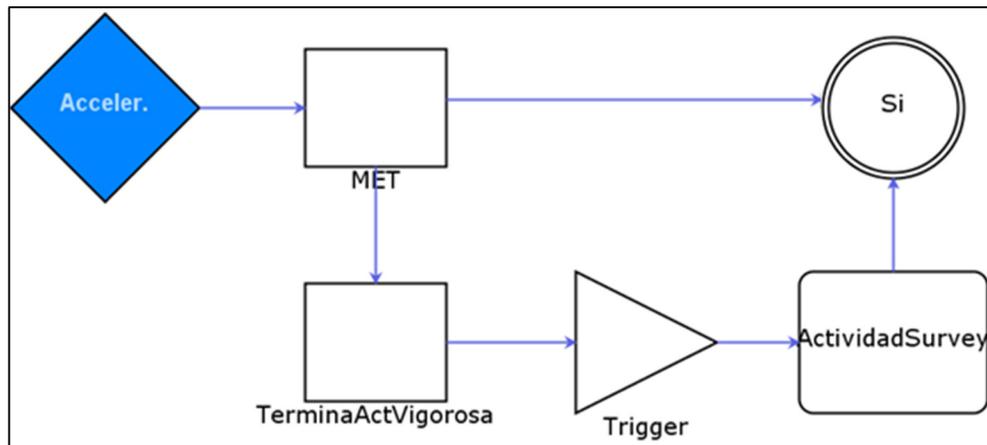


Figura 4. Proyecto de sensado creado con el Editor InCense.

Cabe mencionar que la arquitectura de InCense está inspirada en el patrón arquitectónico de ductos y filtros. Pero esto no se hace evidente hasta que se analiza a mayor detalle. Por ejemplo en la Figura 4 se puede apreciar un nuevo proyecto de sensado creado con el Editor InCense. Se trata de un proyecto para detectar cuando el usuario del teléfono ha terminado de realizar una actividad vigorosa y luego la Aplicación Móvil le solicita que conteste un cuestionario. Al inicio del flujo de datos se encuentra un sensor: el acelerómetro. Luego la salida del sensor es la entrada de un componente (creado por el usuario) para estimar el equivalente metabólico de tarea (MET) que, a su vez, transmite el resultado de su procesamiento a un componente que tiene el propósito de evaluar si el portador del teléfono ha terminado una actividad vigorosa o no. Hasta este punto, se obtiene un falso o verdadero como dato de entrada para el *trigger*. Cuando se obtiene un verdadero se inicia automáticamente un cuestionario para preguntar al individuo sobre la actividad que realizó y esto es almacenado en la memoria del teléfono. En la Figura 4 también se observa una

conexión (ducto) que va del componente MET hasta el sink (memoria del teléfono). La salida directa del componente MET se almacena también en la memoria del dispositivo a la vez que se transmite al siguiente componente. Con esta información se podría generar un modelo personalizado para estimar adecuadamente la actividad de un usuario o un grupo similar de usuarios ya que la información del cuestionario se complementa con los datos del nivel de intensidad de la actividad física así como la duración del evento.

InCense cuenta con objetos ya definidos que permiten la construcción de nuevos proyectos de sensado y que forman parte de la notación gráfica de la plataforma.

**Tabla 2. Figuras de los elementos existentes en InCense, necesarios para construir proyectos.**

Figura	Nombre
	<i>Sensor</i>
	Disparador de evento ( <i>Trigger</i> )
	Cuestionario
	Componente
	Almacenamiento ( <i>sink</i> )

La Tabla 2 muestra la representación gráfica de los distintos elementos que forman parte de la notación gráfica de InCense, mismos que se explican a detalle en las siguientes líneas.

### 3.1.1. Sensores

Es un objeto de software genérico que modela un sensor (transductor) en el dispositivo móvil. A partir de este objeto se implementa el acelerómetro o cualquier otro sensor disponible en los teléfonos inteligentes más modernos permitiendo la manipulación de

manera directa a través de la plataforma de sensado. Está presente en el Editor InCense y en la Aplicación Móvil.

Un sensor es el inicio de toda campaña de sensado con InCense y es responsable de generar datos que luego son transmitidos a un componente o directamente al almacenamiento. Su representación gráfica es un rombo con el nombre del sensor (Tabla 2). El color puede variar de acuerdo al sensor de que se trate. La Tabla 3 muestra el código de colores asignado a cada sensor de acuerdo a los colores asignados en el Editor InCense.

**Tabla 3. Código de colores para los sensores de acuerdo al Editor InCense.**

Color	Sensor
Azul	Acelerómetro
Verde	Micrófono
Marrón	Nivel de batería
Azul fuerte	Bluetooth
Naranja	GPS
Morado	WiFi
Negro	Llamadas telefónicas
Gris	Estado del teléfono

### **3.1.2. Disparador de evento (Trigger)**

Elemento que permite definir eventos específicos en el teléfono inteligente. Por ejemplo si el usuario se encuentra realizando actividad física vigorosa, al terminar la actividad, puede comenzar un cuestionario. Presente también en el Editor InCense y la Aplicación Móvil. Se representan gráficamente con un triángulo apuntando hacia la derecha (Tabla 2).

### **3.1.3. Cuestionario (Survey)**

Como se mencionó al inicio del capítulo, con InCense también se puede realizar sensado participativo, por ello existen los cuestionarios. Este objeto se puede desplegar después de que suceda cierto evento o a una hora específica. De igual manera que los sensores y eventos, estos se encuentran implementados en el Editor InCense y en la Aplicación Móvil. La Figura que lo representa es un rectángulo de esquinas redondeadas (Tabla 2).

### **3.1.4. Componente (Component o Filter)**

Este es posiblemente uno de los objetos más sofisticados en la arquitectura de InCense. En la arquitectura original de InCense se le designó como filtro, sin embargo, este nombre

generaba algo de confusión al relacionarse con el concepto de filtro en el procesamiento de señales o imágenes por lo que se decidió cambiarlo a componente. De manera gráfica se le representa con un rectángulo (Tabla 2).

Como se mencionó anteriormente, un componente tiene una entrada de datos. Mismos que son procesados y transmitidos a otro componente o en su defecto, almacenados. Funciona como una entidad, un componente realmente está formado por dos objetos: un DataFilter y un Data. El primero es responsable del procesamiento y el segundo tiene el propósito de almacenar los datos una vez que son procesados y transportarlos al siguiente componente. Además, dado que son creados por el usuario, el Editor InCense y la Aplicación Móvil deben estar sincronizados para que en ambos se encuentre implementado el componente.

### **3.1.5. Almacenamiento (Sink)**

De manera opuesta al componente, este es el objeto más sencillo de agregar a un proyecto ya que sirve para indicar el final del mismo. Representa el almacenamiento en un medio local (del teléfono inteligente) al final del flujo de datos del proyecto. Almacena lo mismo los datos provenientes de un sensor que las respuestas de un cuestionario. Su representación gráfica son dos círculos concéntricos (Tabla 2).

## **3.2. Análisis de literatura para detección de componentes candidatos a implementar en InCense**

Con el objetivo de demostrar que la plataforma puede ser usada para desarrollar una variedad de componentes y familiarizarse con los retos que su implementación implican para el desarrollador, se decidió diseñar e implementar algunos de ellos. En ese sentido, uno de los retos que existía era el de identificar cuales componentes eran los adecuados para diseñar e implementar. Por lo que se realizó un análisis de la literatura existente sobre el sensado orientado a las personas y el sensado con teléfonos móviles. El análisis involucró la revisión de 44 publicaciones en revistas arbitradas y 2 tesis de maestría. De dichas publicaciones, se identificaron 92 componentes que se pueden agrupar en 17 categorías de acuerdo a su propósito. Debido a que 3 de ellos son multipropósito y de alto nivel, se agruparon bajo la categoría de “Otros”.

La Tabla 4 muestra un listado de todos los tipos de componentes que se detectaron en la revisión de literatura. Para efectos de este trabajo de tesis, el número de incidencias (frecuencia) en una categoría denota la relevancia o la necesidad de un tipo de componente, por lo que es la manera en que se encuentran ordenados.

Para determinar cuáles componentes se iban a implementar se estableció un compromiso entre la practicidad y la relevancia académica. Esto es porque algunos de esos componentes podrían llevar meses de diseño e implementación. En algunos casos, estos aún son problemas abiertos de investigación.

**Tabla 4. Listado de categoría o tipo de componente y su frecuencia de aparición en literatura.**

<b>Categoría / Tipo de Componente</b>	<b>Descripción</b>	<b>Frecuencia</b>
Ubicación	Sitio geográfico donde se encuentra el usuario, basado en su posición exacta o como referencia de otros elementos como redes Wi-Fi.	11
Relación o Cercanía	Estima la relación que existe entre dos o más usuarios.	9
Clasificación de Sonido de Fondo	En una señal de audio, se clasifica el sonido ambiental y sus características (excluyendo la voz).	8
Detector de Actividad de Voz	Solo se trata de saber si el audio del micrófono contiene voz o no; y en algunos casos, características como el sexo o si se trata de una conversación.	7
Proximidad	La proximidad física de un usuario a otro.	6
Congestión Vehicular	Cuando un individuo se transporta en un vehículo y utilizando uno o varios sensores se detecta que el o los individuos se encuentran en una congestión o existe una congestión más adelante en el camino.	6
Velocidad de Desplazamiento	Estimar la velocidad a la que se desplaza el usuario sin importar el medio.	6
Ambientales	Mediciones sobre características ambientales como nivel de CO <sub>2</sub> , temperatura, humedad y otros con el propósito de obtener un mapa más fiel sobre condiciones actuales del clima o detectar riesgos a la salud de una comunidad.	6
Ingesta y Gasto Calórico	Estimación de: La ingesta del individuo estimada en calorías o el gasto calórico después de actividades físicas.	5
Opinión	Encuestas directas en el teléfono para conocer la opinión de un usuario acerca de un tema en específico.	5
Uso del Teléfono	Registro del uso de las características del teléfono: Multimedia, SMS, cámara, etc. El objetivo es hacer inferencias acerca de si se olvidó el teléfono en un sitio, experiencia de uso, etc.	5
Trayecto	Conocer el trayecto de uno o varios usuarios cuando se dirigen de un lugar "A" a un lugar "B"; observando patrones en el mismo.	4
Movimiento Físico	La acción (física) que está realizando el usuario, principalmente levantarse, caminar, correr, etc.	4
Biométricos	Mediciones biométricas sobre el individuo para determinar cosas como nivel de estrés, agotamiento, riesgos de salud.	4
Otros	Los componentes agrupados en esta categoría en particular son difíciles de clasificar debido a que son de alto nivel o no menciona claramente su propósito o implementación.	3
Estimador de actividad física	Clasificador de alto nivel. Que permite estimar la actividad que realiza el usuario. Por ejemplo, cocinando, comiendo, durmiendo, conduciendo, etc.	3
Medio de Desplazamiento	Determinar si el usuario se está desplazando a pie, en automóvil, en transporte público, bicicleta, etc.	3

El componente de ubicación se posicionó como el que aparece con mayor frecuencia en la literatura consultada. Además resulta interesante porque existen al menos un par de estrategias para determinar la ubicación del portador del teléfono móvil. La más directa es obteniendo las coordenadas de latitud y longitud del GPS. Sin embargo no siempre es

posible obtener una lectura del GPS, muchas veces por dificultades climáticas o porque no hay línea de vista con los satélites del sistema. En estos casos resulta muy conveniente conocer la ubicación de distintos puntos de acceso a redes Wi-Fi. Comparando las redes Wi-Fi que se encuentran en rango del alcance al teléfono inteligente con la lista de redes conocida es posible estimar una ubicación para el portador del mismo. Esta es una alternativa ingeniosa ya que además de la ubicación del individuo, generalmente provee de un contexto. Por ejemplo si el individuo se encuentra en una red Wi-Fi asociada a un cine o un restaurante por tiempo prolongado es posible inferir la actividad o rango de actividades que podría estar realizando el portador del teléfono.

A medida que las ciudades crecen, también se incrementan los problemas de tráfico y la contaminación derivada del aumento de vehículos en una ciudad. Por ello resulta de particular relevancia estudiar el fenómeno de la congestión vehicular. El componente para detectar una congestión vehicular es bastante sofisticado ya que para implementarlo es necesario analizar de manera conjunta los datos recolectados de múltiples sensores. Por ejemplo, el acelerómetro permite conocer la velocidad de desplazamiento y las desaceleraciones que se hacen de forma abrupta (muy frecuentes en un embotellamiento). Por otro lado se encuentra el GPS que permite conocer la ubicación del automóvil y su trayectoria. Vistos como un conjunto, los datos del GPS de todos los automóviles también permiten conocer la cantidad de vehículos concentrados en un segmento determinado de una vía, lo que es un indicio de una congestión vehicular.

### **3.3. Creación de componentes en InCense**

Después de analizar los componentes, se seleccionaron cuatro de ellos para ser diseñados e implementados: podómetro, estimador de distancia de recorrido (a pie), detector de actividad de voz, y estimación de nivel de actividad física. El diseño de los primeros dos resultó ser una tarea relativamente sencilla ya que existe bastante literatura que detalla claramente su implementación de manera que solo fue necesario emular el trabajo documentado en dicha literatura. Por el contrario, la detección de actividad de voz es un problema abierto y en el que se ha realizado una cantidad importante de trabajo de investigación. Una de las lecciones aprendidas con la implementación de los componentes

seleccionados viene de observar que el procesamiento de los datos en algunos componentes se hace por etapas y que, en algunos casos, más de un componente requiere de realizar la misma etapa en el procesamiento. Ejemplo de esto es el algoritmo de la transformada rápida de Fourier (FFT<sup>7</sup>, por sus siglas en inglés) que suele ser muy utilizado en el procesamiento de señales. Por esto vale la pena dividir un componente en varios para reutilizar el mismo código en la creación de varios componentes. A continuación, se muestra el trabajo de diseño del podómetro, distancia del recorrido y el detector de actividad de voz humana.

### **3.3.1. Podómetro**

Con este componente se pretende emular la funcionalidad de un podómetro. El enfoque que se tomó fue el de dividir este problema dos, primero se debe detectar un paso y después contabilizar todos los pasos.

El problema de detectar el evento de un paso de un individuo a partir de la señal de un acelerómetro ha sido tratado de manera exhaustiva. Por mencionar algunos de los esfuerzos hechos al respecto se encuentran (Kwapisz et al., 2011; Lee et al., 2011; Wang et al., 2011; Zhao, 2012). Incluso es posible diferenciar cuando se da un paso en cuesta o subiendo escalones como se documenta en el trabajo de (Knight et al., 2006). Sin embargo, el objetivo que se persigue con el presente trabajo de tesis no es aportar al problema específico de detección de un paso, por lo que para efectos de esta tesis es suficiente implementar un podómetro sencillo. De este modo, el componente analizará la señal proveniente del acelerómetro (sensor) y deberá ser capaz de detectar los eventos de paso que se registren en la misma sin importar la dirección o velocidad del mismo.

Para realizar el análisis de la señal del acelerómetro se eligió la técnica reportada en (Loredo Medina, 2011) debido a su simplicidad y al hecho de que la técnica no requiere que el teléfono se encuentre orientado con respecto del cuerpo del individuo. El hecho de que no se asume una posición fija para el teléfono inteligente es muy conveniente porque no existe una convención entre los usuarios de teléfonos inteligentes con respecto de cómo portar el dispositivo (bolsillo, cinturón, colgado del cuello, etc.). Lo que si se requiere es

---

<sup>7</sup> Algoritmo para calcular la transformada discreta de Fourier y permite pasar una señal del dominio del tiempo al dominio de la frecuencia.

que el teléfono esté tan cerca como sea posible del centro de masa del individuo que lo porta (el área abdominal).

La Figura 5 muestra una serie de lecturas obtenidas del acelerómetro de un teléfono inteligente (Xperia Neo<sup>8</sup>) que corre Android como sistema operativo. En esta figura se puede observar datos de aceleración en metros por segundo al cuadrado ( $\frac{m}{s^2}$ ) contra el número de muestra. Cada muestra obtenida del sensor de tres ejes en el dispositivo móvil tiene una etiqueta de tiempo en milisegundos que indica el momento exacto en que se obtuvo. Además, se tiene un valor de aceleración para cada uno de los tres ejes: X, Y y Z. Por motivos de espacio en la gráfica, a cada lectura se le asigna un índice que representa el orden en que fue tomada, entre cada muestra existen aproximadamente 20 milisegundos de separación, de manera que la gráfica corresponde a aproximadamente 4.5 segundos de lectura del acelerómetro.

Como se indica en la Figura 5, el eje X está representado por una línea sólida (serie X), el eje Y está representado por una línea punteada (serie Y) y el eje Z se dibuja como una línea de punteado más fino (serie Z). Idealmente el acelerómetro del teléfono debería reportar valores de 0 cuando el dispositivo se mantiene inmóvil o al menos cruces por cero cuando se cambia de dirección. Como se observa en la Figura 5, no es el caso puesto que el eje Y y el eje Z están mostrando magnitudes muy lejanas a 0. Esto se debe a que la fuerza de gravedad está influenciando estos valores. Por supuesto que la fuerza de gravedad está distribuida entre los ejes del acelerómetro de acuerdo a la orientación del teléfono móvil y esta distribución puede variar conforme se modifique la orientación del dispositivo. En algunos casos la fuerza de gravedad puede registrarse en solo uno de los ejes (cualquiera de los tres), en solo dos de ellos o en los tres. Cabe mencionar que este es el comportamiento esperado por parte del acelerómetro puesto que un individuo puede portar el teléfono móvil de múltiples formas.

---

<sup>8</sup> <http://www.sonymobile.com/mx/products/phones/xperia-neo/>

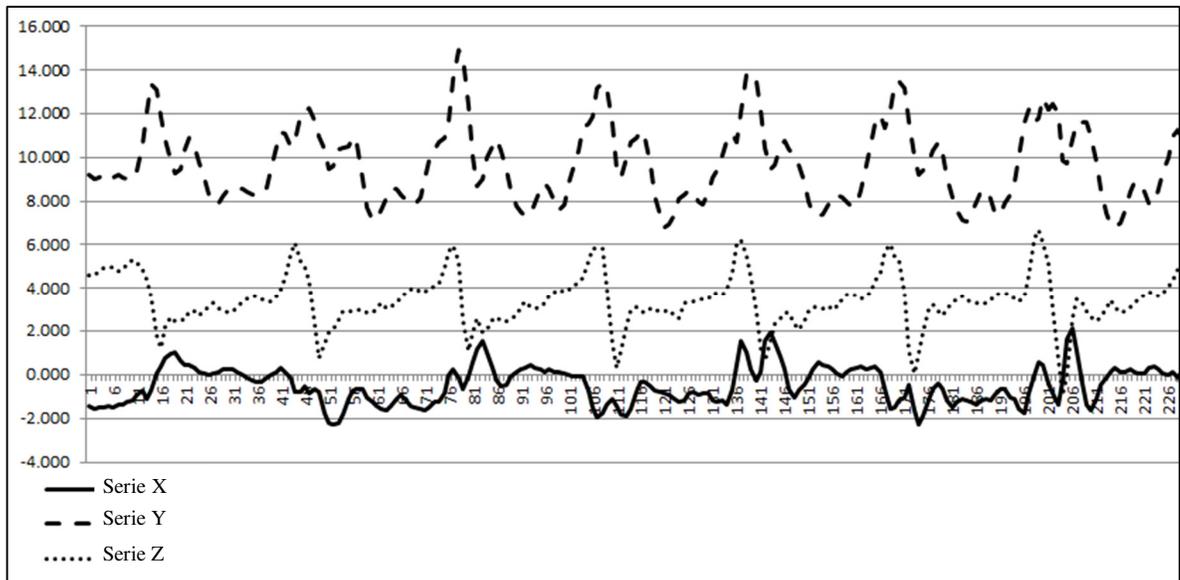


Figura 5. Lectura de un acelerómetro de tres ejes en el tiempo. Valores en metros sobre segundo al cuadrado.

La técnica reportada en (Loredo Medina, 2011) para detectar un paso, asume que el acelerómetro solo está arrojando valores de la aceleración causada por el movimiento del usuario, es decir, sin tomar en cuenta la gravedad. Esto es un inconveniente porque si se observa en la misma Figura 5, la Serie Y presenta valores que varían entre el rango de  $6 \frac{\text{m}}{\text{s}^2}$  y  $16 \frac{\text{m}}{\text{s}^2}$ . Esto indica claramente que el vector de la gravedad no ha sido eliminado de la lectura de los tres ejes, lo que fue confirmado de la documentación de Android (“Android Developer: Sensor Event,” 2012). En la misma documentación se propone una técnica para eliminar la fuerza de gravedad reflejada en los tres ejes: un filtro pasa altas (*high-pass filter*, en inglés). La Figura 6 muestra el pseudocódigo para este propósito.

```

1 final float alpha = 0.8;
2 gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
3 gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
4 gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];

5 linear_acceleration[0] = event.values[0] - gravity[0];
6 linear_acceleration[1] = event.values[1] - gravity[1];
7 linear_acceleration[2] = event.values[2] - gravity[2];

```

Figura 6. Pseudocódigo para remover la fuerza de gravedad de una lectura del acelerómetro. Fuente: (“Android Developer: Sensor Event,” 2012, p. 1).

Inicialmente `gravity` contiene valores en 0, al igual que `linear_acceleration`. La variable `event.values` contiene los valores de aceleración de la muestra obtenida del acelerómetro en ese momento (cambia conforme se obtienen nuevas muestras). Estas 7 líneas de código se ejecutan cada que se toma una muestra del mismo acelerómetro como si se encontraran en un ciclo, por ello es que los valores de `gravity` y `linear_acceleration` evolucionan constantemente. El código asume que se pretende remover la gravedad de todas las lecturas y no de una sola por separado. De ahí que que la magnitud de la gravedad en cualquiera de los tres ejes de una lectura del acelerómetro como las mostradas en la Figura 5 depende del valor de la lectura anterior (sin la gravedad). El resultado de la lectura del acelerómetro después de remover la gravedad de cada uno de los tres ejes es almacenado en el vector de tres posiciones `linear_acceleration`. El resultado de remover la gravedad de las 3 series que se observan en la Figura 5, se puede ver en la Figura 7 (utilizando el algoritmo mencionado anteriormente).

Como se puede observar en la Figura 7, al inicio de la nueva serie se registran valores altos de aceleración, esto se debe a que el valor calculado de aceleración lineal de una lectura del acelerómetro depende directamente del valor de la lectura anterior, lo que provoca un periodo de estabilización del algoritmo al inicio de la serie. Sin embargo, solo se requiere de una fracción de segundo para ello.

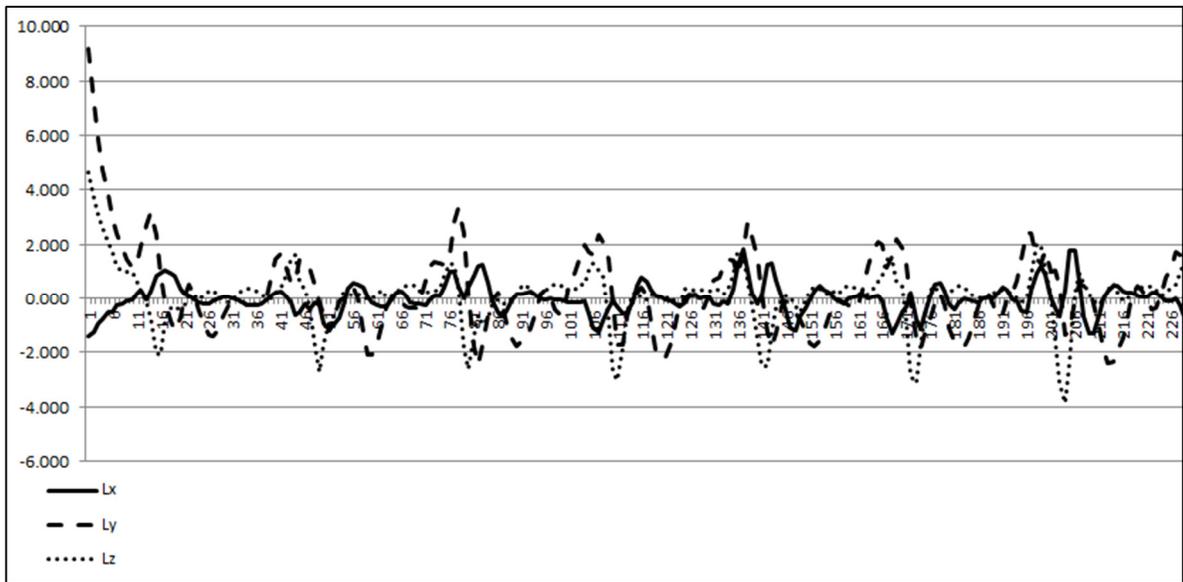


Figura 7. Serie de lecturas del acelerómetro sin el componente de la gravedad.

La técnica reportada en (Loredo Medina, 2011) todavía requiere que se haga una operación adicional con los datos del acelerómetro: el cálculo del vector magnitud de cada una de las lecturas en la serie. En la Figura 8 se puede observar el resultado de esta operación.

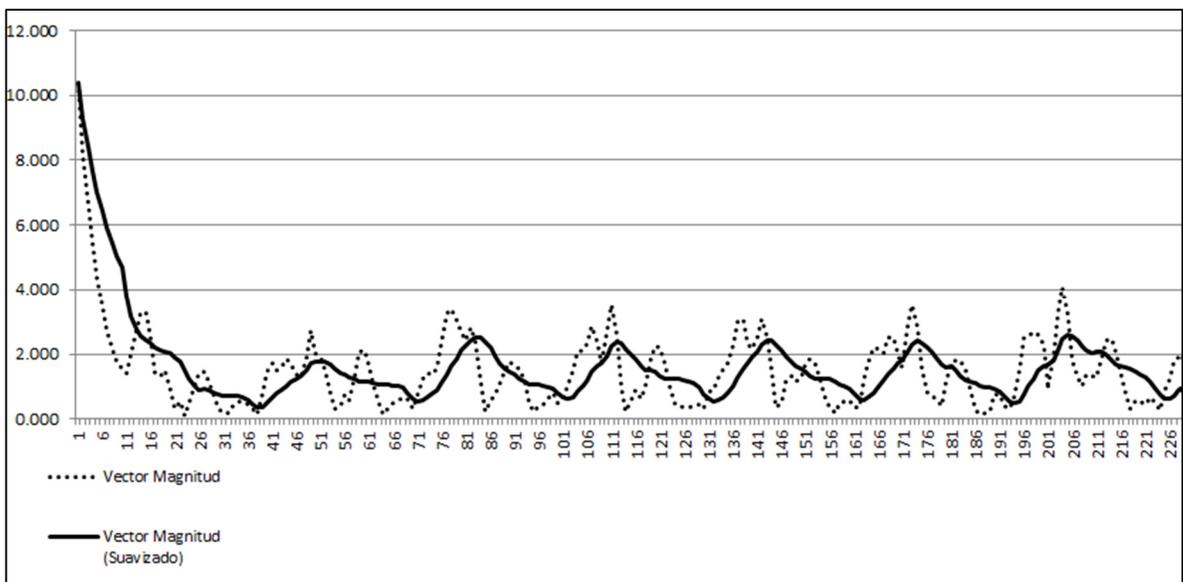


Figura 8. Vector magnitud de una serie de datos del acelerómetro y la misma serie con la técnica de suavizado.

Siguiendo con el procedimiento descrito en (Loredo Medina, 2011) en cualquier segmento de la serie que corresponde a un solo paso, varias zonas de inflexión<sup>9</sup> de la señal (vector magnitud en la Figura 8) por lo que es necesario realizar un suavizado de la serie utilizando una técnica conocida como promedio móvil que se describe a continuación (véase Figura 9).

