Procesamiento de Eventos Complejos para la determinación de rutas en ciudades inteligentes*

Alejandro Pérez-Vereda, David Bandera, and Carlos Canal

Universidad de Málaga, España
apvereda@uma.es, dbandera@lcc.uma.es, canal@lcc.uma.es

Resumen Una de las principales características de una ciudad inteligente es la publicación de datos abiertos que monitorizan diferentes aspectos de la misma, desde sistemas de movilidad a niveles de polen. Sin embargo, esta información se ofrece a los ciudadanos de forma genérica y carente de contexto. Sin el concurso de las personas en la generación y recogida de información, no es posible tener en cuenta sus preferencias y necesidades, ni el uso que hacen de los servicios. Para reducir este fenómeno y contextualizar los datos abiertos con información acerca de los ciudadanos, diseñamos anteriormente la arquitectura de referencia People as a Service, que permite recopilar información de los usuarios a partir de sus *smartphones*. En este trabajo vamos un paso más allá en la integración de un motor de inferencia para esta arquitectura usando la tecnología de Procesamiento de Eventos Complejos. En particular, recopilamos información de los sensores del *smartphone* para analizarla, transformándola en conocimiento sobre los hábitos del usuario. La posterior combinación de este conocimiento con los datos abiertos de la ciudad, permitirá que sus servicios puedan adaptarse a cada persona.

Keywords: Ciudades Inteligentes · People as a Service · Procesamiento de Eventos Complejos.

1. Introducción

Actualmente, las ciudades inteligentes están inmersas en proyectos de sensorización para monitorizar y recabar información de muy diversa índole: disponibilidad del transporte público, tráfico, índices de contaminación, etc. El objetivo principal de esta práctica es el de mejorar la oferta de servicios desplegados en la ciudad y hacer más eficiente la gestión de la misma y sus recursos [12]. Parte de los datos recolectados pasan también a ser un servicio en sí mismos, siendo ofertados dentro del catálogo de datos abiertos de la ciudad.

Esta práctica podría ser muy útil, pero los ciudadanos que habitan en estas ciudades no han experimentado una mejora sustancial en su vida cotidiana. Para Cohen [5], este enfoque actual permite únicamente mejorar el crecimiento de la economía, pero debería extenderse de forma que también permita la mejora de

* Este trabajo ha sido financiado por el Gobierno de España a través del proyecto TIN2015-67083-R (MINECO/FEDER)

la calidad de vida de los ciudadanos. De hecho, esta es la razón de ser y uno de los pilares sobre los que se sustenta el concepto de *smart city* [11].

No obstante, la información recolectada por la ciudad inteligente carece de datos sobre las personas que habitan en la ciudad y de cómo estas hacen uso de sus recursos y servicios. Esto provoca que los datos abiertos ofertados también carezcan en gran medida de un valor real y de utilidad para los ciudadanos. Se trata de información genérica y descontextualizada, lo que dificulta su uso y comprensión por parte de los usuarios de a pie.

En trabajos anteriores, los autores de este artículo abogaron por la adopción de un modelo de computación social denominado *People as a Service* (PeaaS) [8]. Este modelo fue ideado para mejorar la integración de las personas con la Internet of Things (IoT). Se trata de una arquitectura de referencia que habilita a los *smartphones* para crear y gestionar perfiles virtuales con información acerca de los hábitos, movimientos y preferencias de sus usuarios. Así, creamos un ecosistema de información compartida por los usuarios, que en conjunción con los datos abiertos, permita avanzar hacia una ciudad inteligente con un verdadero valor añadido para sus habitantes. De esta forma, se puede lograr la adecuación de servicios y recursos gracias a la información obtenida como retroalimentación de su uso.

En este trabajo damos un paso más en la implementación de esta arquitectura por medio de un motor de inferencia capaz de combinar los datos recibidos desde los diversos sensores del *smartphone* y analizarlos en tiempo real para obtener información de mayor valor semántico. Este motor está basado en la tecnología de Procesamiento de Eventos Complejos (CEP, de sus siglas en inglés) [10]. Concretamente, en este trabajo los datos que suministramos al motor CEP están relacionados con el posicionamiento GPS, centrándonos en la obtención de un perfil virtual compuesto por las rutinas de desplazamiento del usuario.

La estructura de este artículo es la siguiente. En la sección 2 ofrecemos una introducción al estado del arte en cuanto a estas tecnologías. En la sección 3 presentamos el motor de inferencia y el análisis de los datos recogidos por el teléfono para elaborar el perfil de desplazamientos de su usuario. Más adelante, en la sección 4 mostramos los resultados obtenidos tras este análisis acompañados de una discusión sobre líneas futuras de trabajo y mejoras a llevar a cabo. Finalmente, encontramos las conclusiones de este trabajo.