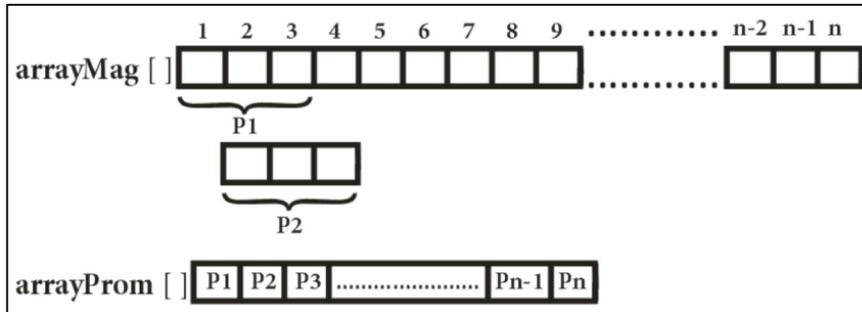


Figura 9. Método de suavizado de la señal (Loredo Medina, 2011, p. 53).

Inicialmente se tiene un vector de tamaño  $n$  con una serie de números que se pretende tratar como una señal y suavizarla. Para ello se requiere tomar los primeros  $x$  números en la serie y calcular la media aritmética de estos números para obtener el primer dato de la serie suavizada. Suponiendo  $x = 3$  se tomarían las posiciones 1, 2 y 3 del vector original (véase Figura 9) y se calcula su media aritmética para obtener la primera posición de la nueva serie suavizada. Enseguida se corre hacia la derecha un espacio en el vector original y se toman las posiciones 2, 3 y 4 para calcular el siguiente valor de la posición 2 del vector suavizado. Esto se repite hasta llegar a  $n - 1$ . Como se puede intuir la media aritmética se tendrá que calcular con los últimos dos números de la serie original (no existe una posición  $n + 1$ ). Análogamente para la última posición del vector suavizado, solo se copiará la posición  $n$  del vector original.

En la técnica reportada en (Loredo Medina, 2011) se sugiere un  $x = 3$ . Sin embargo, después de probar con varios valores, se determinó que el componente que se diseñó para el presente trabajo de tesis tuviera un  $x = 10$  ya que mostró ser el que proporciona una señal más suavizada sin perder la información de un paso en la señal. Es decir, si el valor de  $x$  es

<sup>9</sup> No confundir con el concepto matemático de punto de inflexión. Esta expresión se refiere únicamente al hecho de que en ciertas zonas la señal cambia su tendencia a aumentar o decrecer en magnitud.

muy grande se puede perder por completo la forma de la señal y tener una línea casi plana de la serie suavizada. El resultado de aplicar la técnica de suavizado al vector magnitud, se puede observar también en la Figura 8.

La técnica reportada en (Loredo Medina, 2011) continúa con la identificación de segmentos de inflexión en la señal suavizada y supone que si se detecta que la señal decrece en magnitud después de haber mostrado un incremento se puede contabilizar con un paso. Después de haber implementado el podómetro tal cual se propone en ese trabajo, se observó que se contabilizaban pasos incluso con usuarios estando en reposo. Este problema se podría atribuir a la diferencia del hardware utilizado.

Después de analizar la señal se detectó que existían zonas de inflexión con poca separación de tiempo entre ellas, producto de ruido en la señal que no pudo ser eliminado con la técnica de suavizado, de manera que cualquier movimiento ligero se contabilizaba como un paso. Por ello fue necesario hacer una modificación y establecer un umbral de energía para validar que se trata de un paso o un movimiento de intensidad similar. El umbral se calculó con los datos obtenidos de una campaña de sensado conducida entre los meses de octubre y diciembre de 2011, más detalles de esta campaña se pueden encontrar en (Pérez Gamboa, 2012).

El proceso para calcular el umbral fue el siguiente: En primera instancia se detectaron periodos de tiempo en que los individuos hacían caminatas. Enseguida, de cada individuo se calculó la media aritmética del máximo y el mínimo de toda la serie. Luego se promediaron estos valores para obtener un umbral que fuera válido al menos para ese grupo de individuos en particular (en su defecto, individuos con características de movilidad, estatura y peso similares). Al final, el umbral obtenido fue de  $1.2 \frac{\text{m}}{\text{s}^2}$ .

Cuando la señal hace un cruce de pendiente negativa por este umbral se contabiliza como un paso. En la Figura 10 se puede observar cuales eventos de cruce por pendiente negativa serían contabilizados como pasos.

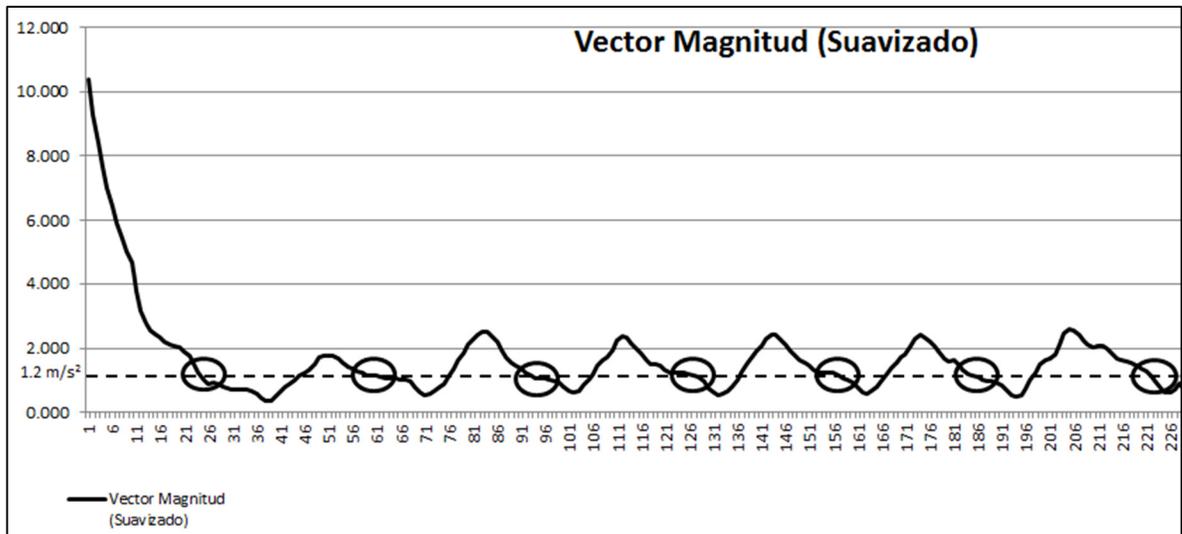


Figura 10. Señal suavizada del acelerómetro con los eventos de pasos enmarcados en círculos.

Después de detectar cada paso de forma individual, la siguiente tarea es contabilizarlos. De este modo, el componente de contar pasos consiste en:

- Obtener los datos del acelerómetro (Acc).
- Aplicar el filtro pasa altas (HP\_Filter).
- Calcular el vector magnitud (VMag).
- Suavizar la señal y detectar los cruces por pendiente negativa (Step\_Cntr).

Utilizando la notación de InCense el componente puede expresarse como se muestra en la Figura 11.

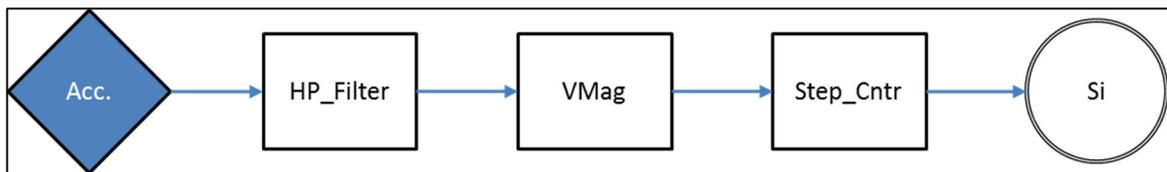


Figura 11. Podómetro con la notación gráfica de InCense.

Este componente fue implementado en InCense, pero no se muestra el proceso de implementación debido a que se reserva la explicación detallada de cómo se crea un componente para el caso de estudio que se presenta en el Capítulo 4.

### 3.3.2. Distancia de recorrido

Algunos componentes toman como entrada de datos la salida de otro componente, este es el caso del estimador de distancia del recorrido (pedestre). Estimar la distancia de un recorrido se puede realizar al menos de dos maneras.

La primera es utilizar las lecturas de GPS que se puedan hacer con el teléfono, pero con esta técnica si no se cuenta con una línea de vista directa a los satélites de posicionamiento puede generar errores muy grandes. A esto se agrega el error provocado por la precisión que tenga el GPS integrado al teléfono y de la frecuencia de muestreo.

Por ejemplo, en la Figura 12 se puede observar un recorrido a pie desde el punto A al punto B marcado por la línea de punteado más grueso.

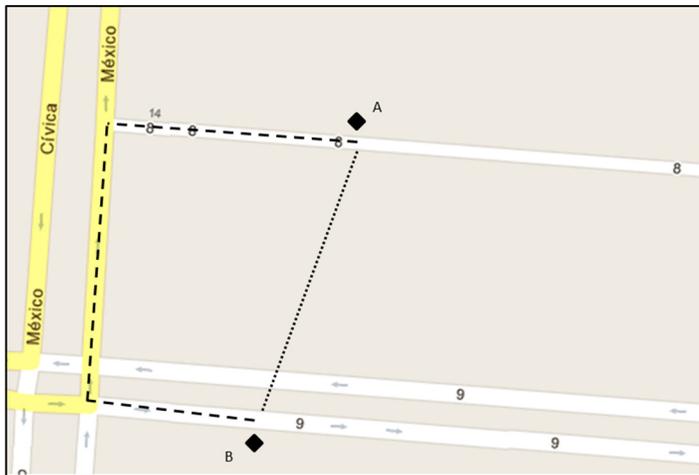


Figura 12. Recorrido pedestre del punto A al punto B alrededor de la cuadra.

En este recorrido, el individuo da vuelta en dos ocasiones para llegar a su destino. Suponiendo que A y B coinciden con las únicas lecturas de GPS que se tienen del recorrido, entonces la estimación de la distancia del trayecto (utilizando solo la información del GPS) seguramente sería igual a la distancia recta entre estos dos puntos (la línea de punteado más fina). Por supuesto que existen tecnologías como la API de Google Maps que permitiría calcular la distancia de un recorrido del punto A hasta el punto B tomando en cuenta las calles que se encuentran trazadas en sus mapas. Con esto se eliminaría el problema de calcular la distancia con la línea recta entre estos dos puntos. Sin embargo,

existe más de una forma de llegar desde A hasta B y la API de Google Maps calculará el recorrido más corto entre A y B, lo que no necesariamente corresponde al recorrido hecho por el usuario. Tal estimación de la distancia del recorrido en ambos casos corre el riesgo de diferir de la distancia real del recorrido.

En caso de que se requiera mayor precisión, es necesario tener puntos intermedios en el recorrido y mientras más puntos se tengan, más precisa será la estimación. Pero existe el inconveniente de que el tiempo entre las lecturas del GPS puede variar dependiendo del clima, de la disponibilidad de satélites, incluso si el usuario está usando el teléfono debajo de la ropa. En algunos casos puede tardar varios minutos. Esto implica que otro medio de estimación de distancia de recorrido es necesario, sobre todo porque no se requiere conocer el camino que tomó el individuo, sino la distancia que recorrió.

Como se mencionó al inicio de la sección existe otra manera de estimar la distancia del recorrido. De acuerdo a (Zhao, 2012) existe una relación directa entre la estatura de un individuo y la velocidad de desplazamiento. Implicando que cuando el individuo da un paso se puede estimar la distancia de desplazamiento calculándola a partir de la velocidad de desplazamiento en cada paso. De la Tabla 5 se puede observar una correspondencia directa entre la cantidad de pasos dados cada dos segundos y la velocidad de desplazamiento estimada para esos dos segundos, de ahí que una estimación cercana a la distancia recorrida por un individuo durante esos dos segundos se calcula multiplicando la la distancia de la zancada por los pasos (Zhao, 2012).

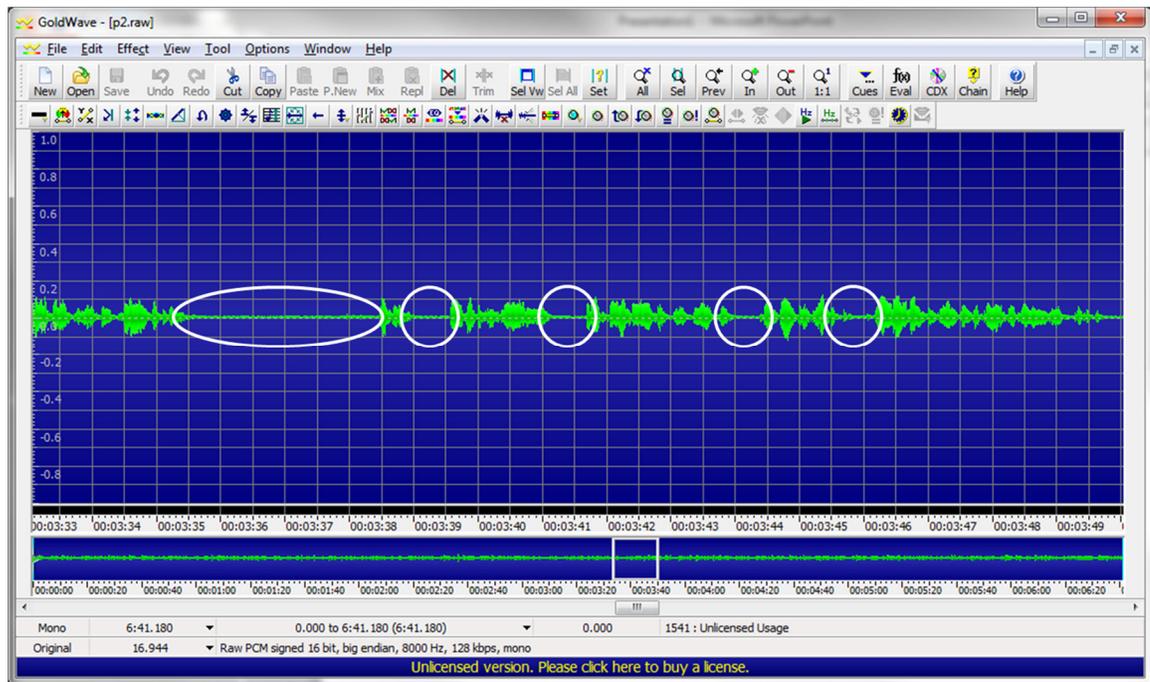
Tabla 5. Distancia recorrida cada 2 segundos (Zhao, 2012, p. 4)

Pasos c/2s	Desplazamiento Zancada (m)	Desplazamiento 2 s
0~2	estatura/5	zancada*pasos
2~3	estatura/4	zancada*pasos
3~4	estatura/3	zancada*pasos
4~5	estatura/2	zancada*pasos
5~6	estatura/1.2	zancada*pasos
6~8	estatura	zancada*pasos
>=8	1.2 x estatura	zancada*pasos

Realizar la estimación con esta segunda técnica tiene dos ventajas sobre la primera (GPS). En primer lugar esta que junto con la distancia también se obtiene la velocidad del recorrido. Además, al realizar la estimación cada dos segundos se obtiene un mayor detalle sobre las características del recorrido ya que se puede inferir si iba trotando, caminado lentamente o si hizo pausas durante el recorrido. Además al conocer este detalle también se puede calcular el gasto calórico del individuo derivado de la distancia recorrida (Zhao, 2012).

### 3.3.3. Detector de actividad de voz

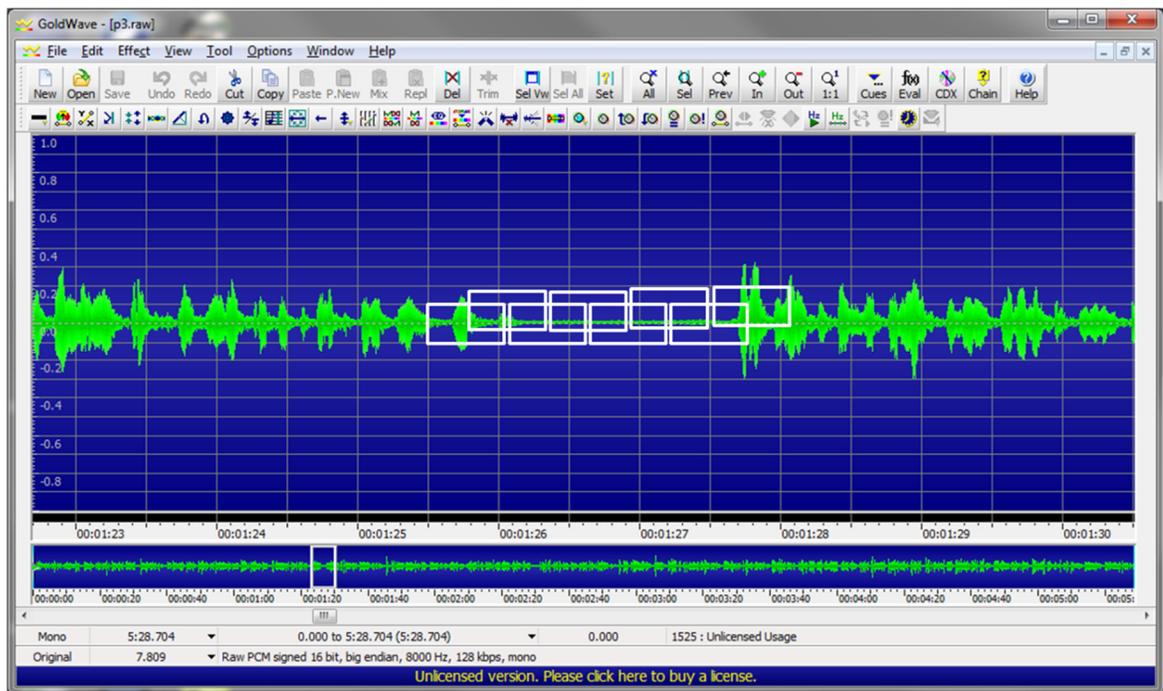
El problema de detección de actividad de voz (DAV) consiste en determinar la existencia de habla humana en una señal de audio por medio de técnicas de procesamiento de señales. En la Figura 13 se observa una señal de audio en la que se han enmarcado los periodos de silencio con los óvalos. El resto de la señal representa periodos donde existe voz humana, además de audio de otras fuentes. Inicialmente se podría pensar que el problema se resume a encontrar estos periodos de silencio, lo que equivale a conocer el volumen en la señal (magnitud de cada muestra de audio). Sin embargo, eso solo sería el inicio de la solución, ya que una vez descartados esos periodos de no voz, lo que sigue es procesar los periodos que si contienen información. Es decir, aquellos segmentos de audio donde se nota una oscilación en la amplitud de la señal y que no se sabe con certeza si contienen voz humana o cualquier otro tipo de información como música, ruido o sonido ambiental.



**Figura 13.** Señal de audio con los periodos de silencio marcados.

Este trabajo de tesis no pretende proponer una nueva técnica de DAV, sino que desde el inicio se buscó utilizar alguna técnica existente que sea fácilmente adaptable a un teléfono móvil. Por ello se buscó en literatura técnicas que se puedan implementar en un dispositivo móvil (ligeras).

Hay trabajo muy extenso en el tema, pero cabe destacar la investigación de (Miluzzo et al., 2008), en la que se realiza este tipo de detección (DAV) para determinar si existe una interacción social del usuario del teléfono con otro individuo (conversación). Sin embargo, la clasificación (voz y no-voz) no se hace en el dispositivo móvil, sino que se graba el audio y se genera un archivo para ser procesado posteriormente en un equipo de cómputo distinto al teléfono. Esto se debe a que el procesamiento de una señal de audio suele consumir muchos recursos computacionales y si no se tiene cuidado puede congelar el teléfono móvil.



**Figura 14.** Marcos con 50% de traslape sobre señal de audio.

Se hizo una revisión de literatura existente, especialmente en el tópico de DAV. El trabajo más cercano a lo que se buscaba es SoundSense (Lu et al., 2009). En este trabajo se entrena un algoritmo de clasificación para hacer la detección de actividad de voz. Inicialmente en SoundSense se propone que la señal de audio se divida en marcos y que estas tengan un traslape entre los marcos del 50% (vease Figura 14). Las características de la señal de audio que se usan para entrenar el algoritmo son nivel de energía en el tiempo (cálculo Raíz-Media-Cuadrado o RMC), centroide espectral, frecuencia fundamental (pitch), ancho de banda y entropía espectral calculados de la misma manera que se hizo en SoundSense (Lu et al., 2009).

La primera etapa de la creación del componente fue el diseño del clasificador y su entrenamiento. Un requisito importante del clasificador era que no consumiera muchos recursos de cómputo en el dispositivo móvil lo que de inmediato llevó la atención hacia clasificadores como árbol de decisión, redes neuronales o máquina de soporte de vectores (SVM).

De acuerdo a (Lu et al., 2009) la máquina de soporte de vectores ha mostrado un buen desempeño con anterioridad, además, es ideal para realizar la clasificación en tiempo real. La idea básica de la SVM es que se tienen una o varias clases de objetos o instancias agrupadas en clases y estas se encuentran divididas por una función de manera que cuando una instancia se evalúa en la función, esta automáticamente queda clasificada de acuerdo al resultado de la evaluación. Por su simplicidad al momento de clasificar este fue el algoritmo elegido para realizar la clasificación entre Voz y No-Voz.

Para generar un modelo de clasificación se requiere de un entrenamiento previo, por ello se realizó la recolección de datos (voz) para entrenar el algoritmo. Se reclutó a 12 personas: 6 hombres y 6 mujeres, con edades entre 20 y 50 años. Se solicitó a los participantes que leyeran un texto durante aproximadamente 5 minutos mientras eran grabados utilizando un teléfono móvil (Xperia Neo) colocado a una distancia de 90 centímetros de la cabeza de los participantes. Cada participante fue grabado por separado a excepción de una pareja (hombre y mujer) que se les pidió que leyeran el texto de forma alternada para simular un dialogo. La configuración del dispositivo de captura (micrófono) fue la siguiente: velocidad de muestreo de 8 kHz y codificación PCM a 16 bits.

Todos los participantes leyeron el texto en distintos ambientes con pocas interferencias de ruido ambiental. Adicional a los 11 archivos de audio, se grabaron 5 audios con sonido ambiental (Tabla 6). El propósito de esto, fue el de enriquecer el entrenamiento del algoritmo al proveer de información que proviene de un ambiente no controlado. La configuración en este caso fue la misma que con los individuos.

**Tabla 6. Listado de los archivos audios y participantes utilizados para generar el modelo de clasificación.**

Participante / Archivo	Descripción	Instancias	Etiqueta Voz	Etiqueta No-Voz
g1	Pareja de participantes simulando un diálogo (H yM).	1166	1107	59
p1	M	1272	1156	116
p2	M	1566	1212	354
p3	H	1283	1105	178
p4	H	1201	993	208
p5	H	1188	987	201
p6	M	1784	1346	438
p7	H	1480	1068	412
p8	M	1347	1138	209
p9	M	1549	1224	325
p10	H	1249	947	302
a1_lab	Audio ambiental grabado en interior de un laboratorio.	1297	1297	0
a2_calle	Audio ambiental grabado del ruido de calle.	1723	1723	0
a3_dpto	Audio ambiental grabado en el interior de una recamara de un departamento.	2787	2787	0
a4_cocina	Audio ambiental grabado en el interior de una cocina doméstica.	1690	1690	0
a5_tv-calle	Audio ambiental del interior de de un departamento con la T.V. encendida y la ventana abierta (ruidos de calle).	1068	1068	0

Posteriormente se segmentaron los 16 archivos en marcos de 4,096 muestras 0.512 segundos de longitud como se muestra en la Figura 14. Esto es porque para realizar el cálculo de algunas de las características usadas en el entrenamiento del algoritmo, es necesario que las instancias tengan una longitud de alguna potencia de 2. Estos marcos se convertirían luego en instancias para entrenar el algoritmo.

Cada marco se trató como una señal de audio independiente del resto y manualmente se etiquetó como Voz o No-Voz según correspondiera. Después se procedió a calcular el nivel de energía (basado en el volumen), entropía espectral, centroide espectral, ancho de banda y tasa de cruce por cero. Estos valores calculados se normalizaron para facilitar la etapa de entrenamiento: en cada característica se detectó el valor máximo de todos los marcos, enseguida al valor calculado para cada marco se dividió entre este máximo de tal manera que después de la normalización los valores iban de 0 a 1. De acuerdo a (Lu et al., 2009)

cada una de estas características contribuyen a diferenciar del resto un segmento de audio que contiene voz.

Para la clasificación de los datos se utilizó Weka (Hall et al., 2009), una colección de herramientas para realizar minería de datos, que incluye varios clasificadores. En particular se utilizó el clasificador SMO de Weka (Platt, 1999), que es una implementación de la máquina de soporte de vectores optimizada para reducir el tiempo de entrenamiento.

Una vez preparados los datos, lo siguiente fue medir la precisión del modelo de clasificación propuesto. Para ello se decidió utilizar el método de validación cruzada. En este método se toman las instancias de 15 archivos para generar un modelo y clasificar el resto de las instancias (1 archivo). Esto se hizo para cada uno de los archivos (alternadamente), de manera que se generaron 16 modelos diferentes con los que se clasificaron los 16 archivos por separado. La configuración para generar cada modelo se muestra en la Figura 15.

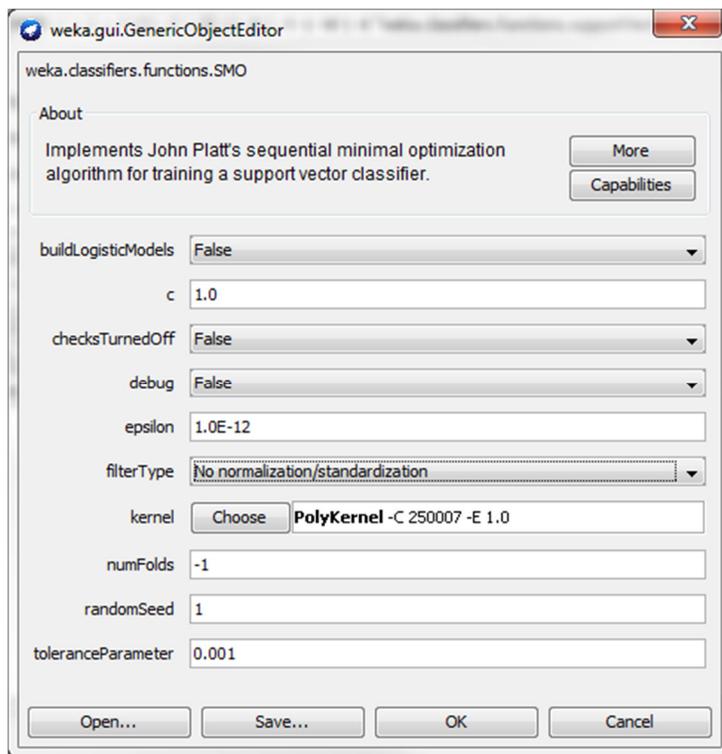


Figura 15. Configuración del entrenamiento del algoritmo SMO en Weka.

La matriz de confusión del modelo propuesto muestra que el 90% de las instancias fueron clasificadas correctamente (Tabla 7), por lo que utilizar este modelo resulta bastante viable y consume pocos recursos en comparación con otras alternativas como  $k$ -NN por ejemplo.