2. Estado del Arte

En la literatura encontramos diversos trabajos que concuerdan con la idea de que las personas deberían tener un papel más importante en las ciudades inteligentes. En [14, 15] se apuesta por el paradigma de *Social Computing* para modelar y analizar el comportamiento social de las personas y desarrollar una tecnología responsiva que tenga en cuenta su contexto. De hecho, existen trabajos que tratan de obtener este conocimiento social colectivo por medio de una plataforma de *crowdsensing* [4].

La recopilación de datos de redes sociales es un ámbito de estudio que cuenta con numerosos trabajos de investigación recientes que tratan de obtener este tipo de conocimiento social a partir de las interacciones de las personas con estas redes. En [9] se estudia el impacto que tuvo el famoso juego Pokemon Go sobre los restaurantes de la plataforma Yelp. En [1] también encontramos un análisis de redes sociales basadas en geolocalización para hacer recomendaciones a grupos de usuarios sobre dónde ir a comer. Ambos trabajos podrían solucionarse de una forma más sencilla con una arquitectura como PeaaS en la que el teléfono de cada usuario es el encargado de analizar de forma automática las interacciones que el usuario realiza a través del propio teléfono gracias a un motor CEP.

La inferencia de patrones de desplazamiento ya se ha abordado anteriormente en [2, 13]. Ambos trabajos tratan de supervisar los desplazamientos de enfermos de Alzheimer. En concreto [13] es un trabajo de los autores de este mismo artículo en el que se lleva a cabo una prueba de concepto de la arquitectura PeaaS pero con un algoritmo *ad hoc* carente de capacidad de respuesta en tiempo real. En nuestro trabajo actual pretendemos hacer evolucionar la arquitectura y ofrecer algoritmos más generales y escalables con propiedades más interesantes.

Se denomina *mobile* CEP al paradigma emergente en el que la tecnología CEP es desplegada en un dispositivo móvil [6]. Este paradigma tiene una serie de ventajas significativas a la hora de implementar el sistema. La primera tiene que ver con el conocimiento del contexto gracias a los datos provenientes de los diferentes sensores del *smartphone*, que además, no precisan de ningún tipo de comunicación con un servidor externo. A raíz de aquí, nace la ventaja acerca de la privacidad de la información y correlación de datos en el propio dispositivo.

Birth et al. [3] implementan una aplicación Android haciendo uso del motor CEP en un *smartphone* siguiendo una arquitectura dirigida por eventos para proporcionar información en tiempo real sobre desplazamientos multimodales (usando diferentes medios de transporte), demostrando así la viabilidad de este tipo de sistemas. Sin embargo, su principal limitación es la arquitectura utilizada, que en ningún momento saca partido al *smartphone* tratando de averiguar información sobre el usuario, sino que este debe configurarla manualmente, quedando la aplicación limitada únicamente a ese propósito.

Como hemos visto, en la actualidad existen diversos trabajos basados en el análisis del comportamiento de las personas que trabajan en ámbitos de geolocalización. Las ventajas del sistema que proponemos son su escalabilidad, portabilidad y facilidad de uso. Nuestra propuesta no tiene limitaciones de ámbito ni área, es capaz de reaccionar en tiempo real y, además, el objetivo es diseñarlo para que sea de bajo coste y fácilmente portable.

3. Motor de Inferencia

La capacidad computacional que tienen hoy en día los *smartphones* los convierte en dispositivos más que capaces de realizar diferentes tareas que requieran más recursos, antes relegadas únicamente a los ordenadores. Así, también son capaces de incluir un motor de inferencia con tecnología CEP en ejecución para

la implementación de la arquitectura PeaaS. La principal característica de esta arquitectura es que está diseñada para ejecutarse íntegramente en el *smartphone* otorgándole la capacidad de inferir y compartir un perfil virtual del usuario, convirtiéndose así en una interfaz de este con su entorno.

CEP [10] es una tecnología que proporciona una arquitectura y los mecanismos necesarios para el descubrimiento y análisis de situaciones de interés o eventos complejos mediante la identificación de patrones de eventos más simples. Este análisis se realiza sobre un flujo de datos provenientes de diferentes fuentes o sensores, así como de la información ya extraída anteriormente. CEP ofrece herramientas para detección de causalidad, cronología etc. Se entiende como evento simple cualquier dato atómico e indivisible asociado al momento en el tiempo en el que ocurrió. A partir de la composición y correlación de varios eventos, llegamos a la inferencia de eventos complejos, situaciones con mayor valor semántico.

El motor CEP interactúa de forma directa con el perfil virtual del usuario, que es la pieza clave de PeaaS, extrayendo información nueva, y componiendo la información que ya ha extraído para crear conocimiento. Una de las principales ventajas de CEP es su funcionamiento en tiempo real, que lo diferencia del resto de software de análisis de eventos, reduciendo latencias en la toma de decisiones. Esta característica convierte a CEP en una tecnología apropiada para sistemas asíncronos que precisan una respuesta rápida para reaccionar a cambios en el entorno. Otra característica interesante de CEP, es la capacidad de analizar la información en flujo sin necesidad de almacenarla. La inferencia de información es inmediata filtrando los datos y obteniendo las situaciones de interés de forma directa. Por esto, CEP es también adecuado para el análisis de grandes flujos de datos provenientes de multitud de sensores diferentes.