**Tabla 7. Matriz de confusión para el clasificador DAV. Las columnas son la clasificación otorgada con el modelo.**

a	b	
48.19%	3.74%	a = Voz
5.75%	42.31%	b = No-Voz

Una vez que la técnica mostró una precisión suficiente para los propósitos del presente trabajo de tesis se procedió a realizar un último modelo que incluyera todas las instancias de entrenamiento. Esto es porque partiendo de la primicia de que el modelo se genera a partir de estas instancias se intuye que el mismo se enriquecería incluyendo todos los datos de entrenamiento. Sin embargo esto no es necesariamente cierto, por ello se utilizó Weka para generar el modelo y medir su precisión. El resultado final es el siguiente modelo (función), que fue generado con Weka utilizando el total de las instancias:

$$3.3681 \times \text{entropía} + 18.6285 \times \text{centroide} - 26.1846 \times \text{anchoBanda} + 9.0636 \\ \times \text{tasaCruceCero} + 11.2475 \times \text{energía} - 4.3134$$

Al entrenar el algoritmo de clasificación utilizando todas las instancias (los 16 archivos) la precisión del clasificador se modificó, sin embargo sigue manteniéndose muy cercana al 90% inicial, por lo que se decidió continuar con este modelo para el diseño final del componente (véase Figura 16). La validación que hace Weka es la misma validación cruzada de la que se habló anteriormente, con la diferencia de que Weka no distingue a que archivo pertenece cada instancia y solo divide el total en 16 partes iguales (véase Figura 16).

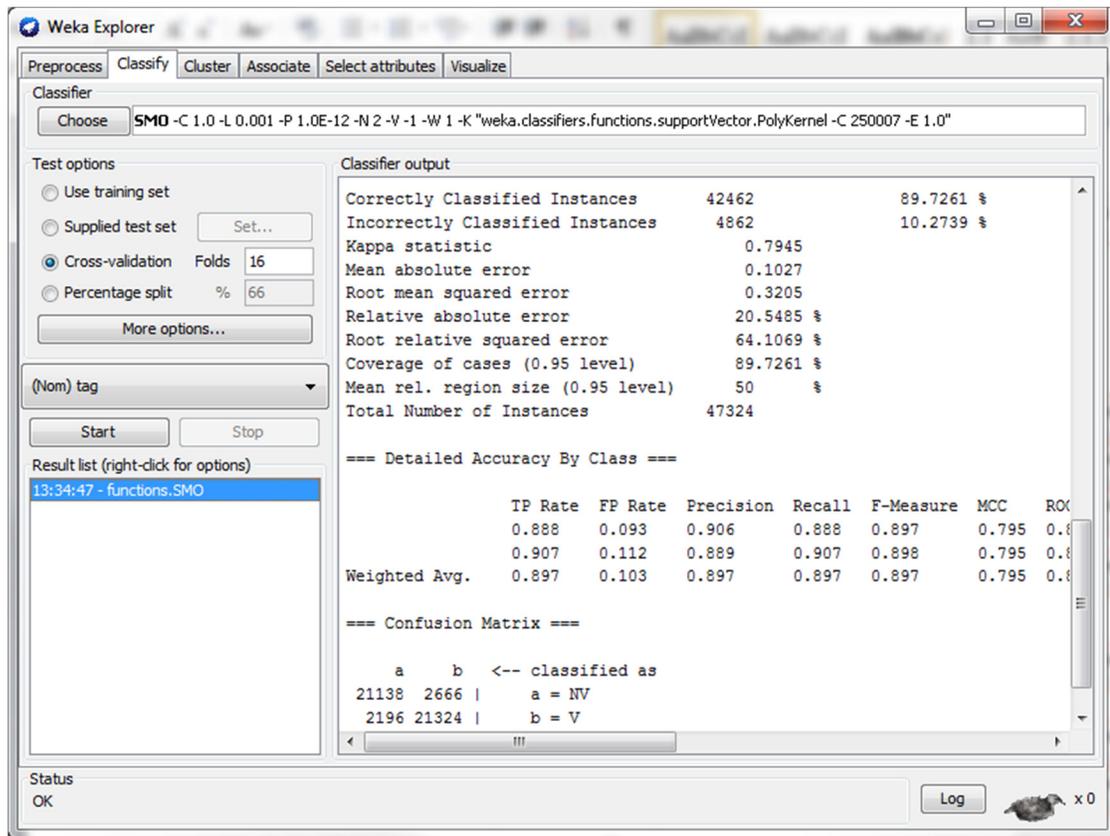


Figura 16. Captura de pantalla de la validación cruzada del modelo construido con Weka.

Por último, para crear el componente solo se implementó el clasificador (la función) en InCense (véase Figura 17).

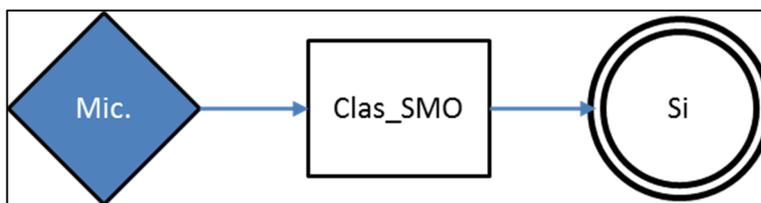


Figura 17. Componente DAV expresado con la terminología de InCense.

### 3.3.4. Proceso de creación de un nuevo componente

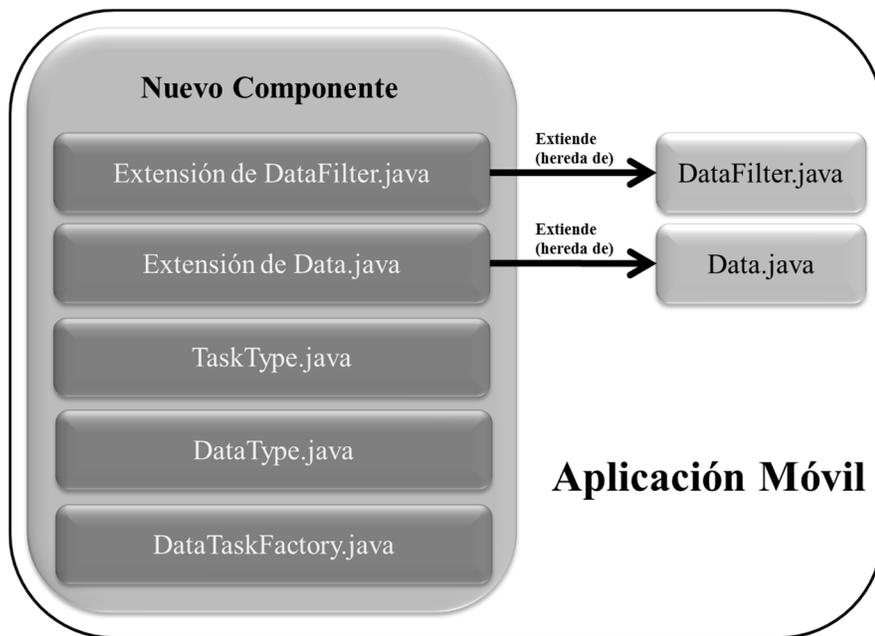
En (Pérez Gamboa, 2012) no existe de manera explícita documentación que detalle el proceso a seguir para crear un nuevo componente. Por ello, en esta sección se muestra dicho proceso utilizando solamente las herramientas disponibles en la arquitectura original.

Otro de los propósitos de esta sección es abrir la discusión para evidenciar la facilidad de crear un componente con la arquitectura extendida.

Si bien es cierto que existe un mecanismo para agregar componentes a InCense, también es cierto que este mecanismo requiere un alto nivel de habilidades técnicas, así como también un amplio conocimiento sobre el código fuente de InCense y su arquitectura original. Es por esto que el primer paso para alguien que quisiera implementar un nuevo componente consiste en familiarizarse con detalles técnicos de la implementación de la Aplicación Móvil. Cabe mencionar que este paso le tomó a un par de usuarios (una investigadora y un estudiante de maestría) con conocimientos de programación y habilidades técnicas considerablemente avanzadas, alrededor de un mes y la implementación de varios componentes para adquirir este conocimiento sobre la aplicación móvil.

Una vez que se ha adquirido este conocimiento sobre el código fuente y la arquitectura original de InCense, el siguiente paso es diseñar el componente a nivel conceptual, de manera similar a como se hizo con los tres componentes que anteriormente se mostraron en este documento.

Una vez que se tiene el diseño del componente, el usuario debe bajar hasta el nivel de código fuente. Un componente se encuentra distribuido en cinco clases que son parte de la implementación de la Aplicación Móvil: *TaskType.java*, *DataType.java*, *DataTaskFactory.java*, una clase que extiende *Data.java* y otra que extiende *DataFilter.java* (véase Figura 18). Las últimas dos clases son nombradas a criterio del usuario de acuerdo al nombre del componente. De manera que para implementar un componente es necesario editar estas 5 clases.



**Figura 18. Componente distribuido en 5 clases de código fuente.**

A continuación el usuario debe codificar el componente en la Aplicación Móvil. Todos los componentes en InCense heredan de la clase *DataFilter*, (véase Figura 18) la cual es una clase abstracta que implementa todos los métodos necesarios para el funcionamiento de los componentes (transmisión de datos, callbacks, etc.) con la excepción del método de recepción de datos *computeSingleData*.

El método *computeSingleData* es el responsable de procesar los datos que provienen de otro componente o de un sensor y es aquí donde se codifican los clasificadores si se requiere. Junto con esta modificación se debe agregar el nombre del componente (nombre de la clase que lo implementa) a la clase *TaskType.java*. Por ejemplo, en el caso del contador de pasos se agrega *Step\_Cntr* a la lista.

Lo siguiente es modificar la clase *DataTaskFactory*. El propósito de esta clase es el de generar instancias de los sensores o componentes necesarios para los distintos proyectos de sensado de acuerdo a la configuración de la Aplicación Móvil. Por lo tanto, la modificación extra en dicha clase tiene el objetivo de habilitar el uso del componente creado.

Todo componente tiene una salida de datos. Por ello se requiere un objeto que almacene estos datos. Para ello InCense cuenta con la clase *Data*. El usuario debe crear una nueva clase que extienda a *Data.java* (véase Figura 18) y crear las variables necesarias para almacenar los datos y los respectivos métodos y funciones de acceso. Por ejemplo, si se trata de un componente que procesa los datos del acelerómetro para clasificar la actividad del usuario, entonces se sabe que la salida será una cadena de texto (correr, saltar, reposo, etc.). Por ello, la clase que extienda a *Data.java* deberá tener una variable de tipo *String* con su setter y su getter<sup>10</sup>. El nombre de esta clase también debe agregarse a *DataType.java*.

Un elemento importante en la arquitectura de InCense es el Editor InCense, que permite crear nuevos proyectos de campañas de sensado. Como se mencionó anteriormente, la Aplicación Móvil y el Editor InCense deben estar sincronizados en cuanto a los componentes disponibles en el kit para ser utilizados en un proyecto. La razón es bastante simple: hasta este punto el nuevo componente se encuentra implementado en la Aplicación Móvil y se puede instanciar y utilizar en una campaña de sensado, pero, si en el Editor InCense no aparece, será imposible generar una configuración (proyecto) para la Aplicación Móvil que lo utilice.

Implementar un nuevo componente en el Editor InCense es bastante más difícil que en la Aplicación Móvil porque no hay mecanismos (ni siquiera de bajo nivel) que permitan la adición de componentes de forma estructurada. Todos están agregados como código duro en el Editor InCense, de manera que es fácil perder la contabilidad de los componentes que se agregan. Esto aunado al hecho de que agregar uno requiere de un alto nivel de habilidad técnica. Si se considera que InCense debe estar diseñado para usuarios con un nivel bajo de habilidad técnica en programación, entonces resulta casi imposible agregar un componente nuevo al Editor InCense. Como alternativa, se cuenta con un paquete en el código fuente de la Aplicación Móvil que se llama *test*. Para crear un nuevo proyecto de sensado que aproveche el nuevo componente, se necesita realizar modificaciones con un grado de dificultad similar al proceso de implementar un nuevo componente en la Aplicación Móvil.

---

<sup>10</sup> *Setter* y *getter* es el término para referirse al método y función que dan acceso a una variable de manera indirecta. Es una buena práctica de programación tenerlos para aquellas variables en una clase que representan atributos de acceso público.

### **3.3.5. Retos en el diseño y codificación de un nuevo componente**

Si bien es cierto que existe un mecanismo para crear componentes nuevos en InCense, este mecanismo no está documentado y, como se menciona en la sección anterior, tiene cierto grado de complejidad, lo que convierte el proceso de creación de un componente en una tarea muy difícil para una persona con limitados conocimientos técnicos y poca práctica incluso para personas con conocimientos de programación.

El mayor problema con que se encuentra el usuario es que es necesaria la edición del código fuente, lo que implica estudiar a fondo la arquitectura de InCense y su implementación para conocer qué clases editar y cómo hacerlo. Sin embargo, se debe recordar que InCense es un kit de investigación cuyo diseño esta hecho pensando en usuarios con bajo nivel de habilidad técnica. Dicho de otra forma, la tarea de agregar un nuevo componente a la plataforma de sensado es una tarea fuera del alcance de un usuario con poca habilidad técnica en programación. Incluso un usuario con experiencia en programación se encontraría con retos serios a sus habilidades. Sin embargo, es preciso notar que es posible que el programador conozca mejor la plataforma a medida que se agregan nuevos componentes, por lo que la tarea se puede hacer más sencilla. Pero de nuevo, el usuario para el que está diseñado InCense no es un programador con experiencia y crear un nuevo componente no es una tarea frecuente. De ahí que no repite la tarea con suficiente frecuencia y probablemente deba adquirir nuevamente parte del conocimiento necesario para realizarla.

Como se comentó, tanto la Aplicación Móvil como el Editor InCense deben estar sincronizados en cuanto a los componentes de los cuales puede disponer un usuario para utilizar en un nuevo proyecto de sensado, pero de acuerdo al diagrama arquitectónico de la sección 3.1 (véase Figura 3) no existe una comunicación clara entre el editor y la Aplicación Móvil. Esto puede generar situaciones difíciles de manejar para la Aplicación Móvil. Por ejemplo, si un usuario crea un nuevo proyecto con el editor y dicho proyecto incluye un componente en particular que no está implementado en la Aplicación Móvil, entonces cuando se asigne la nueva información de la campaña de sensado a la Aplicación Móvil, esta simplemente se cerrará debido a que no puede instanciar dicho componente.

Entonces, cuando un usuario decida crear un nuevo componente, este deberá realizar una doble implementación del componente: en la Aplicación Móvil y en el Editor InCense.

Otro reto que afrontar es llevar un registro de los componentes que se han implementado a lo largo del uso del kit. Después de algunos meses de usar InCense y agregar componentes, simplemente puede olvidarse su funcionamiento, parámetros, e incluso su existencia.

### **3.4. Creación de campañas de sensado**

Realizar una nueva campaña de sensado implica distribuir un nuevo archivo de configuración para la Aplicación Móvil, el cual indica qué sensores y componentes se usan, cuándo iniciar la captura de los datos, así como la relación entre ellos. Lo anterior, suponiendo que no se requiere un nuevo componente en la campaña, de lo contrario, también es necesario distribuir una nueva versión de la Aplicación Móvil que cuente con dicho componente implementado.

#### **3.4.1. Editor InCense**

Con el propósito de facilitar la creación de nuevos proyectos de sensado, se diseñó un Editor InCense que realiza exclusivamente esta tarea. Pero debido a algunos inconvenientes que se explicaron en la sección anterior, utilizar esta herramienta presenta un reto técnico importante en el caso de agregar un nuevo componente a la lista de componentes disponibles.

Asumiendo que no se requiere agregar un nuevo componente, el Editor InCense resulta de gran utilidad ya que tiene codificadas algunas de las validaciones básicas de las relaciones permitidas entre los componentes. Por ejemplo, un sensor no puede ser la entrada de otro sensor (véase Figura 19). Sin embargo todavía se considera que existen algunas áreas de oportunidad, por ejemplo sí se permite agregar dos sensores del mismo tipo como en la Figura 19. En este caso, cuando la configuración llegue a la Aplicación Móvil, esta intentará instanciar dos veces el mismo sensor, lo que ocasionaría un cierre imprevisto de la misma.

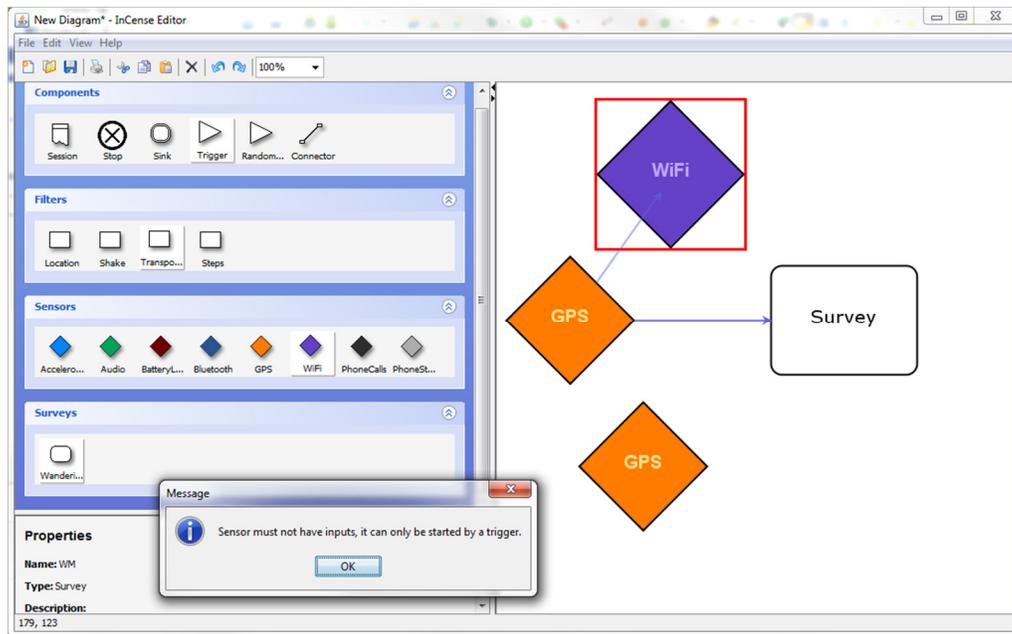


Figura 19. Imagen de la pantalla principal del Editor InCense

### 3.4.2. Archivo de configuración

La Aplicación Móvil requiere de un archivo que contenga la configuración necesaria para realizar la recolección de datos. Dentro de este archivo debe estar contenida información como los sensores y componentes que deben instanciarse además de la relación entre estos componentes para conocer el flujo que seguirán los datos recolectados. Este archivo debe colocarse en una carpeta específica en el teléfono móvil.

El formato elegido para darle estructura a la configuración de un nuevo proyecto es JavaScript Object Notation (JSON) (Pérez Gamboa, 2012). En el archivo de configuración, cada componente o sensor y su configuración está representado por un nodo en el archivo JSON. En la Figura 21 se muestra el archivo de configuración correspondiente al proyecto mostrado en la Figura 20. En la sección `tasks` del archivo de configuración se describe todos los elementos que forman el proyecto mientras que en la sección `relations` se muestra el flujo de datos del proyecto (véase Figura 21).

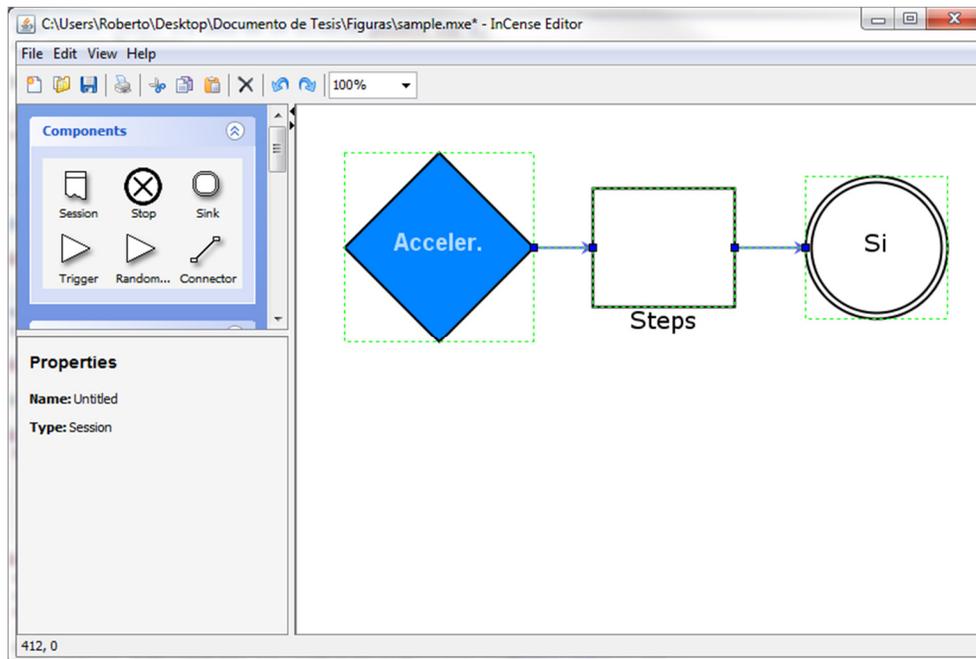


Figura 20. Ejemplo de un proyecto simple de sensado.

```

1 {"sessionsSize":1,
2  "surveysSize":0,
3  "sessions":{
4    "mainSession":{
5      "name":"mainSession",
6      "tasks":[
7        {"type":"Sensor","name":"Accelerometer","extras":{},"taskType":"Sensor"},
8        {"type":"StepsFilter","name":"Steps","extras":{},"taskType":"StepsFilter"},
9        {"type":"Sink","name":"Sink","extras":{},"taskType":"Sink"}],
10     "relations":[
11       {"task1":"Accelerometer","task2":"Steps"},
12       {"task1":"Steps","task2":"Sink"}],
13     "durationMeasure":null,
14     "durationUnits":0,
15     "autoTriggered":false,
16     "startDate":0,
17     "endDate":0,
18     "repeatMeasure":null,
19     "repeatUnits":0,
20     "repeat":false,
21     "notices":false,
22     "sessionType":null
23   }
24 },
25 "surveys":null}

```

Figura 21. Archivo de configuración de un nuevo proyecto.

### 3.5. Resumen del capítulo

En el campo de sensado con teléfonos móviles existe la oportunidad de contribuir con una plataforma de sensado que sea de propósito general, flexible y configurable. La plataforma de sensado InCense fue diseñada para cumplir con estas características. Sin embargo todavía existen aspectos de la misma que se pueden mejorar con el objetivo de facilitar el proceso de adición de componentes al reducir la exposición del usuario al código fuente de InCense. A continuación se listan las deficiencias mencionadas:

- Es necesaria una herramienta que permita al usuario conocer las capacidades de la plataforma y las características de los sensores y demás elementos de InCense.
- La adición de nuevos componentes a la plataforma de sensado requiere que el usuario modifique de manera directa código fuente de InCense.
- Cuando se crea un nuevo componente se debe implementar en la Aplicación Móvil y en el Editor InCense.

En el siguiente capítulo se desarrolla la propuesta para extender la arquitectura de InCense con el propósito de facilitar la adición de nuevos componentes a la plataforma de sensado.

## Capítulo 4

---

### InCense Extendido

En el capítulo anterior se mostró el proceso de creación de un componente así como el proceso que el usuario debe seguir para desarrollar una aplicación InCense y agregar nuevos filtros cuando se utilizan las herramientas existentes en la arquitectura original. El Capítulo 3 también mostró los pasos a seguir para desplegar una nueva campaña de sensado haciendo uso de la interfaz gráfica o, de su alternativa, directamente en código fuente. En ambos casos, se observó la oportunidad de rediseñar estos procesos teniendo en cuenta que el kit (InCense) debe resultar técnicamente accesible para los usuarios. Después de estudiar los procesos mencionados, se diseñó una propuesta para facilitar el uso de InCense. Y el resultado de este análisis es la propuesta de arquitectura extendida para el kit de investigación. Este capítulo presenta la arquitectura extendida haciendo una comparación con la arquitectura original.

#### 4.1. Proceso de creación de componentes con la arquitectura original

Del análisis hecho en el capítulo anterior se identificaron cinco actividades que el usuario debe realizar para crear un componente. Debido a que se hará una comparación entre la arquitectura original y la arquitectura extendida describimos brevemente estas actividades:

- a) **Estudiar arquitectura y código fuente.** Es en esta actividad donde el usuario adquiere conocimiento profundo sobre todo el kit de investigación (InCense): código fuente y arquitectura. Esta actividad requiere de un nivel alto de habilidad técnica en programación.
- b) **Diseñar nuevo componente.** Esta actividad implica realizar un diseño del componente a nivel conceptual. Sin embargo esta actividad requiere conocimiento previo del kit de investigación y sus capacidades para diseñar un componente que sea viable de implementar en InCense. Adicionalmente se requiere que el usuario tenga conocimiento sobre el dominio de aplicación del componente a diseñar.
- c) **Implementar el componente en la Aplicación Móvil.** Esta actividad requiere que el usuario edite el código fuente de la Aplicación Móvil. Por ello es indispensable un alto nivel de habilidad técnica en desarrollo de software, especialmente sobre la plataforma Android.

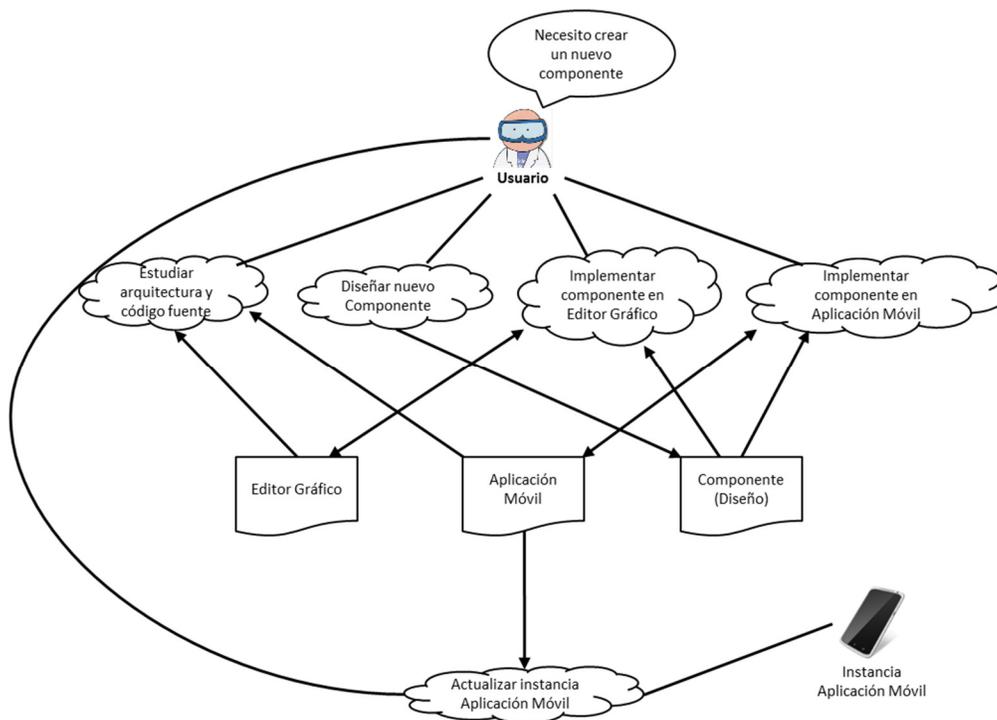
- d) **Implementar el componente utilizando la interfaz gráfica.** Actividad que implica editar el código fuente del Editor InCense, por lo que es necesario que el usuario tenga un conocimiento profundo sobre el código fuente del mismo Editor InCense.
- e) **Actualizar instancia de Aplicación Móvil.** Después de implementar el componente en la Aplicación Móvil es necesario realizar una actualización de la instancia que ejecutan los distintos teléfonos inteligentes que se encuentran participando en una campaña de sensado. Esta actividad no requiere de habilidades técnicas avanzadas.

La Figura 22 muestra una gráfica rica<sup>11</sup> que modela el proceso en cuestión. En una gráfica rica se pueden encontrar al menos tres tipos de elementos: roles, artefactos y actividades. Los roles y actividades se unen por medio de líneas sólidas que indican un involucramiento directo por parte del rol en determinada actividad. Por otro lado los artefactos son entradas o salidas de las actividades y se unen con las actividades por medio de flechas que indican si son salida, entrada o ambas.

En el proceso de creación de un nuevo componente con la arquitectura original se puede observar claramente como el usuario tiene participación directa en todas las actividades. Incluso aquellas que demandan un alto nivel de habilidad técnica por parte del usuario a pesar de que el diseño de InCense debería ser accesible a usuarios con bajo nivel de habilidad técnica (véase Figura 22).

---

<sup>11</sup> Diagrama utilizado para modelar un proceso desde la perspectiva de un individuo que participa en el mismo (Checkland, 2000).



**Figura 22. Gráfica rica del proceso de creación de un nuevo componente.**

Del análisis del proceso resulta evidente la necesidad de evitarle al usuario la dificultad de estudiar el código fuente y la implementación del componente de manera directa sobre el mismo código fuente. Con esto se elimina la necesidad de que el usuario posea un alto nivel de habilidades técnicas. Una estrategia para lograr esta abstracción fue la de automatizar las actividades de implementación de código. La producción automatizada de un modelo de automóvil requiere que todas las unidades sean esencialmente iguales (carrocería, motor, etc.) salvo por algunas características como pintura. De la misma manera que se producen los autos en serie, cuando se automatiza la creación de componentes se debe determinar que partes de los mismos serán iguales en todos los componentes y cuáles serán las que puedan ser personalizadas. La siguiente sección propone la estructura de un componente genérico que sirva de base para la automatización de la creación de nuevos componentes.

## 4.2. Componente genérico

Un análisis inicial del proceso de creación de componentes con la arquitectura original (véase Figura 22) muestra que el usuario está involucrado en todas las actividades del proceso de forma directa. Algunas de estas actividades obligan al usuario a exponerse al código fuente de la Aplicación Móvil y de la interfaz gráfica. De ahí que sea necesario agregar una capa de abstracción más en el proceso que facilite al usuario crear los componentes.

El trabajo del Capítulo 3 sirvió también para analizar la estructura de los componentes y en base a este análisis encontrar una estructura genérica, en forma de plantilla, que permita definir un nuevo componente. Después de haber implementado varios componentes, se observó que son fundamentalmente iguales entre sí y que sólo se diferencian en 3 aspectos: los nombres de los componentes, las variables que se utilizan y en el código para procesar los datos.

Como se explicó en el Capítulo 3, si se observa a nivel de código fuente, un componente está formado por varias clases distribuidas entre la Aplicación Móvil y el Editor InCense. Esta perspectiva puede ser difícil de comprender para un usuario con bajo nivel de habilidad técnica, de ahí que fue necesario realizar una abstracción de un componente (véase Figura 23).



Figura 23. Diagrama de componente genérico.

Como se puede apreciar de la Figura 23, un componente se forma de código para procesar datos, un nombre del componente, y variables de diferentes tipos. Esto implica que un usuario podría generar nuevos componentes al proveer estos tres elementos. Se podría decir que la estructura propuesta es un modelo a partir del cual se generan nuevas instancias de componentes.

Este nuevo escenario donde un componente es un objeto y no un conjunto de clases (código fuente), pone al problema de crear un nuevo componente en un nivel de abstracción más elevado y a la vez simplifica las cosas desde el punto de vista del usuario. Otra ventaja que viene con esta nueva perspectiva de un componente es la portabilidad. Es decir, se abre la posibilidad de que los usuarios compartan sus componentes ya que independientemente de cómo se haga la implementación de esta estructura siempre y cuando se puedan transferir nombre, código fuente, y variables entre los usuarios, se podrá replicar un componente determinado.

A continuación, se describen cada uno de los elementos que conforman un componente:

Nombre del componente. Un componente se forma con un conjunto de clases (código fuente en java). Cada una con una función específica. Por ejemplo una que implementa el procesamiento de los datos, otra que implementa el almacenamiento de los datos procesados y otras más. Estas clases y el componente en general requieren de un identificador único, para lo cual se usa el nombre base del componente para generar un nombre convencional específico para cada clase. Por ejemplo, la clase que implementa el procesamiento de los datos puede llamarse <Nombre>ProcessData donde Nombre se reemplaza por el nombre base del componente. Cabe mencionar que el nombre debe seguir la convención de Java para designar una clase. De esta manera pueden nombrarse automáticamente las clases necesarias para implementar el componente utilizando Nombre como base.

Código de procesamiento de datos. Este elemento de un componente es el código fuente que se encarga de procesar los datos. Debe recordarse que InCense está escrito en Android<sup>12</sup> por lo que este código debe estar escrito en Android también. El usuario es responsable de proveer este código fuente (solo esta sección), pero debe seguir ciertas convenciones que a continuación se listan:

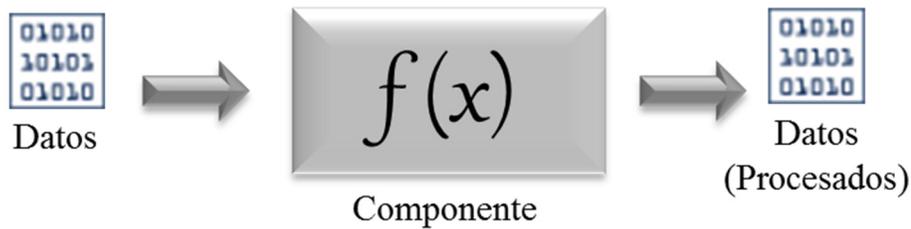
- Se debe conocer de antemano el tipo de paquete de datos que se recibe y el nombre de los atributos para acceder a los datos (variables).
- El usuario debe utilizar Android para crear este código fuente.
- El código fuente creado por el usuario debe estar escrito como funciones y métodos de acuerdo al diseño del usuario.
- Siempre deberá crear una función que se llame `processData`. Esta función será el punto de entrada para el procesamiento de datos y recibe como argumento un paquete de datos.
- Al final, la función `processData` debe regresar los datos procesados por medio de la sentencia `return`. Los datos procesados deben estar contenidos en una clase de tipo `Data` o en su defecto hacer una invocación a dicha clase.

El usuario también es el responsable de verificar la sintaxis del código. Junto con el componente se genera también una clase de tipo `<Nombre>Data`. El propósito de esta clase es la de generar objetos para almacenar los datos procesados en el componente y transmitirlos al siguiente componente por medio de la sentencia `return` en `processData`.

Esta es posiblemente la parte más complicada de un componente, pero debe considerarse que la alternativa para el usuario es averiguar en qué clases (de la Aplicación Móvil y el Editor InCense) debe introducir esta parte de código para procesar los datos. En contraste, con la abstracción propuesta en esta sección, el usuario solo debe ocuparse del procesamiento de los datos y realizar su diseño del componente (véase Figura 24).

---

<sup>12</sup> Android además de ser un sistema operativo para teléfonos inteligentes, también es un lenguaje de programación precisamente para hacer aplicaciones que corren en el sistema operativo del mismo nombre. Está basado en el lenguaje Java.



**Figura 24. Diseño conceptual de un componente.**

Variables. Un componente puede necesitar variables de ámbito global para el procesamiento de datos, debido a que con la arquitectura extendida no tiene acceso al resto del código de InCense y solo puede intervenir en el procesamiento de datos, se requiere un mecanismo que le permita declarar este tipo de variables. Las variables pueden ser de cuatro tipos: configuración, globales, estáticas, y salidas. Las variables de configuración tienen el propósito de establecer valores de operación de los componentes, por ejemplo el nivel de energía para contabilizar un paso en un podómetro, estas son establecidas desde la etapa de diseño del componente. Luego están las variables globales, que son auxiliares en el código fuente, que en caso de que el usuario decida crear más de una función en el procesamiento de datos, estas son accesibles desde cualquier parte de su código. Las estáticas tienen el propósito de almacenar valores constantes, como  $\pi$  (pi) o alguna otra constante definida por el usuario. Por último, las variables de salida sirven para almacenar los datos que han sido procesados, sin embargo, estas variables serán declaradas como atributos del paquete de datos del componente.

El componente conceptualizado representa una capa de abstracción más para el proceso de creación de componentes con lo que se disminuye el nivel de exposición directa del usuario al código fuente de InCense. Sin embargo, es necesario generar el resto del código fuente para el componente (completo, como se describió en el Capítulo 3). Esto supone un reto técnico importante, ya que el resto del proceso debe ser transparente para el usuario. Es decir, una vez que el usuario haya proporcionado los elementos necesarios para generar el componente (código de procesamiento de datos, nombre y variables), el resto del proceso debe ser automático, evitando así intervención innecesaria por parte del usuario, o en su defecto bajar el nivel técnico requerido para realizar el resto del proceso.

InCense utiliza Eclipse<sup>13</sup> como plataforma de desarrollo. Haciendo una revisión de las capacidades del Ambiente Integrado de Desarrollo (IDE, por sus siglas en inglés) de Eclipse, se detectaron dos características que para la propuesta de arquitectura extendida resultan de trascendencia:

1. La funcionalidad del IDE de Eclipse se puede extender mediante la adición de plug-ins.
2. Existe un proyecto en Eclipse que busca agregar capacidades de modelado de software al IDE de Eclipse: The Eclipse Modeling Framework (EMF). Como parte de este proyecto se encuentra Java Emitter Templates (JET). JET es una tecnología de que permite generar código fuente en tiempo de ejecución por medio de plantillas predefinidas.

La posibilidad de extender las capacidades de Eclipse y además generar código fuente en tiempo de ejecución son las tecnologías habilitadoras para la implementación de una capa de abstracción en el proceso de creación de componentes. En la siguiente sección se detalla la propuesta e implementación de la arquitectura extendida.

### **4.3. Arquitectura extendida**

Recordando que el objetivo general del presente trabajo de tesis es “facilitar la implementación de nuevos componentes” en esta sección se detalla la propuesta de una extensión a la arquitectura de InCense para lograr dicho objetivo. Primero, se identifican los obstáculos existentes en el kit de investigación (InCense) que impiden que un usuario con bajo nivel de habilidad técnica pueda crear nuevos componentes con facilidad. Entonces, se detallan las mejoras propuestas para la arquitectura original de InCense.

El trabajo realizado en el Capítulo 3 sirvió también para documentar el proceso de creación de componentes con la arquitectura original. La Figura 22 es una gráfica rica de este proceso. Idealmente, el usuario solo debería realizar la actividad de “diseñar un nuevo componente”, pero como se muestra en la Figura 22 (el proceso de creación de un nuevo componente) el usuario tiene la responsabilidad directa de realizar otras tareas más

---

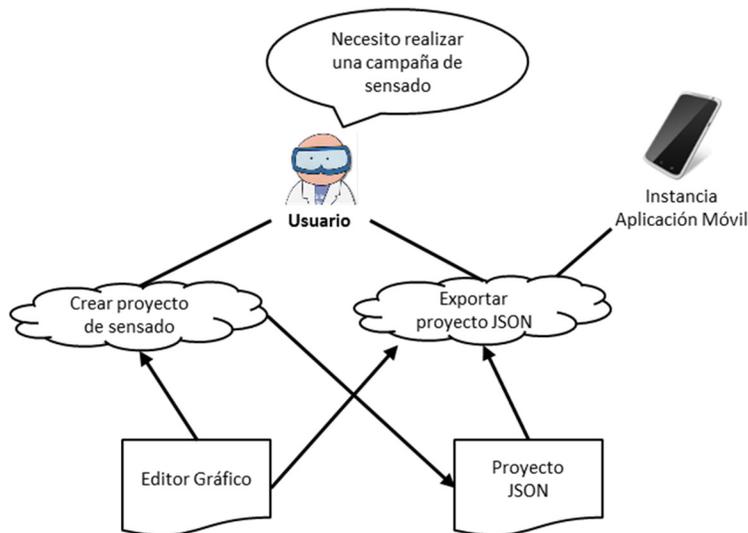
<sup>13</sup> Eclipse es un ambiente de desarrollo de software que soporta varios lenguajes, entre los que se encuentran Java y Android. También soporta la extensión del ambiente por medio de la adición de plug-ins.

técnicas. Y son estas tareas las que se considera que deben ser asistidas por una herramienta de software.

También se detectaron obstáculos que en general dificultan el proceso de creación de componentes para un usuario inexperto en programación. Estos obstáculos son:

- El usuario debe estudiar a fondo la arquitectura e implementación de InCense para entender las capacidades de sensores, componentes o cualquier otro elemento del kit.
- La creación de nuevos componentes y campañas de sensado que incluyan estos componentes requiere de un alto nivel de habilidad como desarrollador de software.
- El usuario debe llevar la contabilidad de los componentes que ha agregado y mantener en sincronía la Aplicación Móvil y el Editor InCense.

Además de estudiar el proceso de creación de componentes, se analizó el proceso de creación de proyectos de sensado (véase Figura 25) ya que ambos están ligados: cuando se crea un nuevo componente se agrega a la Aplicación Móvil y al Editor InCense.



**Figura 25. Proceso de creación de proyectos de sensado.**

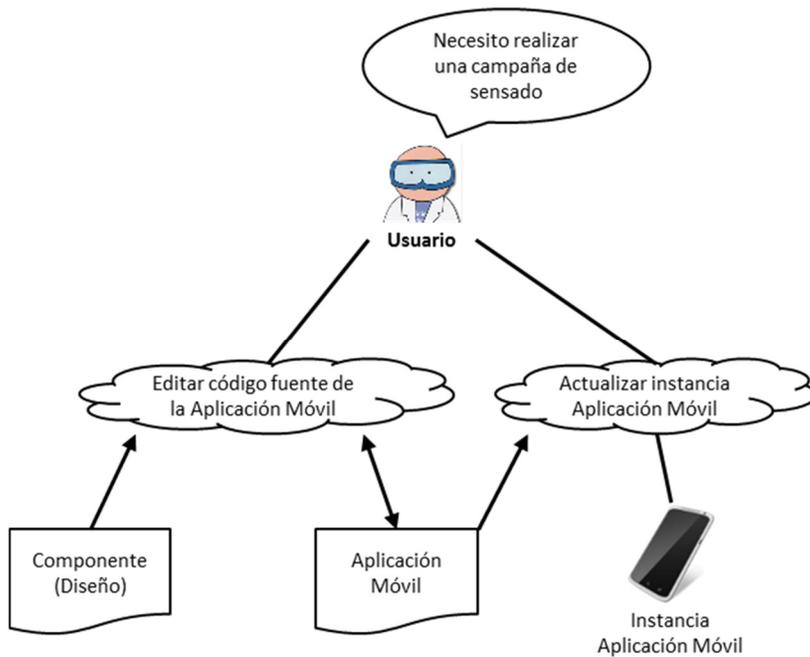
Idealmente, en la creación de proyectos de sensado solo se realizan 2 actividades que toman como entrada 2 artefactos, el Editor InCense y el proyecto en formato JSON. Además del usuario, también se encuentra el dispositivo móvil que representa un rol (la instancia de la

Aplicación Móvil corriendo en el teléfono inteligente). A continuación se describen brevemente las actividades.

Crear proyecto de sensado. El usuario abre el Editor InCense y del menú principal selecciona crear nuevo proyecto. Luego el usuario arrastra los sensores y componentes disponibles necesarios para construir un proyecto de sensado de acuerdo a la notación de InCense y las características del proyecto. El usuario debe almacenar el proyecto.

Exportar proyecto JSON. Una vez que se encuentra en pantalla el proyecto construido previamente, el usuario selecciona del menú principal la opción exportar y luego indica el directorio de la computadora para guardar el archivo JSON. Hecho esto, el usuario debe copiar el archivo en el directorio principal de la Aplicación Móvil que se encuentra en ejecución en el teléfono inteligente, eventualmente la Aplicación Móvil reiniciará con la nueva configuración o en su defecto el usuario puede iniciar / reiniciar una nueva sesión de sensado en la Aplicación Móvil.

Este es el proceso que idealmente se debe seguir para crear un nuevo proyecto de sensado y desplegarlo en uno o varios teléfonos móviles, pero no siempre es así. Agregar un nuevo componente al Editor InCense (véase Figura 22) es una actividad que como se mencionó anteriormente es compleja y el usuario suele evitarla. La Aplicación Móvil tiene un conjunto de clases agrupadas en un folder en el código fuente que permiten crear nuevas campañas de sensado. El único propósito de haber agregado este paquete fue el de crear campañas de sensado para realizar pruebas sobre la Aplicación Móvil durante la implementación de la misma y sin el Editor InCense puesto que también estaba en implementación. Después de haber hecho las pruebas de funcionalidad sobre la Aplicación Móvil, estas clases para realizar pruebas no fueron eliminadas. Esta herramienta es más sencilla de usar si se desea crear nuevas campañas de sensado con los componentes que se encuentran implementados en la Aplicación Móvil. De ahí que se generó un proceso alternativo de creación de campañas de sensado. La Figura 26 muestra la gráfica rica de este proceso alternativo en sustitución del que se muestra en la Figura 25.



**Figura 26. Proceso alternativo de creación de campaña de sensado.**

En el proceso alternativo mostrado en la Figura 26 hay dos actividades principales:

Editar código fuente en la Aplicación Móvil. En esta actividad se edita la clase `Project.java` que se encuentra en el paquete `test` en la Aplicación Móvil. Para utilizar un nuevo componente que se agregó recientemente a la plataforma, es necesario tener en cuenta el diseño del mismo. Cada método de esta clase es un proyecto de sensado distinto y en cierto punto del código, el usuario indica el proyecto a iniciar mediante una llamada al método deseado. Dado que cada método es un proyecto de sensado distinto, el tamaño de esta clase puede crecer demasiado, lo que dificulta llevar la contabilidad de los proyectos de sensado que se tienen y su mantenimiento o edición.

Actualizar instancia de la Aplicación Móvil. Debido a que el nuevo proyecto de sensado se crea en la misma Aplicación Móvil, es necesario que con cada nuevo proyecto, el usuario actualice las instancias de la Aplicación Móvil en todos los teléfonos inteligentes que participarán en el proyecto.

Refiriéndose únicamente a la creación de proyectos de sensado, las ventajas de utilizar el Editor InCense sobre el paquete `test` son claras; paradójicamente el usuario opta por utilizar

esta última alternativa para evitar la implementación del componente en el Editor InCense (véase Figura 22).

A las áreas de mejora que se mencionaron antes, también debe agregarse que:

- Al usuario se le dificulta implementar componentes en el Editor InCense, lo que lo obliga a utilizar la alternativa de la Aplicación Móvil para crear proyectos de sensado: el paquete test.

En las siguientes líneas se hace una propuesta para extender la arquitectura de InCense. Esta propuesta aprovecha las áreas de mejora que se detectaron y realiza las modificaciones necesarias a la arquitectura original, resultando en la arquitectura extendida. Por la naturaleza de las áreas de oportunidad detectadas se considera que realizando estas mejoras, se facilitará el proceso de creación de componentes y nuevos proyectos de sensado. Logrando con ello el objetivo principal del presente trabajo de tesis.

#### **4.3.1. OntoInCense**

Una conceptualización es una visión abstracta y simplificada del mundo que se quiere representar y una ontología es la conceptualización de un dominio de conocimiento en específico (Gruber, 1995).

Una de las dificultades con las que se encuentra un individuo al utilizar InCense por primera vez, es precisamente estudiar su arquitectura y código fuente. En (Rodríguez et al., 2012) se propone el diseño de una ontología para modelar el dominio de conocimiento de InCense, con el objetivo de reducir la intervención del usuario en la creación de componentes: OntoInCense.

Si una ontología es una visión simplificada de un dominio de conocimiento en específico, con OntoInCense se simplifica también el dominio de InCense, de ahí que es más fácil para el usuario conocer o familiarizarse con la plataforma. La actividad “Estudiar arquitectura y código fuente” de la Figura 22 (proceso de creación de componentes) se simplifica de manera parcial con la propuesta de OntoInCense. Sin embargo, aún es necesario un mecanismo para reducir la exposición directa del usuario al código fuente de InCense.

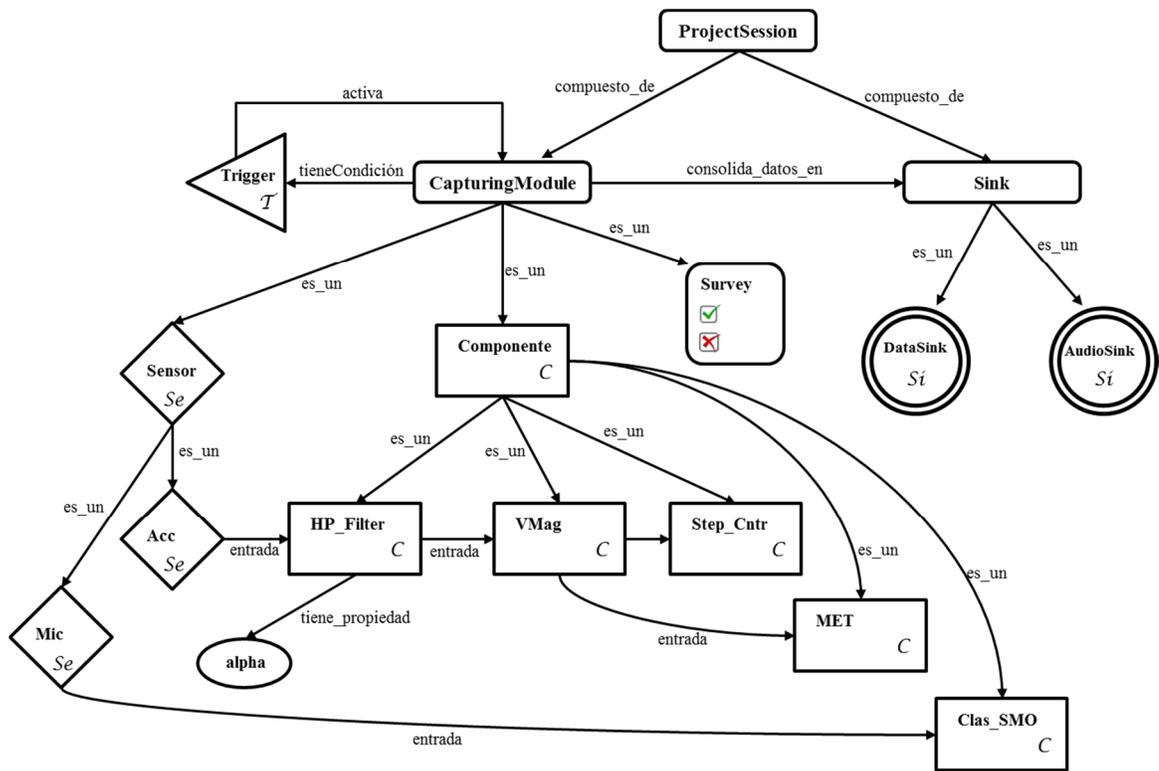


Figura 27. Representación gráfica de OntoInCense, con algunos componentes (C) de ejemplo.

En la Figura 27, se muestra el conjunto de objetos que forma el dominio de conocimiento de InCense y la relación existente entre ellos; en otras palabras se trata de la representación gráfica de OntoInCense. A continuación se documenta el diseño de la ontología y sus elementos principales.

*ProjectSession*. Es la raíz de la ontología, el objeto principal y representa un proyecto de sensado. Se compone de *CapturingModule* y *Sink*.

*CapturingModule*. Es un módulo de captura, una forma generalizada de sensor, componente y *survey*. Una analogía adecuada sería la herencia de clases en programación. En cierto modo *CapturingModule* es padre de sensor, componente y *survey*.

*Trigger*. Este elemento toma una condición de entrada que en caso de ser cierta inicia un módulo de captura (sensor, *survey* o componente). La entrada de este elemento suele ser la salida de un módulo de captura.

*Sink*. Parte de *ProjectSession*, puede ser de dos tipos: de datos y de audio. El primero es para consolidar o almacenar las salidas de datos un sensor o componente mientras que el segundo es exclusivamente para almacenar los datos sensados como un archivo de audio crudo. El *survey* es un módulo de captura que almacena directamente sin necesidad de usar un *sink*.

*Sensor*. Este elemento representa a un sensor genérico, hereda de *CapturingModule*; a su vez sensor es padre del resto de los sensores implementados en la plataforma (*InCense*) y que están conceptualizados en la ontología: acelerómetro, GPS, micrófono y otros.

*Survey*. Este es el único elemento que habilita a la plataforma para realizar sensado participativo de manera explícita. El usuario puede crear cuestionarios al extender la ontología generando hijos de *survey*.

Componente. Este elemento de *OntoInCense* representa un componente genérico. Cada vez que el usuario desee agregar un componente a la plataforma (*InCense*) debe extender este elemento como se muestra en la Figura 27 donde ya están agregados los componentes que diseñaron en el capítulo 3 (*HP\_Filter*, *VMag*, y *Step\_Cntr*). A la vez, es a este nivel donde se define la relación entre cada componente. Por ejemplo, en la Figura 27 se puede apreciar como *VMag* sirve como entrada de *Step\_Cntr* y *MET*. También vale la pena notar que *HP\_Filter* tiene asociado un atributo: *alpha*. Si el componente requiere de alguna variable de configuración esta se define en *OntoInCense*.

Existen varias plataformas para la creación y edición de ontologías y bases de conocimiento, entre ellas destaca *Protégé* (“*Protégé*,” 2012) por ser libre y de código abierto además de contar con el soporte de una institución prestigiada en la comunidad científica, como es la Universidad de Stanford. Además, existe un plug-in que permite integrar *Protégé* directamente en el ambiente de desarrollo *Eclipse*, lo que resulta muy conveniente tomando en cuenta que es la plataforma de desarrollo de *InCense*. Tomando en cuenta lo anterior, se decidió implementar *OntoInCense* con *Protégé*.

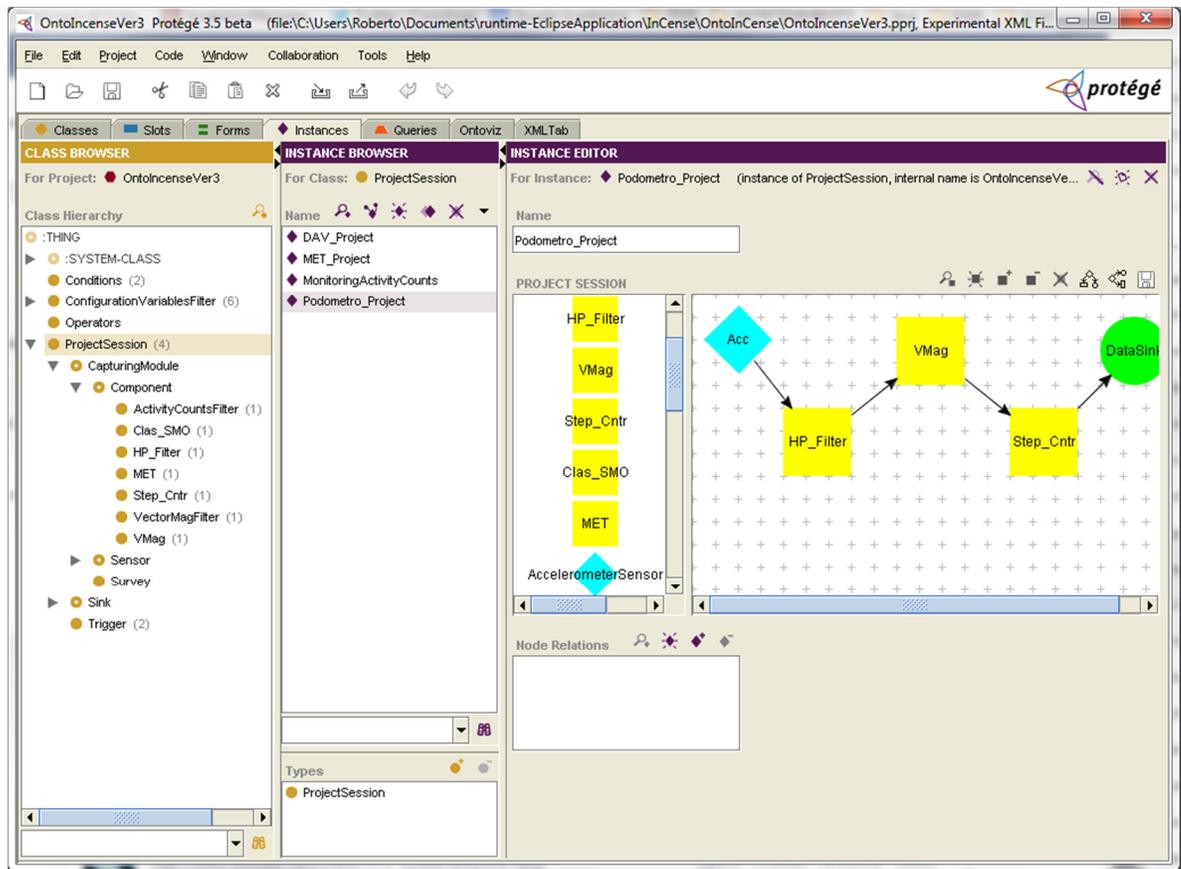
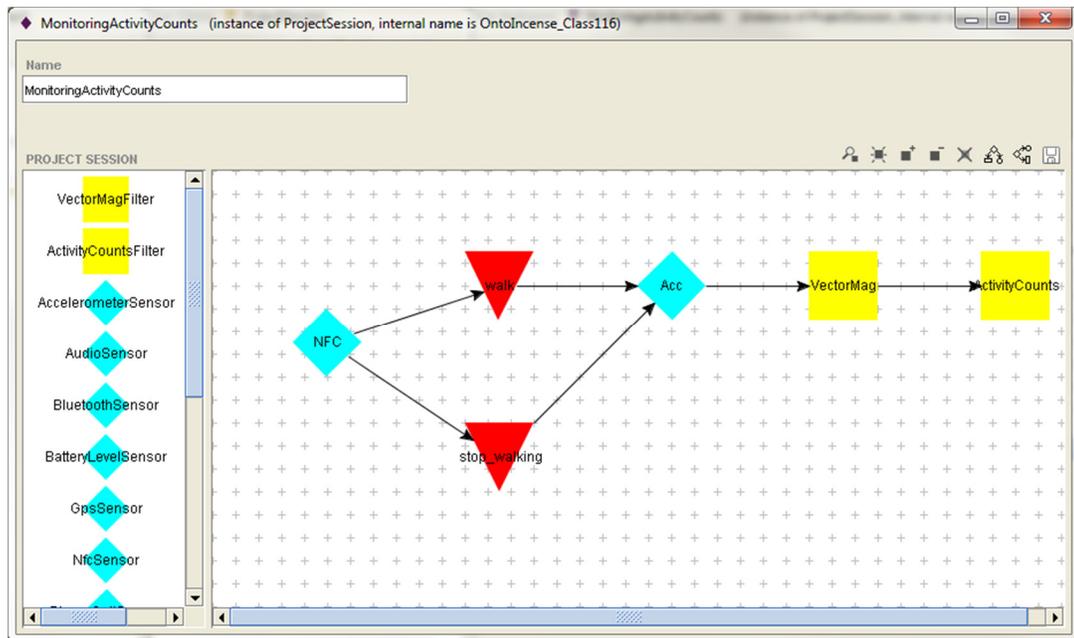


Figura 28. Captura de pantalla de OntoInCense en Protégé.

En la Figura 28 se observa en la columna de la izquierda la implementación de OntoInCense. Al centro se encuentran las instancias de ProjectSession (proyectos de sensado) y a la derecha la representación gráfica de cada proyecto de sensado. En el explorador de clases (class browser, Figura 28) se aprecia como ProjectSession contiene todos los elementos del diseño de la ontología (véase Figura 27) más las variables de configuración. En la implementación fue necesario agregar algunos elementos como *conditions* y *operators*; estos nuevos elementos fueron agregados (fuera de ProjectSession) para facilitar la adición de las restricciones inherentes a cada elemento de InCense. Por ejemplo, en un *Trigger* solamente es posible utilizar ciertos operadores, por lo que fue necesario agregar una clase adicional que contiene una lista con los operadores válidos en InCense.

También se puede observar que Protégé tiene algunas restricciones en cuanto a la representación gráfica de los elementos de la ontología. Cada una de las clases se puede representar con una forma geométrica determinada: cuadrado, rombo, círculo, elipse, triángulo, o hexágono. Por ello, fue necesario asignar nuevas formas a los elementos existentes en InCense: componente, *trigger*, *sink*, sensor, y *survey*. Cabe mencionar que esta asignación es temporal, por lo que no implica un cambio definitivo en la notación gráfica de InCense y solo es válida para el trabajo de la ontología. En OntoInCense, los componentes se muestran como cuadrados color amarillo y los sensores como un rombo azul. Un *sink* está simbolizado con un círculo verde, mientras que un *survey* se representa con un cuadrado morado, por último un *trigger* se muestra como un triángulo rojo. En la Figura 28 se observa un diagrama que representa un proyecto de sensado, este diagrama es equivalente al podómetro presentado en el Capítulo 3.

Protégé incluye un editor gráfico que permite la creación de instancias de un diseño de ontología. La Figura 29 muestra un proyecto de sensado creado a partir de OntoInCense. En esta Figura se puede observar como del lado izquierdo se encuentran los sensores, componentes y demás elementos que existen en la arquitectura de InCense. Estos elementos están disponibles para agregar en el espacio del diagrama y conectarlos entre sí de manera muy similar a como se hace con el Editor InCense de la arquitectura original.



**Figura 29.** Instancia de OntoInCense en el editor gráfico.

Anteriormente se mencionó que unos de los obstáculos con los que se encontraba el usuario al crear un nuevo componente eran:

- Implementar el componente en el Editor InCense (arquitectura original)
- Tener actualizados la Aplicación Móvil y el Editor InCense (arquitectura original)

Estos obstáculos se presentan debido a que en la arquitectura original no existe una conexión clara entre la Aplicación Móvil y el Editor InCense, en cierta forma funcionan como partes independientes una de la otra. Por ejemplo puede suceder que el componente este implementado en la Aplicación Móvil pero no en el Editor InCense lo que genera problemas en el despliegue de campañas de sensado. De ahí que el proceso de creación de proyectos de sensado depende del proceso de creación de componentes (véase Figuras 22, 25 y 26).

Considerando lo anterior, se propone reemplazar el Editor InCense de la arquitectura original de InCense con el editor gráfico de Protégé. De esta manera OntoInCense pasa a ser el vínculo necesario entre el proceso de creación de componentes y el proceso de creación de proyectos de sensado, eliminando la responsabilidad del usuario de

implementar el componente en el Editor InCense (véase Figuras 25 y 26). Por otro lado, con la propuesta del componente genérico la creación de componentes se simplifica al reducir el nivel de exposición que tiene el usuario hacia el código fuente ya que la implementación del elemento componente en OntoInCense se hizo tomando en cuenta el componente genérico.

Aunque ya no es necesario implementar el componente en el Editor InCense de la arquitectura original, todavía es necesario que se genere el código fuente del componente en la Aplicación Móvil. Si bien es cierto que cuando el usuario diseña el componente también proporciona el código necesario para procesar los datos, debe recordarse que la implementación de un componente en la Aplicación Móvil está distribuida entre 5 clases de código fuente. Esto quiere decir que aún es necesario generar el complemento del código proporcionado por el usuario.

#### **4.3.2. Generar código fuente con Java Emitter Templates (JET)**

A nivel de código fuente, un componente está distribuido en cinco clases y el usuario solamente proporciona una fracción del código necesario por lo que se requiere generar de manera automática el resto del código fuente.

Hasta donde se tiene conocimiento, solo existe una tecnología para Java y soportada por eclipse que permita generar código fuente en tiempo de ejecución. Esta tecnología se llama Java Emitter Templates (JET) y forma parte de un proyecto más grande orientado a crear herramientas para el modelado de software que se integren con el ambiente de desarrollo Eclipse (“JET Tutorial,” 2012).

JET es una tecnología que permite generar código fuente a partir de plantillas definidas por el programador. Primero el programador define la estructura del documento a través de las plantillas con las secciones de contenido dinámico. El compilador JET traduce estas plantillas a funciones de código fuente que dados ciertos parámetros regresan una cadena de texto representando el documento generado.

El primer paso para utilizar JET es analizar las clases (Java) que se generarían en tiempo de ejecución. En una clase de Java existen secciones que son fijas y otras cuyo contenido es

dinámico. Una vez que se identificaron estas secciones en las diferentes clases involucradas en un componente, se generó un *template* para cada una de ellas. La Figura 30 muestra una sección de un *template*, específicamente el que corresponde al procesamiento de los datos y los transmite al siguiente componente en un proyecto de sensado: <Nombre>ProcessData.Java. Nombre corresponde al nombre designado para el componente.

```

1  <%@ jet package="edu.incense.manager.java.emitters" imports="java.util.*"
   class="DataComponentEmitter"%>
2  <%ComponentStructure componentStructure = (ComponentStructure) argument;%>
3  <%byte i = 0;%>
4  <%Date date = new Date();%>
5  /**
6   * New Component <%=componentStructure.getComponentName()%> generated
   automatically <%=date.toString()%>
7   */
8  package edu.incense.android.datatask.filter;
9  import edu.incense.android.datatask.data.*;
10 public class <%=componentStructure.getComponentName()%>ProcessData extends

DataFilter {
11   /*Section to declare configuration variables*/
12   <%for (i = 0; i<componentStructure.getConfigurationVariables().length; i++) {%>
13   private <%=componentStructure.getConfigurationVariables()[i][0]%>
14   <%=componentStructure.getConfigurationVariables()[i][1]%>;
15   <%}%>
16
17   /*Section to declare and initialize operation variables*/
18   <%for (i = 0; i<componentStructure.getOperationVariables().length; i++) {%>
19   private <%=componentStructure.getOperationVariables()[i][0]%>
20   <%=componentStructure.getOperationVariables()[i][1]%> =
   <%=componentStructure.getOperationVariables()[i][2]%>;
21   <%}%>

```

**Figura 30.** Segmento de una plantilla.

JET agrega un compilador al proyecto de desarrollo (JET Builder). Este compilador traduce el template a una clase Java que finalmente sirve para generar código fuente. Este compilador funciona en tiempo de desarrollo.

En el código de la Figura 30 se observan los caracteres <% y %>, estos sirven para marcar el inicio y fin de etiquetas. Las etiquetas enmarcan sentencias que son ejecutadas por el

compilador de JET. Las instrucciones válidas para utilizar en los templates son las mismas que se usan en JSP.

Al inicio del código (línea 1) en la Figura 30 se puede identificar el encabezado que deben llevar todos los templates, en esta línea se declara el nombre de la clase que servirá para generar código fuente en tiempo de ejecución además del paquete donde deberá ser creada en el proyecto, de esta manera está disponible en tiempo de ejecución (véase Figura 31).

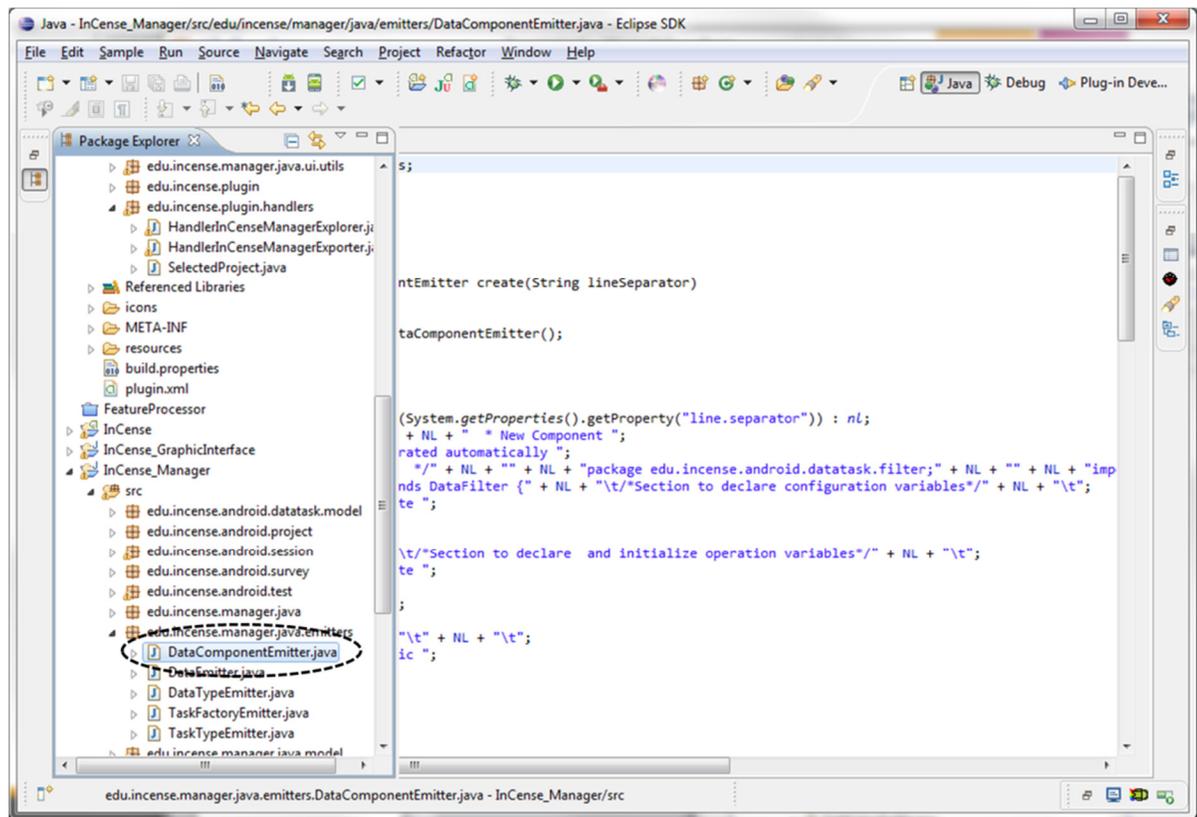


Figura 31. Template en su forma de clase de Java (después de JET Builder).

La estructura de la clase que corresponde al componente se puede apreciar hasta la línea 10 de código en la Figura 30. Las líneas 8 y 9 corresponden a elementos fijos (se encuentran en todos los componentes). A partir de ahí el template comienza a mostrar contenido dinámico (diferente para cada componente), por ejemplo, la línea 10 corresponde al nombre del componente, mientras que las líneas 12-20 sirven para declarar variables.

La función principal de un template traducido se llama `generate` (después de JET Builder) y recibe como argumento un objeto (`Java.lang.Object`) que por defecto se llama `argument`. Como se sabe, en el API de Java Object es el padre del resto de las clases de Java, por eso es que `argument` se puede invocar (`cast`) como cualquier otro tipo de objeto que se desee. En la solución propuesta para generar código fuente esto se aprovecha para combinar el template con los atributos del componente que se desee crear. Como se mencionó anteriormente se realizó una conceptualización de un componente que tiene tres elementos: nombre del componente, código para procesar datos y variables. Para aprovechar el componente genérico y combinarlo con un template traducido se realizó la implementación del mismo en Java (véase Figura 32). El nombre para la clase resultante fue: `ComponentStructure`. Se puede observar en la Figura 30 (línea 2) como se toma `argument` que contiene las características del componente y se realiza una invocación a `ComponentStructure`.



**Figura 32. Implementación del componente genérico.**

La Figura 32 muestra la clase `ComponentStructure` como se mostraría en un diagrama de clases, en la mitad superior de la Figura se encuentra el nombre de la clase junto con las variables que almacenan los atributos de la misma. En este caso se tiene el nombre del componente y las estructuras que guardan el código para procesar datos y las variables definidas por el usuario. En la mitad inferior se encuentran los métodos para acceder a esos atributos de manera indirecta (*getters* y *setters*).

Se crearon 5 *templates*, uno por cada clase involucrada en la creación de componentes, en cada *template* se envía `ComponentStructure` como argumento para generar el código fuente complementario, necesario para automatizar el proceso de creación de componentes. La mayor ventaja de usar JET y *templates* en conjunto con el componente genérico es que el usuario no necesita familiarizarse con el código fuente de InCense ya que este

conocimiento está implícito en la creación de los *templates*. Considerando la utilidad del componente genérico en combinación con JET y el hecho de que esta combinación de tecnologías coadyuva a reducir la intervención del usuario en la creación de componentes se podría decir que son las tecnologías habilitadoras que hacen posible facilitar el proceso de creación de componentes.

Si bien es cierto que el componente genérico en combinación con JET son las tecnologías que hacen posible facilitar el proceso de creación de componentes, todavía se puede facilitar aún más el proceso de creación de componentes. El código fuente generado en tiempo de ejecución debe almacenarse en un destino, por ejemplo en la Aplicación Móvil. Además, se puede vincular OntoInCense con la Aplicación Móvil para llevar la contabilidad de los componentes registrados en OntoInCense y los implementados en la Aplicación Móvil. Si el editor incluido en Protégé reemplaza al Editor InCense de la arquitectura original, todavía es necesario crear un archivo de configuración para la Aplicación Móvil a partir de las campañas de sensado que se registren en OntoInCense. En otras palabras, se requiere de una herramienta de administración de componentes y campañas de sensado; que además sirva como vínculo entre Protégé y la Aplicación Móvil. Para este propósito se propone utilizar InCense Manager, cuya funcionalidad se detalla en la siguiente sección.

#### **4.3.3. InCense Manager**

Es preciso recordar que la arquitectura extendida es una propuesta para modificar la arquitectura original de la plataforma de sensado (InCense) con el objetivo de facilitar la creación de nuevos componentes. De manera general se plantean importantes retos que un usuario debe enfrentar para crear un nuevo componente, especialmente uno poco experimentado en el desarrollo de software. Dificultades tales como estudiar y comprender la arquitectura de InCense, implementar componentes directamente sobre el código fuente, llevar la contabilidad de los componentes que se implementan y algunas más que ya se han expuesto. Para atacar esta serie de obstáculos y siguiendo el objetivo general del presente trabajo de investigación, se proponen una serie de tecnologías cuyo propósito es facilitar el

proceso de creación de componentes y reducir la intervención del usuario en el mismo: el componente genérico, OntoInCense y JET.

Sin embargo, si el usuario decide usar las tecnologías mencionadas de manera independiente una de la otra, entonces el usuario necesitaría tener un conocimiento profundo de la arquitectura de InCense. Por ejemplo, podría darse el caso de que diseñe un componente y antes de registrarlo en OntoInCense genere las clases de código fuente del mismo componente insertándolas en la Aplicación Móvil para luego intentar utilizar el componente en una campaña de sensado. Esto sería imposible puesto que el componente nunca se habría registrado en OntoInCense. Lo descrito anteriormente se trata de un caso hipotético en el que el usuario no sigue el flujo del proceso de creación de un componente de la manera más conveniente (utilizando la arquitectura extendida).

Por lo argumentado anteriormente, se propone InCense Manager como un elemento más de la arquitectura extendida. InCense Manager es un plug-in que se integra al ambiente de desarrollo Eclipse. El propósito principal de esta herramienta es el de servir como vínculo entre OntoInCense, la Aplicación Móvil y el servidor de proyectos.

Se propone distribuir la Aplicación Móvil junto con OntoInCense e InCense Manager (plug-in) de manera que el usuario pueda abrir la ontología y registrar nuevos componentes como se muestra en la Figura 33. InCense Manager también se integra en el ambiente de desarrollo desde un menú del mismo nombre y con la adición de botones en la barra de herramientas. Al integrar InCense Manager y Eclipse se simplifica aún más el proceso de desarrollo de componentes ya que un plug-in tiene acceso de lectura y escritura a las carpetas del proyecto (Aplicación Móvil). Siendo así, se tiene acceso a OntoInCense y los componentes que sean generados se escriben directamente en la Aplicación Móvil. Al final el usuario tiene en un mismo ambiente todas las herramientas que necesita para diseñar e implementar componentes.

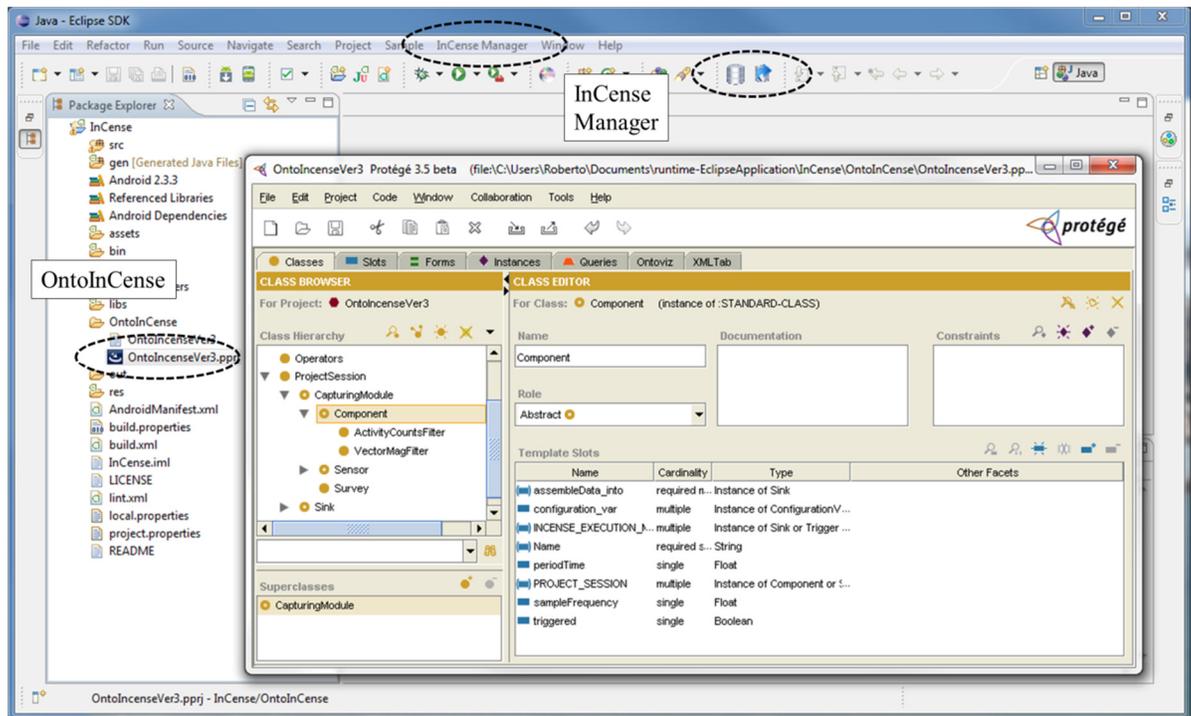


Figura 33. Arquitectura extendida integrada al ambiente de desarrollo Eclipse.

InCense Manager se compone de tres módulos: repositorio de componentes, generador de código (explorer) y el exportador de campañas de sensado (exporter).

Repositorio de componentes. El usuario no tiene contacto de manera directa con este módulo. En cierta forma este módulo es responsable de mantener la contabilidad de los componentes que se han registrado en la ontología y aquellos que ya han sido editados por el usuario. De manera automática lee el contenido de OntoInCense para extraer una lista con los componentes existentes, luego compara la lista con otra lista obtenida de una base de datos (véase Figura 34) que contiene componentes que han sido editados por medio de InCense Manager. Se supone que esta segunda lista contiene los componentes completos; cabe mencionar que OntoInCense no almacena el código fuente para procesar datos en cada componente. De manera que los componentes almacenados en la base de datos deben ser aquellos que ya han sido editados y están completos. Entonces el repositorio de componentes combina ambas listas en una sola, de esta forma se determinan cuáles son los componentes faltantes por editar. Es decir que gracias a este repositorio se mantiene

contabilizados los componentes que se han registrado en OntoInCense y aquellos que se implementaron en la Aplicación Móvil.

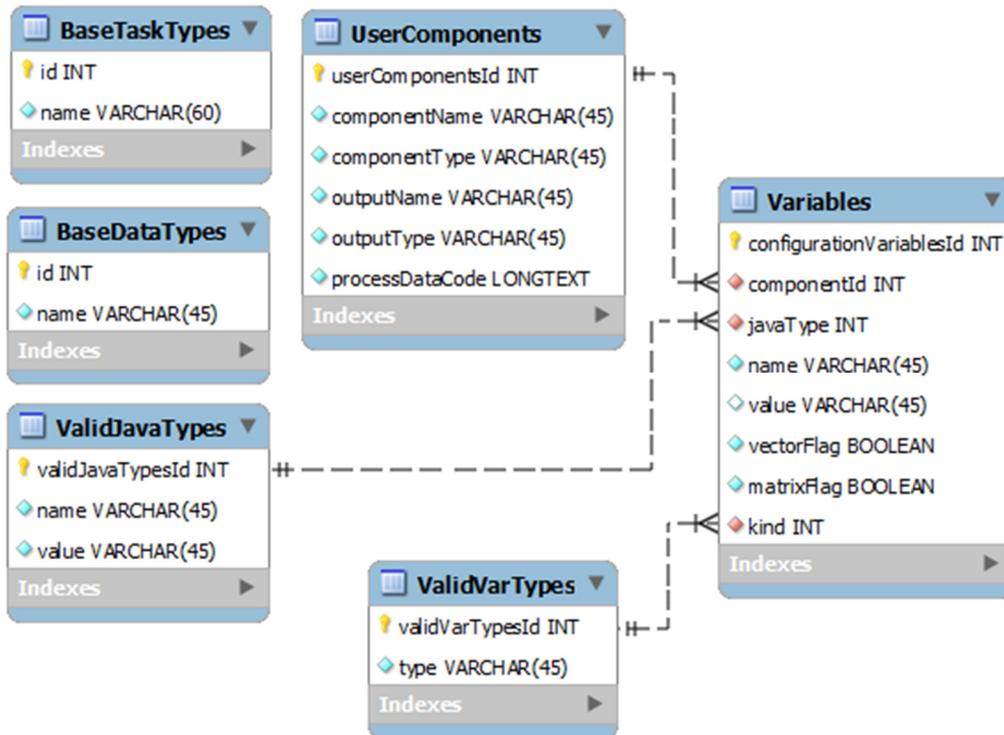


Figura 34. Diagrama Entidad-Relación de la base de datos de componentes.

La Figura 34 muestra el diseño de la base de datos de componentes, cuya implementación se realizó con el manejador de bases de datos MySQL 5. La razón principal de haber utilizado este manejador de bases de datos es que la edición *community* es gratuita y que además cuenta con el soporte de Oracle que es una empresa muy prestigiada en el tema de bases de datos. El componente en sí se almacena en las tablas UserComponents y Variables. La primera tabla almacena el nombre del componente y el código de procesamiento de datos. La tabla Variables se utiliza para almacenar todas las variables asociadas a cada componente, por eso existe una relación de 1 a  $n$  entre el identificador del componente (tabla UserComponents) y el campo homólogo en la tabla Variables. Las otras tablas son catálogos para conocer los tipos de variables existentes en Java. Los catálogos son útiles para establecer una convención de cómo se almacenarán las variables. Por ejemplo en Java existe la variable de tipo entero que se declara como `int`, pero de no

existir una convención o un catálogo de tipos de variables válidos, entonces otro usuario podría nombrarla como `Integer`.

Explorer. Este módulo permite al usuario visualizar los componentes que existen en `OntoInCense` y agregar el código fuente para el procesamiento de los datos (componente genérico) además de las variables que el usuario desee. En este módulo es donde el usuario puede visualizar el repositorio de componentes para conocer cuales ya han sido editados y cuales requieren de edición. La lista de componentes del repositorio se muestra en un `ComboBox` (lista descendente) que se muestra en la esquina superior izquierda de la ventana principal del módulo (véase Figura 35).

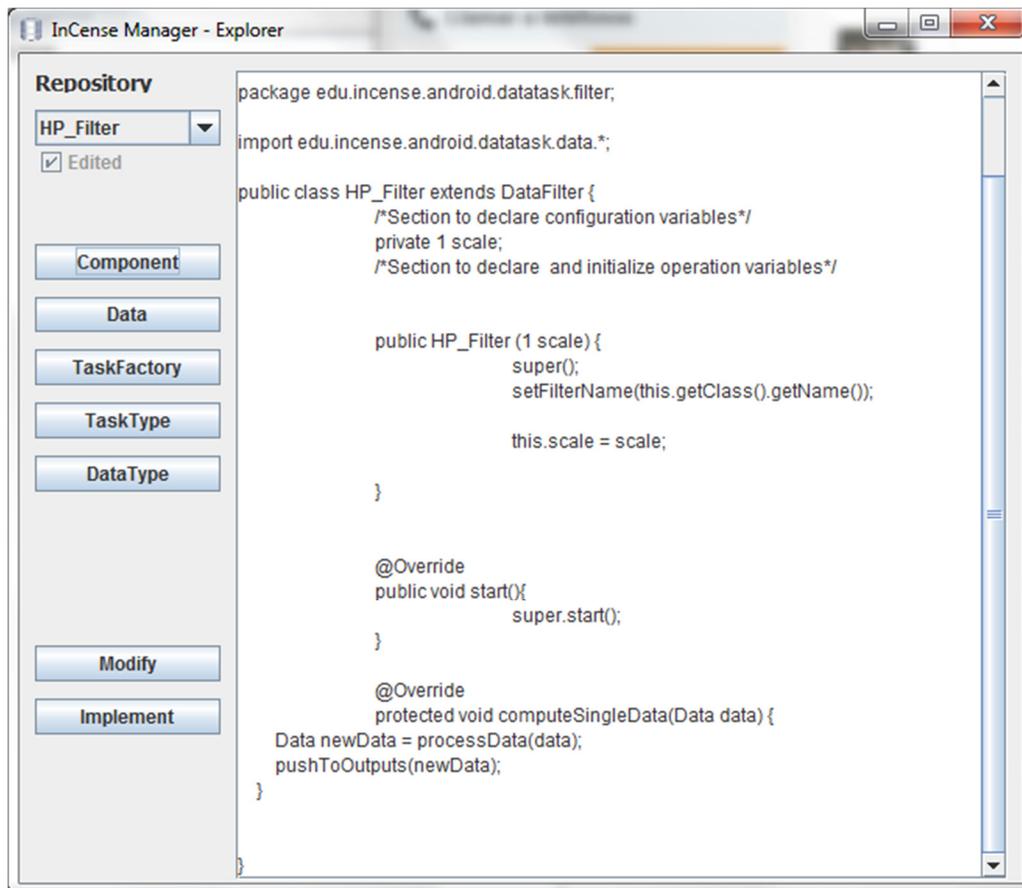
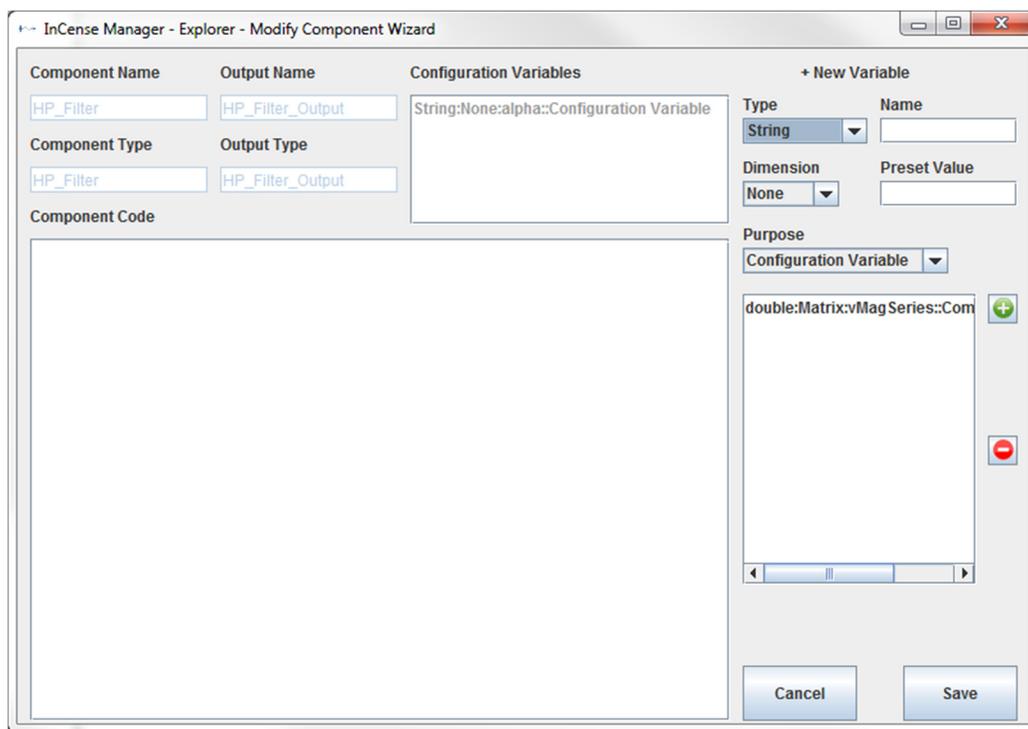


Figura 35. Explorador de componentes (Explorer) - ventana principal.

Cuando el usuario elige un componente de la lista puede visualizar el código fuente completo de cualquiera de las 5 clases de Java que están asociadas con un componente. Solo tendría que presionar sobre el botón con el nombre de la clase que desee visualizar.

Por otra parte, para editar el componente existe una ventana de edición Modify Component Wizard (véase Figura 36). En esta ventana solo se puede editar el código de procesamiento de datos y agregar tres tipos de variables que pueden utilizarse en el código de procesamiento de datos. La razón por la cual no se puede modificar otros elementos como el nombre del componente es que el nombre del componente se designa en OntoInCense.



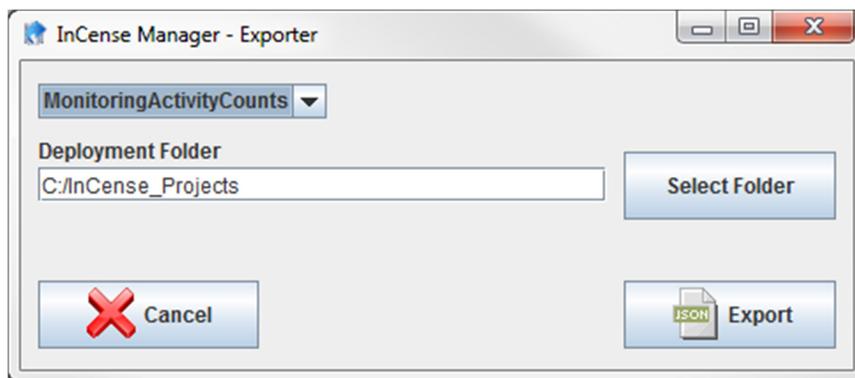
**Figura 36. Explorador de componentes - Modificar componente.**

Una vez que el usuario terminó la edición puede guardar los cambios en la base de datos de componentes y estarán disponibles en lo subsecuente. Entonces, Explorer muestra la ventana principal nuevamente (véase Figura 35).

Siguiendo la Figura 35, el botón Implement escribe en la Aplicación Móvil las 5 clases necesarias para implementar un componente. Para ello se apoya del módulo generador de código que es un conjunto de clases para generar código fuente (templates traducidos).

La mayor ventaja de realizar la edición de componentes con Explorer es que el usuario en ningún momento requiere saber cuáles clases son las que deben agregarse al generar un nuevo componente o si estas extienden alguna nueva clase. Ese conocimiento existe de manera implícita en Explorer y los templates traducidos.

Exporter. Este módulo permite exportar todas las campañas de sensado existentes en OntoInCense como archivos de configuración tipo JSON (arquitectura original). Para lograr esto, InCense Manager utiliza la API de Protégé para leer la ontología, específicamente las instancias de la clase ProjectSession que son las que contienen los diferentes proyectos de sensado. A medida que Exporter lee la ontología, cada elemento existente en los proyectos de sensado es traducido a su equivalente en JSON. La Figura 37 muestra la ventana principal de Exporter. Para generar un archivo de configuración para la Aplicación Móvil el usuario debe seleccionar el proyecto deseado de la lista de proyectos existentes en OntoInCense y seleccionar el folder de destino del archivo JSON. Por último, el usuario solo debe presionar sobre el botón export.



**Figura 37. InCense Manager – Exporter.**

Ya se discutieron por separado todos los elementos que extienden la arquitectura original dando como resultado la arquitectura extendida. Enseguida se explica con un diagrama arquitectónico como se cohesionan y la manera en que se comunican entre sí.

#### 4.3.4. Diagrama de la Arquitectura Extendida

En el Capítulo 3 se estudió la arquitectura original de InCense junto con todos sus elementos y la relación entre estos. La Figura 38 muestra el diagrama de la arquitectura extendida. Como se puede apreciar, se encuentran la mayoría de los elementos originales con la excepción del Editor InCense. Por las razones expuestas en una sección anterior, se reemplazó el Editor InCense con OntoInCense.

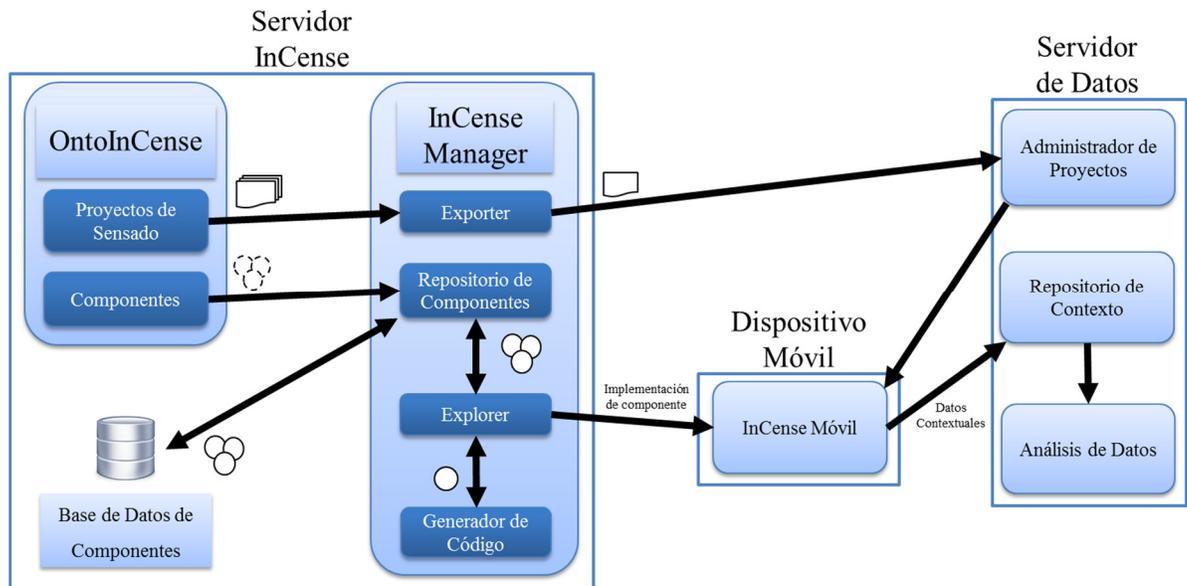
El primer elemento en la arquitectura extendida es OntoInCense. En este elemento se registran nuevos componentes y nuevos proyectos de sensado. Luego está InCense Manager que cuenta con el módulo Exporter, responsable de recuperar el conjunto de proyectos de sensado. Entonces estos proyectos de sensado traducidos a formato JSON como archivos de configuración de proyecto y luego se transmiten al administrador de proyectos.

En el mismo InCense Manager se encuentra el repositorio de componentes que se encarga de leer la ontología y recuperar los componentes registrados por el usuario en OntoInCense. Cabe mencionar que en este punto todos los componentes de OntoInCense se encuentran incompletos puesto que no cuentan con el código fuente necesario para procesar datos. A la vez, el repositorio de componentes también hace una consulta a la base de datos de componentes para obtener el listado de los componentes que sí están completos o ya han sido editados por el usuario. Entonces el repositorio compara ambas listas y las combina en una sola que representa el repositorio completo de componentes. Cabe mencionar que esta tarea es transparente para el usuario.

Explorer es el módulo de InCense Manager que se encarga de obtener el conjunto de componentes existentes en el repositorio para presentarlos al usuario a la vez que facilita su edición. En caso de modificar alguno de los componentes en el repositorio, este será actualizado y almacenado en la base de datos de componentes a través del mismo repositorio de componentes.

Explorer también le permite al usuario generar de manera automática la implementación de los componentes en el repositorio. Esto se hace gracias a la función `generate` de los

templates traducidos que se encuentran en el módulo Generador de Código Fuente. Entonces Explorer escribe directamente sobre la Aplicación Móvil la implementación del componente seleccionado. También vale la pena mencionar que esta última tarea se realiza de manera transparente para el usuario.



**Figura 38. Arquitectura extendida.**

Después de InCense Manager, la arquitectura extendida continúa igual que la arquitectura original. El administrador de proyectos se encarga de distribuir el archivo de configuración a los teléfonos inteligentes que estén corriendo una instancia de la Aplicación Móvil. Luego los datos contextuales recolectados se transmiten al repositorio de contexto para su análisis.

A continuación se describe el nuevo proceso de creación de componentes puesto que este se modifica con la propuesta de la arquitectura extendida.

#### **4.4. Resumen de capítulo**

En este capítulo se realizó un análisis de la arquitectura original para detectar sus limitaciones y proponer mejoras principalmente orientadas a facilitar al usuario el diseño de campañas de sensado. A partir de este análisis se propone un conjunto de modificaciones a la arquitectura dando como resultado la arquitectura extendida. Se propone una ontología

(OntoInCense) para facilitar el aprendizaje de la plataforma y como una capa de abstracción en el diseño de componentes. También se propone la adición de InCense Manager que se compone de tres módulos: el repositorio de componentes, el generador de código y el exportador de campañas de sensado.

Con la arquitectura extendida el proceso de creación de componentes se ve modificado. En el siguiente capítulo se discute el nuevo proceso de creación de componentes en comparación con el de la arquitectura original.

## Capítulo 5

---

### Creación de nuevos componentes con InCense extendido

Con la propuesta de una arquitectura extendida el proceso de creación de componentes ha sido simplificado. En las siguientes líneas se detalla este nuevo proceso a la vez se describe la creación de un componente como caso de estudio: el estimador de actividad física. También se discute como se logró facilitar el proceso de creación de componentes.

#### 5.1. Nuevo proceso de creación de componentes

Para describir el proceso de creación de componentes es necesario identificar las actividades, roles y artefactos que intervienen en el mismo. A continuación se describen las actividades que el usuario debe realizar para crear un nuevo componente.

Estudio de *OntoInCense*. Cuando un usuario se dispone a utilizar el kit de investigación (InCense) invariablemente debe aprender sobre su arquitectura, capacidades, familiarizarse con los elementos de la ontología y como se relacionan entre sí. Por ello lo primero que el usuario debe hacer es estudiar *OntoInCense* que conceptualiza este conocimiento.

Diseño de Componente. Esta actividad debe hacerse tomando como plantilla el componente genérico. El usuario debe asignar un nombre para el componente, en caso de ser necesario también establecer las variables de configuración que se requieran. También es en este momento cuando se escribe el código de procesamiento de datos y en su caso definir variables o constantes conforme se requiera en el código para procesar datos.

Registro del componente en *OntoInCense*. Esta actividad implica crear una subclase de la clase *Component* en la ontología. Una vez creada la subclase el usuario debe establecer el nombre y las variables de configuración (si se requiere alguna). Después solo resta establecer los componentes con los que puede ser asociado el nuevo componente y asignarle forma y color para su representación gráfica.

Edición e implementación del componente. El usuario debe editar el componente en el módulo Explorer de InCense Manager. La edición consiste en agregar el código fuente para procesar datos y el resto de variables que no son de configuración. Entonces el usuario debe presionar sobre el botón Implement para que se genere automáticamente el resto del código fuente.

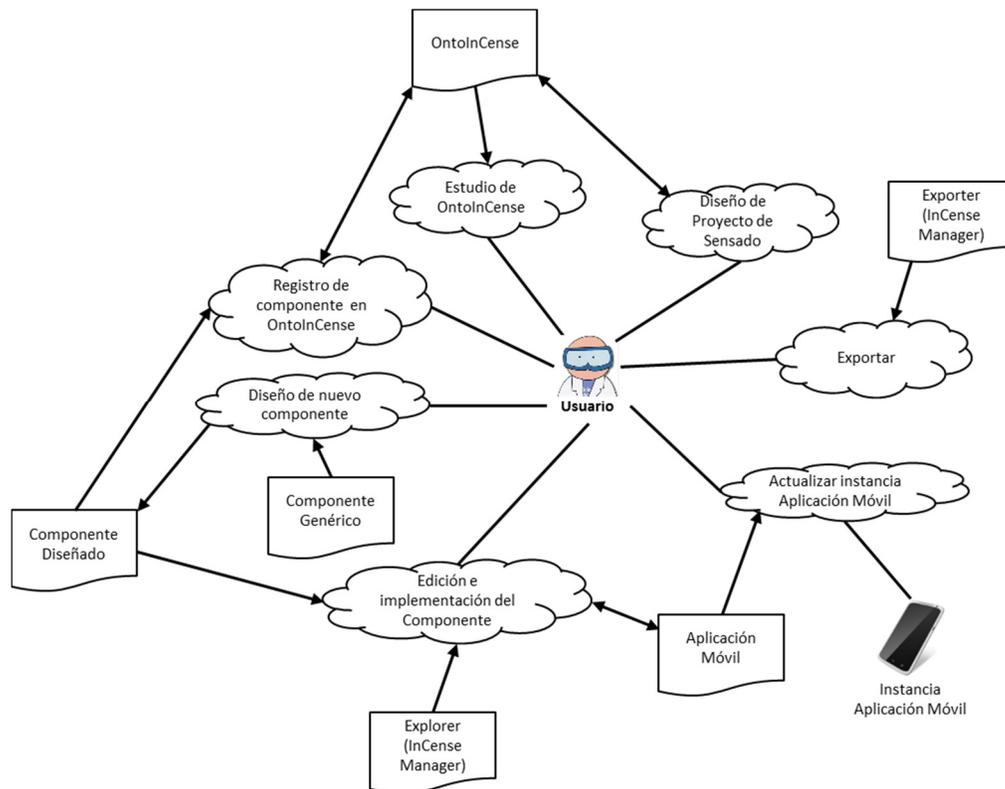
Actualizar instancia de la Aplicación Móvil. Por último el usuario deberá actualizar la Aplicación Móvil en los dispositivos móviles que participan en el proyecto de sensado.

Complementario al proceso de creación de componentes también está el proceso de creación de proyectos de sensado. Este proceso consiste en crear una configuración válida para que la Aplicación Móvil obtenga datos de acuerdo a un proyecto de sensado en específico. Se planteó anteriormente que OntoInCense permite la creación de estos proyectos con un editor gráfico incluido en Protégé. La mayor ventaja de usar OntoInCense para el registro de componentes y proyectos de sensado es que no se requiere doble implementación del componente ni que modifique de manera directa código fuente de la plataforma de sensado (InCense), como se hacía anteriormente en la arquitectura original

El proceso de creación de campañas de sensado solo requiere de dos actividades que el usuario debe realizar: el diseño del proyecto de sensado en OntoInCense y la exportación del mismo a través de Exporter (InCense Manager).

Diseño del proyecto de sensado. Para realizar esta actividad el usuario debe utilizar el editor gráfico incluido en Protégé para editar OntoInCense y crear una nueva instancia de la clase ProjectSession. En el nuevo proyecto el usuario puede incluir los sensores que ya han sido registrados en la ontología y establecer relaciones entre los mismos.

Exportar proyecto de sensado. El módulo Exporter de InCense Manager muestra todos los proyectos de sensado que han sido creados en OntoInCense. Entonces el usuario selecciona el proyecto que desea exportar y la carpeta donde será exportado. Por último, solo debe presionar sobre el botón Export.



**Figura 39. Proceso de creación de componentes y proceso de creación de proyectos de sensado combinados.**

La Figura 39 muestra el proceso de creación de componentes y el de creación de proyectos de sensado combinados tal como se realizarían con la arquitectura extendida. Si se toma en cuenta solo la cantidad de actividades que el usuario debe realizar se puede decir que es similar a los procesos homólogos de la arquitectura original. Sin embargo, hay una diferencia fundamental: Se logró reducir la exposición del usuario al código fuente hasta el punto en que con el nuevo proceso, el usuario no requiere estudiar la implementación de la Aplicación Móvil o del editor gráfico de la arquitectura original.

La siguiente sección diseña e implementa el componente estimador de nivel de actividad. Esto permitirá observar detalladamente el proceso de creación de componentes en la arquitectura extendida.

## **5.2. Caso de estudio: Estimador de nivel de actividad**

Para mostrar cómo se utiliza la arquitectura extendida se seleccionó el estimador de nivel de actividad mencionado en el Capítulo 3. La razón por la cual se eligió este componente es que existe un amplio interés académico por estimar el nivel de actividad de un individuo. De acuerdo a (Puyau et al., 2004), la actividad física está asociada con riesgos de salud y diabetes tipo 2, por lo que se requieren de métodos no intrusivos, válidos y precisos para entender como la intensidad, frecuencia y duración de la actividad física influye en la salud.

En el laboratorio de cómputo móvil y ubicuo del departamento de ciencias de la computación en el CICESE se realiza investigación orientada al cuidado de la salud de los adultos mayores por lo que este componente podría ser de ayuda para el departamento en futuras campañas de sensado.

Para realizar este ejercicio demostrativo con la arquitectura extendida se preparó una computadora portátil a la cual se le instaló Windows 7, el ambiente de desarrollo Eclipse con el plug-in de Android. Además se instaló Protégé 3.5 como editor de OntoInCense. InCense Manager se encuentra en etapa de desarrollo por lo que es necesario ejecutar el código fuente del plug-in a manera de prueba. Lo anterior provoca que se abra una instancia de prueba en Eclipse con un espacio de trabajo completamente en blanco. Por ello también se debe importar el proyecto de la Aplicación Móvil de InCense al ambiente de prueba. Un usuario del kit de investigación tendrá acceso a InCense Manager como un plug-in que podrá descargar directamente a su ambiente de desarrollo.

### **5.2.1. Estudio de OntoInCense**

La primera actividad a realizar cuando se diseña un componente es estudiar OntoInCense. Con esta actividad el usuario conocerá los distintos elementos de la ontología. Así como la documentación de cada componente ya registrado y sus sensores. La Figura 40 muestra la documentación de uno de los componentes. En el editor de clase de Protégé el usuario puede aprender conocer la función para acceder a la variable que contiene los datos que ya fueron procesados en el componente.

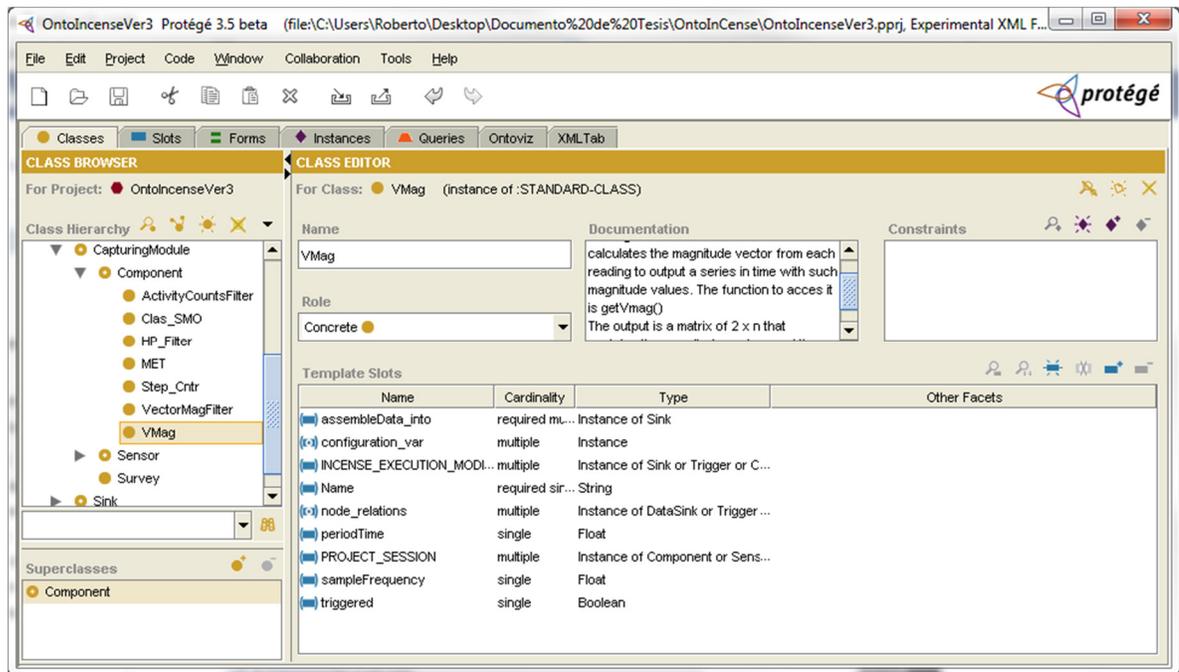


Figura 40. Documentación del componente VMag.

Gracias a esta actividad el usuario aprende sobre la arquitectura de InCense. Cabe mencionar que esta actividad solo será necesaria en las primeras ocasiones puesto que con el uso de la plataforma el usuario ganará experiencia y conocimiento sobre la misma.

### 5.2.2. Diseño del nuevo componente

Para realizar el diseño de un nuevo componente el usuario deberá tomar en cuenta el componente genérico. La asignación de un nombre para el componente es sencilla, debe iniciar con mayúscula y una letra, no debe contener caracteres especiales, excepto por el guión bajo (\_). Dicho de otra forma, debe seguir la convención utilizada para nombrar clases en Java, ya que este nombre será el nombre base para las 5 clases involucradas en la creación de un componente. El nombre designado para el estimador de nivel de actividad es MET (por *metabolic equivalent of task* o equivalente metabólico de tarea).

Luego, se debe proporcionar el código fuente para procesar los datos así como las variables de configuración, globales, salidas y estáticas (en caso de requerirse alguna).

Equivalente Metabólico de Tarea (MET). Para realizar mediciones sobre el nivel de actividad de una persona existen algunas estrategias ya documentadas en la literatura (Avci et al., 2010; Moran et al., 2004; Puyau et al., 2004; Sorber et al., 2012); una buena estimación puede hacerse utilizando acelerómetros para calcular *activity counts* (AC), estos se pueden calcular con tres técnicas: *Proportional Integral Mode* (PIM), *Zero Crossing Mode* (ZCM) y *Time Above Threshold* (TAT). De acuerdo a (Moran et al., 2004; Sorber et al., 2012) PIM resulta ser el que presenta mayor correlación con el gasto energético.

PIM es una técnica que consiste en tomar muestras de la aceleración del usuario en el tiempo y estimar el área bajo la curva cada cierto periodo de tiempo; en (Moran et al., 2004; Puyau et al., 2004) este periodo de tiempo es de un minuto.

Para establecer el nivel de actividad de un usuario del acelerómetro, es necesario convertir el nivel de actividad a unidades de Equivalente Metabólico de Tarea (MET), tal como se hizo en (Crouter, Churilla, et al., 2006; Crouter, Clowers, et al., 2006). Con esta conversión a METs se puede determinar el nivel de actividad de un usuario, al menos en tres rangos: menor a 3 METs equivale a actividad ligera, un rango de 3-6 METs significa actividad moderada y mayor que 6 METs actividad vigorosa (Puyau et al., 2004).

Ahora bien, los AC se calculan de acuerdo a la técnica PIM estimando el área bajo la curva de la serie de valores discretos correspondientes a aceleraciones medidas por un acelerómetro portado por el usuario en una unidad de tiempo, es decir, sin la fuerza de gravedad que está presente en cualquier lectura del acelerómetro. Es necesario observar que los niveles de AC que se mencionan en la literatura revisada y que luego se convierten a METs se encuentran en unidades de gravedad, por lo que si se quisiera implementar un componente de este tipo en InCense es necesario hacer una operación aritmética para convertir los valores arrojados por el acelerómetro en  $\text{m/s}^2$  a unidades de gravedad.

$$\text{Si } 1 \text{ g} = 9.8 \frac{\text{m}}{\text{s}^2}, \text{ entonces } x \frac{\text{m}}{\text{s}^2} = \frac{x}{9.8} \text{ g.}$$

Con la revisión de la literatura sobre el tema se obtuvo la información necesaria para diseñar el resto del componente. Se sabe cómo calcular los METs a partir del acelerómetro

además de que se tiene una escala para determinar el nivel de actividad en base a la cantidad de METs registrados: actividad ligera, moderada y vigorosa.

Código fuente de procesamiento de datos. Se detectaron dos variables muy importantes para el funcionamiento del componente. La primera es un vector con los umbrales de niveles de actividad física: ligera, moderada y vigorosa. Estas variables de configuración, se designaron como *scale* y en la primera posición del vector está el número 3, mientras que en la segunda posición el 6. De esa forma los tres niveles de actividad física quedan divididos con dos umbrales. La segunda es una matriz de  $2 \times n$  que contiene las estimaciones de niveles de actividad y una etiqueta de tiempo para cada estimación. Esta es una variable de salida, es decir que almacena los datos ya procesados. Su designación es *metLevels*.

Para escribir el código de procesamiento de datos el usuario deben seguir algunas reglas respecto al nombre de la función que procesará los datos y los nombres de las clases que almacenan los datos de entrada y los datos procesados:

- La función que procesa los datos que entran al componente se llama *processData*.
- Esta función (*processData*) toma como argumento un objeto de tipo *Data* que contiene los datos del componente o sensor precedente al que se quiere implementar. Por ejemplo, se quiere implementar un estimador de nivel de actividad física (MET). Por lo tanto el objeto *Data* que reciba *processData* como argumento será una instancia de *VMagData*<sup>14</sup>.
- Como retorno, *processData* debe regresar un objeto *Data* que contenga los datos ya procesados. Siguiendo la convención de nombres para los objetos *Data* de cada componente, el nombre para el objeto de retorno será *METData* (<nombre base>*Data*).

Al respecto de las características de un componente en específico, el usuario es responsable de revisar en *OntoInCense* las variables de salida y como acceder a ellas (véase Figura 40).

---

<sup>14</sup> Como se indica en la literatura, el componente que precede a MET es el que calcula el vector magnitud del acelerómetro (*VMag*).

```

1 private Data processData(Data data){
2     VMagData vMagData = (VMagData) data;
3     double[][] vMag = vMagData.getVmag();
4     int windowSize = 45 * 60;
5     double acumulador = 0.0;
6     String[][] activityLevel = new String[vMag.length()][2];
7     int i = 0;
8
9     for (i = 0; i < vMag.length(); i++){
10        acumulador += vMag[i][1] / 9.8;
11        if (i % windowSize == 0){
12            activityLevel[i][0] = Double.toString(vMag[i][0]);
13            if (acumulador > scale[1]){
14                activityLevel[i][1] = "Vigorosa";
15            } else if (acumulador > scale[0]){
16                activityLevel[i][1] = "Moderada";
17            } else{
18                activityLevel[i][1] = "Ligera";
19            }
20            acumulador = 0.0;
21        }
22    }
23    METData newData = new METData (activityLevel);
24    return newData;
25 }

```

**Figura 41. Código de procesamiento de datos en el componente MET.**

La Figura 41 muestra el código fuente creado para procesar los datos que provienen del componente VMag y enviar los resultados al siguiente componente, *trigger* o *sink* en el proyecto de sensado. Las líneas de código 2 y 3 son para recuperar los datos que llegan desde el componente predecesor (VMag). De la línea de código 4 hasta la 7 se declaran variables locales que son necesarias para el procesamiento de los datos. Las líneas de código desde la 9 a la 22 procesan los datos. Cabe mencionar que en las líneas de código 13 y 15 se hace uso de una variable que no está declarada globalmente: *scale*. Esta variable es de configuración por lo que el usuario solo debe asegurarse de que será registrada en OntoInCense (siguiente actividad). Por último en la línea 23 se crea un objeto de tipo METData cuyo constructor toma como argumento una matriz de 2 dimensiones de tipo String. Esto se debe a que se estableció una variable de salida de estas características para almacenar los resultados del procesamiento de datos.

Con esto queda concluido el diseño del componente. Esta es la actividad que presentaría mayor dificultad para un usuario con un bajo nivel de habilidad técnica en programación. Sin embargo, el hecho de que existen reglas y guías para la creación del código ayuda mucho al usuario. Además, hasta este punto el usuario no se preocupa de conocer a fondo el código fuente de la Aplicación Móvil o un editor gráfico. Solo se concentra en la sección de procesamiento de datos. En un caso extremo, con la guía de un investigador un estudiante de programación podría escribir este código a partir de un pseudocódigo ya que no requiere mucha experiencia. Siendo esta actividad la única en la que el usuario es expuesto a código de programación. Por otra parte, en la medida en que se integren nuevos componentes a InCense cada vez será menos necesario recurrir al diseño de un nuevo componente para implementar una campaña de sensado.

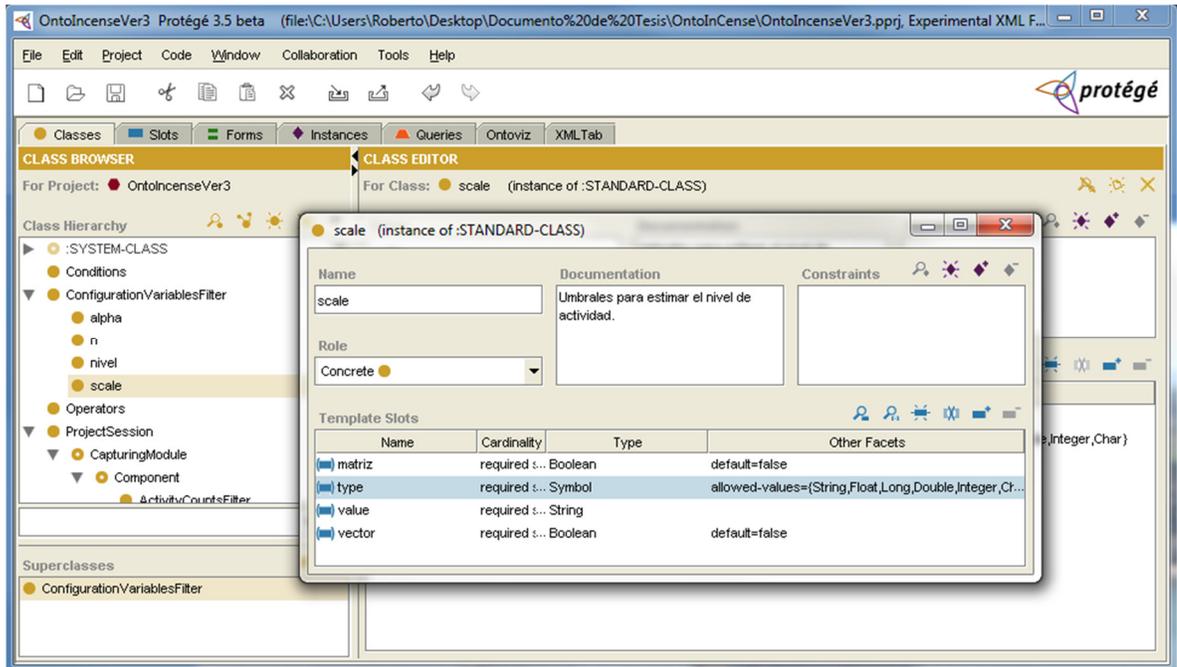
### **5.2.3. Registro de componente en OntoInCense**

La siguiente actividad en orden de ejecución es registrar el nuevo componente en OntoInCense. OntoInCense se incluye en el proyecto de la Aplicación Móvil. Cabe mencionar que en la ontología solo se registran las variables de configuración, dejando para después el registro de las otras variables en InCense Manager. A continuación se muestra paso a paso como es que el usuario debe realizar esta actividad.

Crear variables de configuración. Como se mencionó anteriormente, del diseño del componente genérico se observa que un componente puede tener varios tipos de variables para ser usadas en el mismo. Una instancia de estas variables son las variables de configuración. Este tipo de variables sirve para almacenar información necesaria para configurar o personalizar un componente. En este caso para el componente MET se encontró la variable *scale* que de manera general sirve para establecer los umbrales entre los tres niveles de actividad física: ligera, moderada y vigorosa. Enseguida se describe el proceso para registrarla en OntoInCense.

Este paso consiste en registrar las variables de configuración resultantes de la actividad de diseño del componente. En general, ninguna variable forma parte de la arquitectura de InCense, al menos no de manera directa como un *trigger* o un sensor. Por ello es que la clase de la cual derivan todas las variables de configuración en OntoInCense se encuentra

fuera de la clase `ProjectSession`. De manera que el usuario debe crear una subclase de `ConfigurationVariableFilter` en `OntoInCense`.



**Figura 42.** Registro de variable de configuración en `OntoInCense`.

Cuando se crea la nueva clase se abre un formulario donde el usuario debe asignar un nombre, hacer una breve documentación de la variable (opcional) y asignar un valor para cada uno de los 4 campos de la variable: `matriz`, `type`, `value` y `vector`. Tal como se muestra en la Figura 42.

El par de campos `matriz` y `vector` son de tipo booleano y su valor indican si la variable representa un vector, una matriz o una variable convencional. En caso de que ninguno de los dos este habilitado (valor verdadero), entonces se trata de una variable convencional, si solo `vector` está habilitada la variable representa un vector y en cualquier otro caso la variable representa una matriz de dos dimensiones. El campo `type` tiene el propósito de indicar de qué tipo de variable se trata de acuerdo a los tipos de datos definidos para Java: `String`, `Float`, `Long`, `Double`, `Integer` y `Char`. El campo `value` almacena el valor predeterminado para la variable, es decir aquel valor que utilizará el componente para su funcionamiento.

En el caso de la variable *scale* (necesaria para el componente MET) se creó como una variable vector de tipo `Double` cuyo valor por default es la cadena “{3,6}”. En Java los vectores también se representan como series de valores separados por una coma y delimitados por corchete. Por último se debe crear una instancia de la clase en la ventana de instancias de Protégé. No se requiere editar puesto que ya se estableció en la clase los valores que la instancia tendrá por defecto.

Crear clase del componente. De manera similar a como se hizo con la variable de configuración el usuario deberá crear una subclase de la clase `Component` en `ProjectSession` y designar un nombre además de documentar las variables de salida como se muestra en la Figura 43.

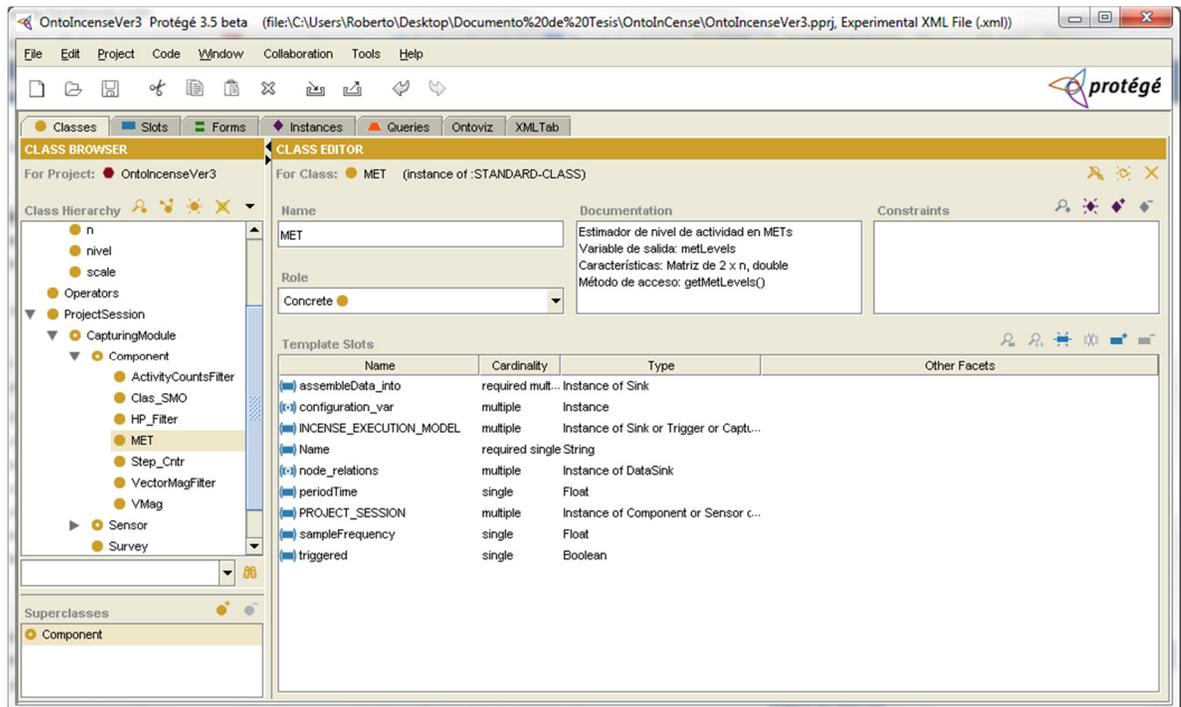
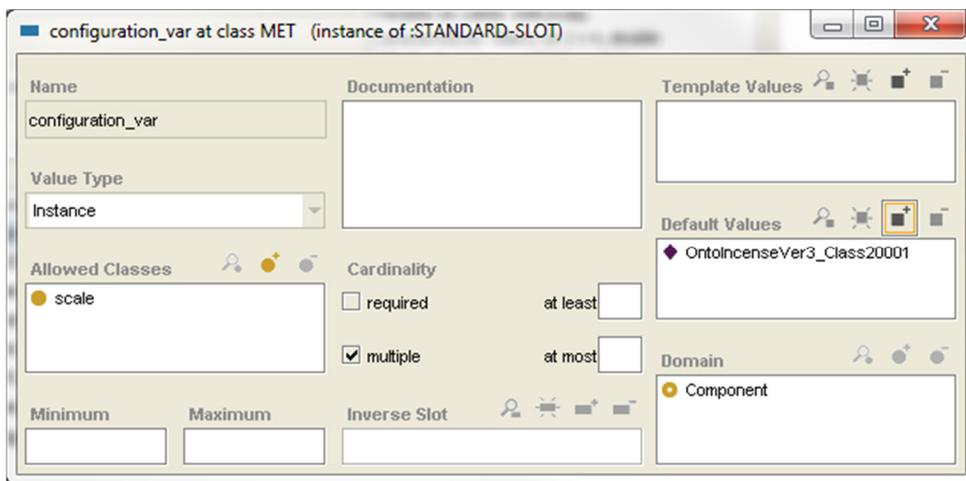


Figura 43. Creación de la clase MET.

Para el componente MET se documentó la variable de salida `metLevels`. Esta documentación es recomendable ya que es una manera más sencilla de que el usuario recuerde el nombre de las variables de salida y deducir el nombre del método para acceder a ellas. Los *getters* para las variables de salida se generan de manera automática por los

templates, por lo que se sigue una convención para nombrarlos. Al nombre de la variable se le antepone la cadena *get* al nombre de la variable, pero antes se cambia el primer carácter del nombre de la variable por mayúscula. De manera que el *getter* de *metLevels* sería: *getMetLevels*.

De igual manera que como se hizo con la variable de configuración, en el caso del componente se deben asignar los valores a los campos que se encuentran en el componente. Los campos más importantes a editar son *configuration\_var* y *node\_relations*. El primero tiene el propósito de asociar la variable de configuración que se creó anteriormente con el componente. El valor por defecto que deberá tener este campo es el de la instancia de la clase de la variable de configuración que se creó anteriormente. La Figura 44 muestra un ejemplo de la edición de *configuration\_var* para el caso de estudio.



**Figura 44.** Edición del campo *configuration\_var*.

El campo *node\_relations* cumple con el propósito de establecer los componentes o elementos con los que MET podrá ser asociado. Por defecto todos los componentes se puede asociar con *trigger* y *sink*. En el caso de MET, siendo un componente de alto nivel estas asociaciones son las adecuadas. Si MET recibe los datos de entrada de otro componente, será necesario editar dicho componente para que pueda ser asociado con MET. La ventaja de hacer esto es que se impiden asociaciones entre componentes incompatibles, por ejemplo el detector de actividad de voz con estimador de nivel de actividad. La Figura 45 muestra la edición de *node\_relations* para MET.

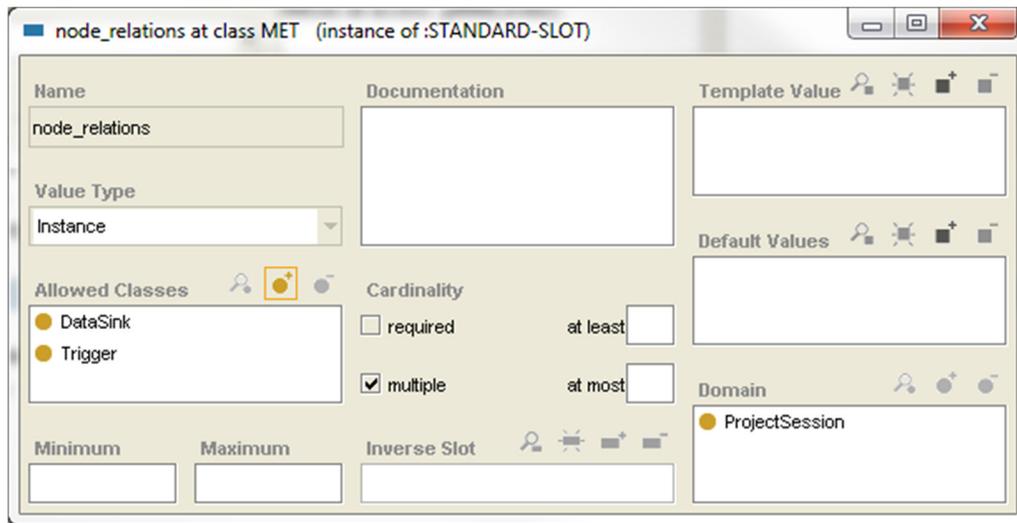


Figura 45. Edición del campo node\_relations.

Una vez que han sido editados los campos que el usuario requiere personalizar, solo resta crear una instancia del componente en Protégé (ver Figura 46, pestaña Instances).

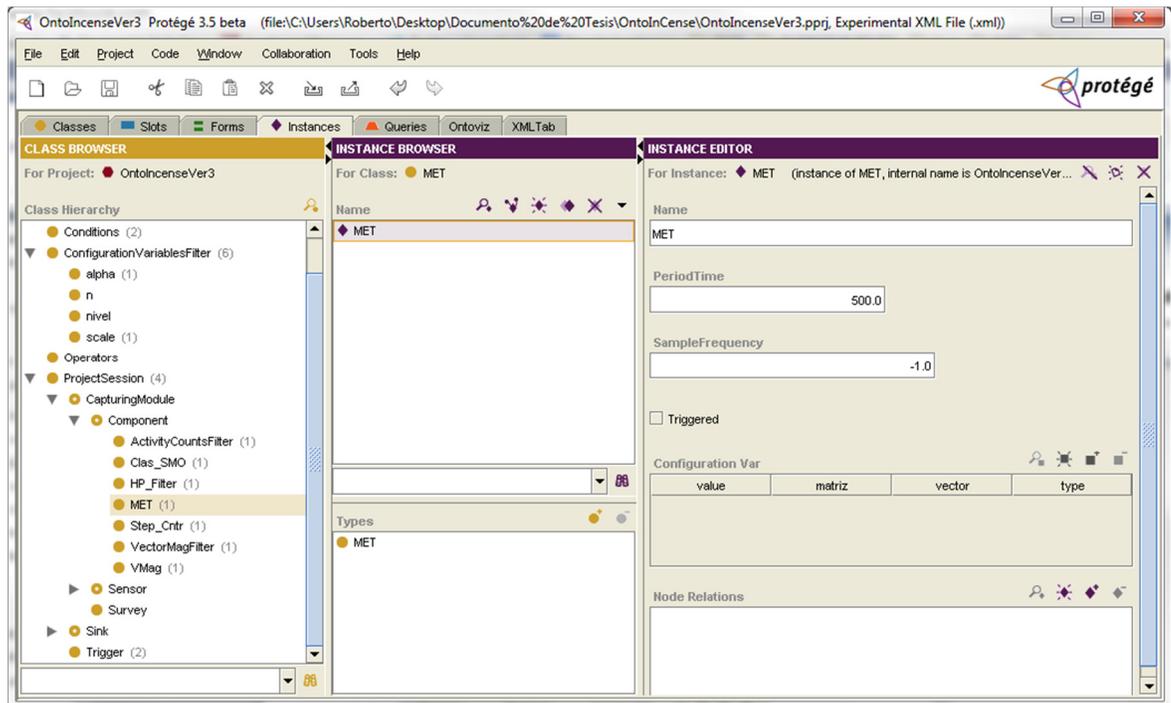
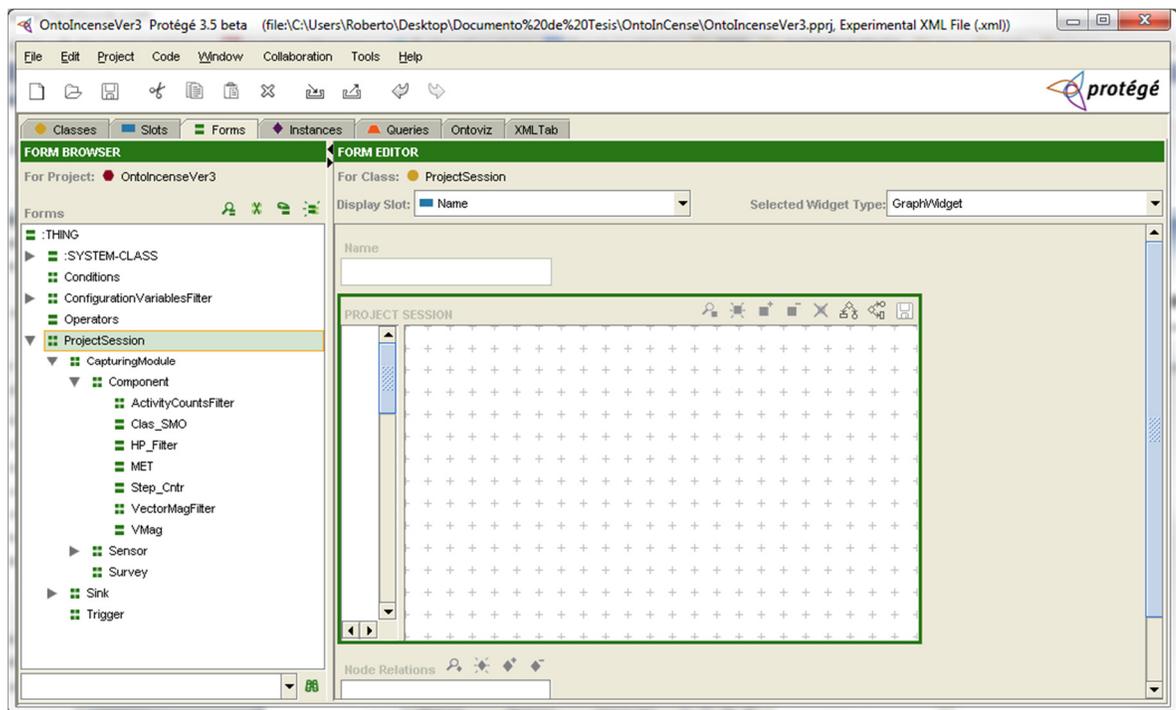


Figura 46. Ventana de creación de instancias en Protégé.

Asignar la forma del componente en el editor gráfico de Protégé. Por defecto, Protégé le asigna a todos los nuevos componentes creados una forma circular azul para su representación gráfica. Como se mencionó anteriormente, se propone que en Protégé todos los componentes tengan forma cuadrada y relleno amarillo puesto que resulta difícil designar una forma idéntica a la de la notación gráfica de InCense. Para modificar la forma que por defecto tiene MET se debe editar en la ventana Forms de Protégé. Se selecciona la clase ProjectSession y se hace doble clic sobre el editor gráfico que se muestra en la Figura 47.



**Figura 47. Ventana Forms para editar la forma del componente.**

Esta acción abre un formulario para personalizar el editor gráfico, en la pestaña Nodes se encuentran listados todos los elementos de la ontología. Se selecciona el componente que se acaba de crear, en este caso fue MET. En primera instancia se puede observar que el campo *shape* tiene configurado elipse. Se editó para cambiarlo por rectángulo y como color se seleccionó el amarillo. La sección Optional Connector Slot también se editó para indicar que como campo de conexión se utilice *node\_relations* y la conexión se muestre gráficamente como una flecha. La Figura 48 muestra dicho formulario.

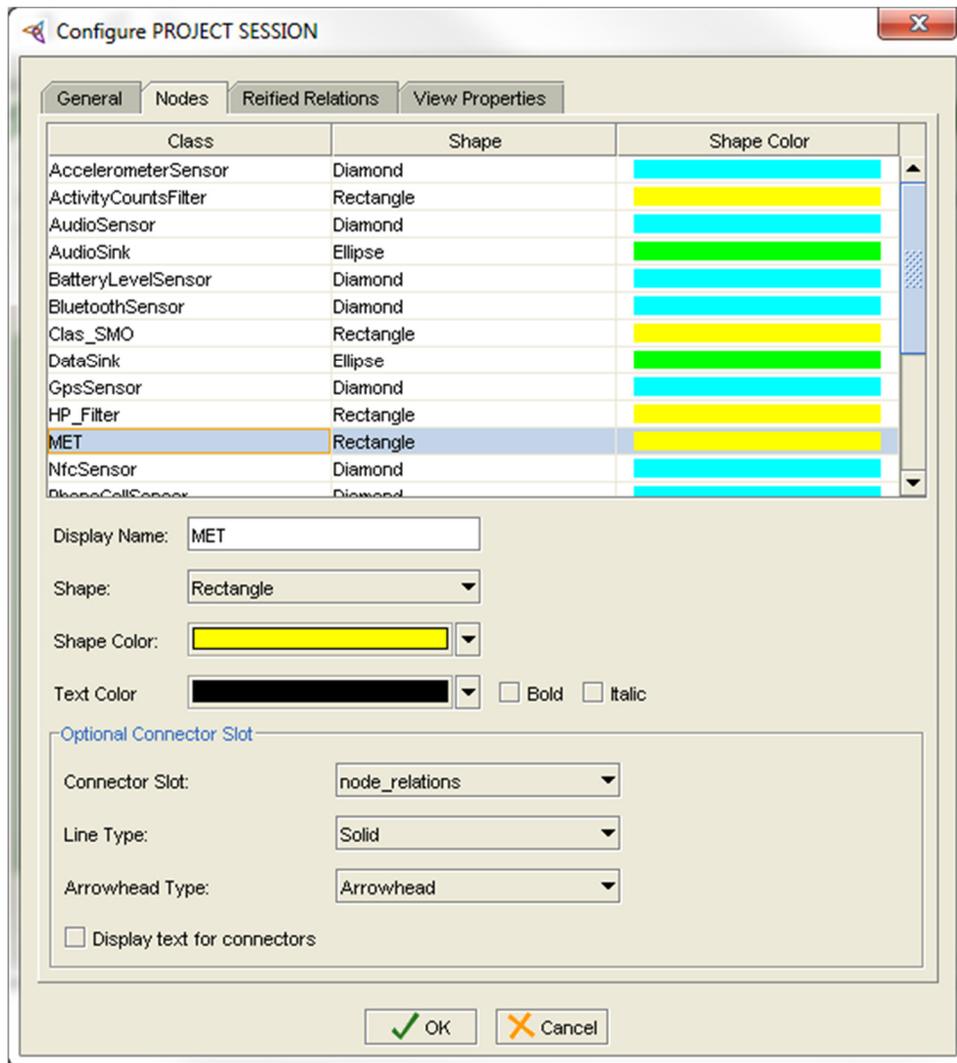


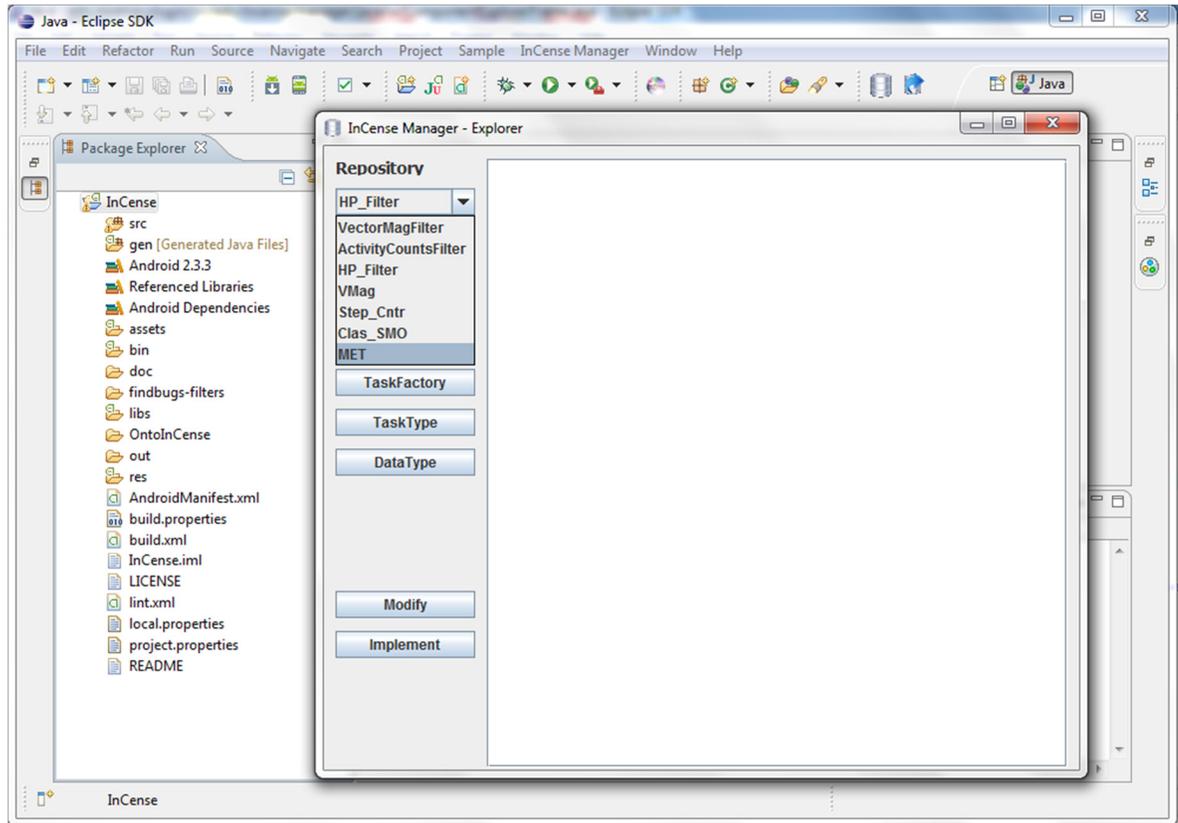
Figura 48. Edición de la forma del componente y conector utilizado en el editor gráfico.

Una vez finalizado el registro del componente en la ontología se cerró Protege para realizar la edición del componente en InCense Manager y asignar el código del procesamiento de los datos y registrar la variable de salida.

#### 5.2.4. Edición e implementación del componente

En esta actividad es donde se genera el código fuente a partir de los templates traducidos. El primer paso consiste en seleccionar el proyecto de la Aplicación Móvil desde Eclipse y luego abrir Explorer desde el plug-in de InCense Manager. Esto cargará la pantalla principal de Explorer como se muestra en la Figura 49. Se observa como en la lista se

muestran todos los componentes que han sido registrados en OntoInCense. Para editar alguno se debe seleccionar su nombre de la lista, en este caso se editó MET.



**Figura 49.** Explorer con el nuevo componente MET en la lista.

Presionando sobre el botón Modify se abre el formulario de edición de componentes: Modify Component Wizard. La Figura 50 muestra la edición de MET. En el campo Component Code solo se pega el código de procesamiento de datos que se creó en el la actividad de diseño, a la derecha del formulario está el mecanismo para agregar variables de salida, en este caso *metLevels*. Para regresar a la pantalla principal el usuario debe presionar el botón Save.

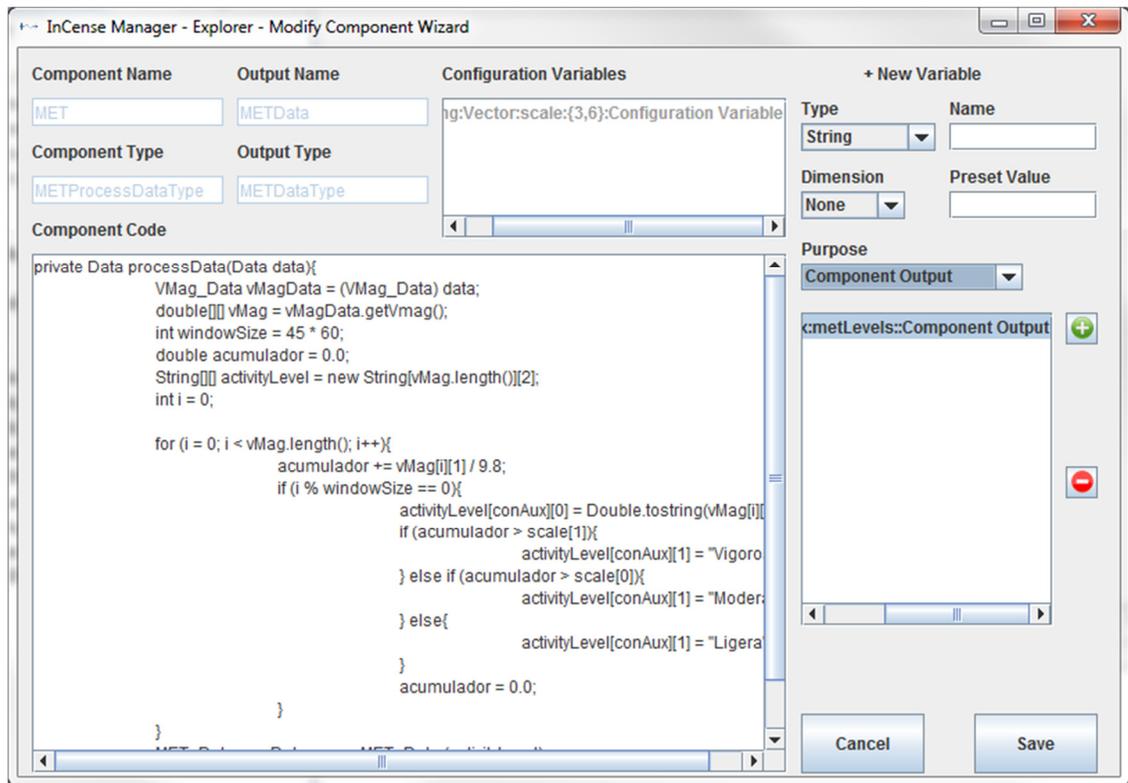


Figura 50. Edición de MET.

Para realizar la implementación del componente solamente se requiere presionar sobre el botón Implement de la ventana principal de Explorer y se genera el código fuente de las 5 clases que conforman el componente (véase Figura 49). Explorer inserta las nuevas clases en sus paquetes correspondientes en la Aplicación Móvil, por supuesto que esto sucede de manera transparente para el usuario. Si el usuario lo desea, también puede visualizar el código fuente completo de las 5 clases en las que se distribuye el componente, solo debe presionar sobre el botón que corresponde a la clase que desee visualizar en la ventana principal de Explorer (véase Figura 49).

### 5.2.5. Actualizar instancia de Aplicación Móvil

Cada vez que se agrega un componente nuevo a InCense, es necesario actualizar la Aplicación Móvil en cada teléfono que este participando en el proyecto de sensado. De otra manera, cuando llegue un nuevo archivo de configuración para la Aplicación Móvil que utilice el nuevo componente, la Aplicación Móvil no será capaz de utilizar dicho componente puesto que se trataría de versión que no lo tiene implementado. Para lograr

esto, Eclipse tiene la capacidad de exportar los proyectos Android a archivos APK que son archivos de aplicación para la plataforma Android. Es responsabilidad del usuario distribuir el archivo de la manera que considere más conveniente ya que InCense no cuenta de momento con un mecanismo automático de actualización de la Aplicación Móvil en los dispositivos participantes.

Realizando las cinco actividades anteriores un usuario puede crear cualquier componente utilizando la arquitectura extendida. En el siguiente capítulo se realiza un análisis de la aportación de la arquitectura extendida a la plataforma de sensado (InCense) y por extensión al campo de sensado con teléfonos móviles. Además se discuten oportunidades de mejora para la arquitectura extendida como trabajo futuro.

### **5.3. Resumen del capítulo**

En este capítulo se presentó de manera detallada el nuevo proceso de creación de componentes basado en la arquitectura extendida. A la vez que se evidenciaron las mejoras logradas a través de la implementación del componente para estimar el nivel de actividad física de un individuo (MET).

En el siguiente capítulo se discuten los resultados logrados y se proponen actividades a realizar como trabajo futuro para mejorar la plataforma de sensado.

## Capítulo 6

---

### Conclusiones y trabajo futuro

Los teléfonos móviles han permeado en la vida diaria de las personas, además de que existe una tendencia a incorporar en estos dispositivos sensores como acelerómetro, brújula digital, GPS, micrófono entre otros. Esto ofrece la posibilidad de sensar a las personas en su vida cotidiana y de manera no invasiva puesto que ya están habituados a portar teléfonos móviles. Esta posibilidad de estudiar a las personas a través de un teléfono móvil ha dado lugar a un nuevo campo de investigación: el sensado con teléfonos móviles.

El potencial del sensado con teléfonos móviles se extiende a varias áreas entre las que se encuentran las aplicaciones de tipo médico, urbano y social. De ahí que existen individuos interesados en aprovechar este potencial y cuya área de experiencia esta fuera de las ciencias de la computación (médicos, psicólogos, urbanistas y otros).

Por otro lado, para realizar campañas de sensado con teléfonos móviles se requiere un alto nivel de habilidad técnica para diseñar e implementar una aplicación en una plataforma de teléfono móvil. En la mayoría de los casos se implementa una Aplicación Móvil especialmente diseñada para una campaña de sensado en específico que recolecta los datos obtenidos con los sensores para almacenarlos y realizar un análisis posterior. Sin embargo, diseñar e implementar una Aplicación Móvil diferente para cada campaña de sensado representa un esfuerzo considerable. De ahí que existe la oportunidad en este campo de investigación de crear plataformas de sensado que sean flexibles y de propósito general, características que permitirán su reutilización en diferentes campañas de sensado.

InCense es una plataforma de sensado con teléfonos móviles que permite la adición de nuevos componentes que permiten el procesamiento de los datos recolectados. El presente trabajo de investigación reporta una modificación a la arquitectura de InCense con el objetivo de facilitar la adición de componentes y el despliegue de campañas de sensado. Se propuso como objetivo de este trabajo de investigación que la arquitectura extendida

facilite el proceso de adición de componentes de manera que el usuario no requiere de un alto nivel de habilidad técnica.

En este capítulo se discuten los resultados logrados. Se presenta también el trabajo por realizar y los retos a resolver.

### **6.1. Conclusiones**

Durante el desarrollo del trabajo de investigación se logró dar respuesta a las tres preguntas de investigación planteadas:

*1. ¿Cuál es el conjunto de componentes que debería incluir InCense, dada la frecuencia en que son usados en la literatura previa en el área?*

Para dar respuesta a esta pregunta se realizó una exploración de la literatura concerniente al sensado con teléfonos móviles con el objetivo de identificar a los componentes que resultaran relevantes para el presente trabajo de investigación. A partir de este análisis se consolidó una lista de componentes agrupados por tipo de componente resultando 17 categorías ordenadas de acuerdo a la frecuencia con la que se reportan en literatura. De esta lista, se seleccionaron 3 componentes para ser diseñados e implementados sobre la plataforma de sensado (InCense): Detección de actividad de voz humana, Estimación de nivel de actividad física, y Conteo de pasos.

*2. ¿Qué mecanismos pueden facilitar la adición componentes a InCense si se toma en consideración que el usuario de InCense posee bajo nivel de habilidad técnica en programación?*

A partir de la experiencia obtenida del diseño e implementación de componentes, se estudió el proceso de adición de componentes a InCense utilizando la arquitectura original. Se observó que la mayor dificultad para un usuario que desea utilizar InCense es la necesidad de un alto nivel de habilidad técnica en programación. Especialmente por la necesidad de modificar de manera directa el código fuente de la plataforma.

Tomando como base la arquitectura original, se propusieron modificaciones a la misma. Estas modificaciones facilitan el proceso de adición de componentes. La arquitectura resultante se denominó arquitectura extendida. La principal aportación de esta arquitectura extendida es la inclusión de OntoInCense e InCense Manager. Ambas herramientas de software que se integran al ambiente de desarrollo Eclipse para asistir el proceso de adición de componentes a InCense.

*3. ¿Qué abstracciones de programación son adecuadas para facilitar la implementación de nuevos componentes en InCense?*

Como respuesta a esta pregunta se propuso el desarrollo de un componente genérico. A partir de esta conceptualización de un componente se creó una clase en java que en conjunto con JET permitió la generación automática de componentes. Gracias a esta combinación ya no se requiere que el usuario edite directamente el código fuente de InCense. Sin embargo, aún es necesario que provea de un mínimo de código fuente para procesar datos. Este debe ser código fuente escrito en java y es parte del componente genérico.

En conclusión, gracias a la extensión de la arquitectura original se logró disminuir la exposición del usuario al código fuente de InCense. En consecuencia se habilita a usuarios con poca habilidad técnica en programación para usar la plataforma de sensado. En otras palabras, se consiguió facilitar la adición de nuevos componentes al kit de investigación así como la creación y despliegue de nuevos proyectos de sensado. Enseguida se listan las aportaciones hechas con el presente trabajo de tesis.

#### **6.1.1. Aportaciones**

Como principal aportación está la propuesta e implementación de dos herramientas orientadas a facilitar el diseño y adición de componentes a InCense: InCense Manager y OntoInCense.

Se integró InCense Manager así como OntoInCense al ambiente de desarrollo Eclipse. Esto facilita la adopción de la plataforma de sensado (InCense) puesto que este ambiente de desarrollo es uno de los más populares además de ser gratuito.

Se realizó una revisión de la literatura para crear una lista de aquellos componentes que son de mayor interés de acuerdo a la frecuencia con que son utilizados en proyectos de sensado con dispositivos móviles.

Se implementaron tres componentes reportados en la literatura en InCense, que a su vez se ejecuta sobre la plataforma android para dispositivos móviles. Los tres componentes están documentados en los capítulos 3 y 4.

Adicionalmente, parte de este trabajo de investigación fue presentado y publicado en las memorias del *1st International Workshop on Ubiquitous Mobile Instrumentation 2012 – UbiMI '12* (Rodríguez et al., 2012).

## **6.2. Trabajo futuro**

Como trabajo futuro se propone la revisión de la interfaz de usuario para InCense Manager con el objetivo de hacerla más amigable aplicando los lineamientos pertinentes de usabilidad.

Se sugiere también realizar campañas de sensado en colaboración con investigadores en áreas como la salud, transporte o sociología. Esto con el objetivo de observar la facilidad de adopción de InCense con la arquitectura extendida por investigadores que sin contar con mucho conocimiento de programación, estén interesados en realizar campañas de sensado.

También se debe considerar la creación de un mecanismo que permita compartir componentes entre distintos usuarios de la plataforma de sensado (InCense). Tomando ventaja de la propuesta del componente genérico esta tarea debería ser tan simple como compartir un archivo de texto en formato XML. En su defecto permitir consultas a servidores de bases de datos externos que contienen repositorios de componentes de otros usuarios. Esta tarea implica la implementación de rutinas de exportación e importación de componentes en InCense Manager.

Por último, aunque InCense Manager elimina la necesidad de que el usuario conozca el código fuente de InCense, todavía es necesario que escriba el código fuente en lenguaje Java para procesar los datos (componente genérico). En este sentido se propone crear una

capa más de abstracción que permita al usuario procesar los datos con instrucciones de más alto nivel de abstracción (más fácil para el usuario) a manera de framework o API como se propone en (“on{X} - automate your life,” 2012).

## Referencias bibliográficas

- Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1), 37–66.
- Android Developer: Sensor Event. (2012). Android. Retrieved October 19, 2012, from <http://developer.android.com/reference/android/hardware/SensorEvent.html>
- Aram, S., Troiano, A., & Pasero, E. (2012). Environment sensing using smartphone. *2012 IEEE Sensors Applications Symposium Proceedings* (pp. 1–4). Torino: IEEE.
- Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R., & Havinga, P. (2010). Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. *2010 23rd International Conference on Architecture of Computing Systems (ARCS)* (pp. 1–10). Hannover: VDE.
- Blom, J., Gatica-Perez, D., & Kiukkonen, N. (2011). People-Centric Mobile Sensing with a Pragmatic Twist: from Behavioral Data Points to Active User Involvement. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11* (pp. 381–384). Stockholm: ACM Press.
- Campbell, A., & Choudhury, T. (2012). From Smart to Cognitive Phones. *IEEE Pervasive Computing*, 11(3), 7–11.
- Checkland, P. (2000). Soft systems methodology: a thirty year retrospective. *Systems Research and Behavioral Science*, 17, 11–58.
- Choudhury, T., Borriello, G., Consolvo, S., Haehnel, D., Harrison, B., Hemingway, B., Hightower, J., et al. (2008). The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, 7(2), 32–41.
- Chronis, I., Madan, A., & Pentland, A. (Sandy). (2009). SocialCircuits: The Art of Using Mobile Phones for Modeling Personal Interactions. *Proceedings of the ICMI-MLMI '09 Workshop on Multimodal Sensor-Based Systems and Mobile Phones for Social Computing - ICMI-MLMI '09* (pp. 1–4). Cambridge, USA: ACM Press.
- Consolvo, S., Libby, R., Smith, I., Landay, J. A., McDonald, D. W., Toscos, T., Chen, M. Y., et al. (2008). Activity Sensing in the Wild: A Field Trial of UbiFit Garden. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1797–1806). Florence: ACM Press.
- Crouter, S. E., Churilla, J. R., & Bassett, D. R. (2006). Estimating energy expenditure using accelerometers. *European journal of applied physiology*, 98(6), 601–612.

- Crouter, S. E., Clowers, K. G., & Bassett, D. R. (2006). A novel method for using accelerometer data to predict energy expenditure. *Journal of applied physiology*, *100*(4), 1324–1331.
- Das, T., Mohan, P., Padmanabhan, V. N., Ramjee, R., & Sharma, A. (2010). PRISM: Platform for Remote Sensing using Smartphones. *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10* (pp. 63–76). San Francisco: ACM Press.
- Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, *5*(1), 4–7.
- Eagle, N., & Pentland, A. (Sandy). (2005). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, *10*(4), 255–268.
- Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, *43*(5-6), 907–928.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18.
- Herrera, J. C., Work, D. B., Herring, R., Ban, X. (Jeff), Jacobson, Q., & Bayen, A. M. (2010). Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment. *Transportation Research Part C: Emerging Technologies*, *18*(4), 568–583.
- Härtel, S., Gnam, J.-P., Löffler, S., & Bös, K. (2010). Estimation of energy expenditure using accelerometers and activity-based energy models—validation of a new device. *European Review of Aging and Physical Activity*, *8*(2), 109–114.
- JET Tutorial. (2012). Retrieved May 14, 2012, from [http://www.eclipse.org/articles/Article-JET/jet\\_tutorial1.html](http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html)
- Kanhere, S. S. (2011). Participatory Sensing: Crowdsourcing Data from Mobile Smartphones in Urban Spaces. *2011 IEEE 12th International Conference on Mobile Data Management* (Vol. 2, pp. 3–6). Sydney: IEEE.
- Kay, M., Santos, J., & Takane, M. (2011). *mHealth: new horizons for health through mobile technologies: second global survey on eHealth*. (K. Lashley, Ed.) (Vol. 3, p. 13). Geneva: Organización Mundial de la Salud.
- Khan, W. Z., Xiang, Y., Aalsalem, M. Y., & Arshad, Q. (2012). Mobile Phone Sensing Systems: A Survey. *IEEE Communications Surveys & Tutorials*, 1–26.

- Knight, J. F., Bristow, H. W., Anastopoulou, S., Baber, C., Schwirtz, A., & Arvanitis, T. N. (2006). Uses of accelerometer data collected from a wearable system. *Personal and Ubiquitous Computing*, *11*(2), 117–132.
- Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, *12*(2), 74–82.
- Lane, N. D. (2012). Community-Aware Smartphone Sensing Systems. *IEEE Internet Computing*, *16*(3), 60–64.
- Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, *48*(9), 140–150.
- Lane, N., Mohammad, M., Lin, M., Yang, X., Lu, H., Ali, S., Doryab, A., et al. (2011). BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing. *Proceedings of the 5th International ICST Conference on Pervasive Computing Technologies for Healthcare* (pp. 1–8). Dublin: IEEE.
- Lee, M.-W., Khan, A. M., & Kim, T.-S. (2011). A single tri-axial accelerometer-based real-time personal life log system capable of human activity recognition and exercise information generation. *Personal and Ubiquitous Computing*, *15*(8), 887–898.
- Loredo Medina, R. (2011). *Sistema de monitoreo de la marcha como apoyo al cuidado de adultos mayores (Tesis de maestría)*. CICESE.
- Lu, H., Pan, W., Lane, N. D., Choudhury, T., & Campbell, A. T. (2009). SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones. *Proceedings of the 7th international conference on Mobile systems, applications, and services - Mobisys '09* (pp. 165–178). Kraków: ACM Press.
- Madan, A., Cebrian, M., Lazer, D., & Pentland, A. (2010). Social sensing for epidemiological behavior change. *Proceedings of the 12th ACM international conference on Ubiquitous computing - UbiComp '10* (pp. 291–300). Copenhagen: ACM Press.
- Miluzzo, E., Lane, N. D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S. B., et al. (2008). Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08* (pp. 337–350). Raleigh: ACM Press.
- Moran, D. S., Heled, Y., & Gonzalez, R. R. (2004). Metabolic rate monitoring and energy expenditure prediction using a novel actigraphy method. *Medical science monitor*, *10*(11), 117–120.

- Mun, M., Boda, P., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., et al. (2009). PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. *Proceedings of the 7th international conference on Mobile systems, applications, and services - Mobisys '09* (p. 55). Kraków: ACM Press.
- Nike + iPod. (2012). Retrieved November 10, 2012, from <http://www.apple.com/mx/ipod/nike/>
- on{X} - automate your life. (2012). Retrieved November 11, 2012, from <https://www.onx.ms>
- Pentland, A. (Sandy). (2005). Socially aware media. *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05* (pp. 690–695). Singapore: ACM Press.
- Platt, J. C. (1999). Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods: support vector learning* (pp. 185–208). Cambridge, USA: MIT Press.
- Protégé. (2012). Retrieved October 19, 2012, from <http://protege.stanford.edu/>
- Puyau, M. R., Adolph, A. L., Vohra, F. A., Zakeri, I., & Butte, N. F. (2004). Prediction of Activity Energy Expenditure Using Accelerometers in Children. *Medicine & Science in Sports & Exercise*, 36(9), 1625–1631.
- Pérez Gamboa, M. (2012). *Herramienta móvil de adquisición de información del comportamiento en poblaciones (Tesis de maestría)*. CICESE.
- Pérez, M., Castro, L. A., & Favela, J. (2011). InCense : A Research Kit to Facilitate Behavioral Data Gathering from Populations of Mobile Phone Users. *5th International Symposium of Ubiquitous Computing and Ambient Inteligence (UCAmI)* (pp. 1–8). Mérida.
- Rodríguez, M. D., Martínez, R., Pérez, M., Castro, L. A., & Favela, J. (2012). Using ontologies to reduce user intervention to deploy sensing campaigns with the InCense toolkit. *Paper presented at the 1st International Workshop on Ubiquitous Mobile Instrumentation at the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12* (pp. 741–744). Pittsburgh: ACM Press.
- Sorber, J. M., Shin, M., Peterson, R., & Kotz, D. (2012). Plug-n-Trust: Practical Trusted Sensing for mHealth. *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12* (pp. 309–322). Low Wood Bay: ACM Press.

- Tanveer, W., M., M.-E. A., G., E.-I., & Aslam, M. (2010). Sensing WithSense - An Intelligent Interface for Participatory Sensing. *2010 Fifth International Conference on Software Engineering Advances* (pp. 400–405). Nice: IEEE.
- Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., & Eriksson, J. (2009). VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones. *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems - SenSys '09* (pp. 85–98). Berkeley: ACM Press.
- Wang, J., Chen, R., Sun, X., She, M. F. H., & Wu, Y. (2011). Recognizing Human Daily Activities From Accelerometer Signal. *Procedia Engineering, 15*, 1780–1786.
- Yamada, Y., Yokoyama, K., Noriyasu, R., Osaki, T., Adachi, T., Itoi, A., Naito, Y., et al. (2009). Light-intensity activities are important for estimating physical activity energy expenditure using uniaxial and triaxial accelerometers. *European journal of applied physiology, 105*(1), 141–152.
- Zhao, N. (2012). Full-featured pedometer design realized with 3-Axis digital accelerometer. *Analog Dialogue*. Retrieved October 23, 2012, from [http://www.analog.com/static/imported-files/tech\\_articles/pedometer.pdf](http://www.analog.com/static/imported-files/tech_articles/pedometer.pdf)
- Zhou, P., Zheng, Y., & Li, M. (2012). How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing. *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12* (pp. 379–392). Low Wood Bay: ACM Press.