El motor CEP más usado hoy en día es Esper. Esper es código abierto y proporciona un lenguaje (EPL) para la formulación de patrones de eventos. Ya ha existido una implementación del motor CEP de Esper en Android [7]. Este proyecto se descontinuó, no siendo ya compatible con las versiones actuales de Android, pero sirve para demostrar que, gracias a la capacidad de análisis en tiempo real con bajo consumo de recursos y a la no necesidad de almacenar todos los datos, es una tecnología idónea para su uso en un teléfono móvil. Debido a la discontinuidad del proyecto, nos vimos obligados a realizar la prueba de concepto de CEP en un ordenador personal. Sin embargo, las conclusiones y avances que aportamos en este artículo son perfectamente extrapolables para su uso en una versión móvil. A continuación presentaremos el análisis llevado a cabo con este motor para la obtención de información sobre los desplazamientos de los usuarios.

3.1. Recogida de datos

El análisis realizado se centra en itinerarios urbanos. A partir de datos GPS trataremos de obtener información sobre las desplazamientos del usuario a lo largo del día: paradas intermedias, medios de transporte, etc. Para la obtención de estos datos hemos desarrollado una aplicación Android que crea un flujo de información a partir de una monitorización del sensor de GPS del móvil.

La aplicación está desarrollada usando los servicios de Google Maps para obtener el geoposicionamiento del usuario. La aplicación monitoriza el GPS del teléfono cada vez que el usuario se desplaza 35 metros (configurable). De esta forma, cada vez que el usuario se desplaza dicha distancia, la aplicación captura su posición y velocidad y las almacena junto con un *timestamp*. Como puede observarse, esta forma de capturar los datos nos da ciertas ventajas a la hora de analizarlos y en cuanto a consumo de memoria y batería en el móvil, ya que mientras el usuario está parado, la aplicación permanece en reposo y no obtenemos medidas innecesarias.

3.2. Análisis de desplazamientos

El motor CEP precisa un flujo de eventos entrantes para analizarlos, no puede analizar datos en bruto. Por ello, definimos un evento simple para representar los datos que recogemos con la aplicación. Esta abstracción a eventos es necesaria para cada una de las entidades del sistema. Hemos denominado a este evento básico *Point* y tiene como atributos la latitud, longitud, velocidad y fecha y hora a la que se tomó la medida.

Para la implementación de esta prueba de concepto hemos diseñado patrones de eventos para detectar las paradas, giros o cambios de velocidad del usuario a lo largo de sus desplazamientos. Para la definición de estos patrones utilizamos una implementación Java que hace uso de una API para el mencionado Lenguaje EPL proporcionado por Esper. Se trata de un lenguaje declarativo, basado en el estándar SQL-92, capaz de trabajar con altas frecuencias de eventos temporales.

El primer evento que detectamos son las paradas. Se dan cuando entre dos puntos *Point* hay una ventana de más de 150 segundos (configurable). El resultado es un evento llamado *MoveStop* con la latitud, longitud y marca de tiempo en la que sucedió la parada.

Una vez detectadas las paradas que realiza el usuario, la detección de los inicios de desplazamiento es inmediata. El primer punto que encontremos tras haber estado parados es el principio de un nuevo desplazamiento. El patrón detecta y lanza al flujo de eventos de entrada los eventos *MoveStart* que encuentre.

Construyendo sobre los dos patrones anteriores ya podemos extraer información con más significado semántico como son los desplazamientos y los lugares que visita el usuario a lo largo del día. Según la forma en que unamos los eventos anteriores obtendremos un patrón para detectar cada uno de ellos. Entre un evento *MoveStop* y un *MoveStart* tenemos un lugar visitado y si es al contrario obtenemos un desplazamiento completo. Los patrones que definen estos eventos se muestran a continuación, estos serán embebidos en un *String* Java y pasados al motor directamente.

```
1 insert into Move
2 select p1.lat as latitude_start, p1.long as longitude_start,
      p1.timestamp as start, p2.lat as latitude_end, p2.long
     as longitude_end, p2.timestamp as end
3 from pattern [every (p1 = MoveStart -> p2 = MoveStop)];
```

```
insert into Place
select p1.latitude as latitude, p1.longitude as longitude,
    p1.timestamp as start, p2.timestamp as end
from pattern [every (p1 = MoveStop -> p2 = MoveStart)];
```

A continuación, pasamos a obtener información sobre la ruta en sí. Una situación de interés son los cambios en la velocidad. De esta forma se puede detectar con un alto nivel de confianza cuando el usuario va andando, en bicicleta, o en un vehículo a motor. Así, analizamos todos los *Point* de forma que les asignamos el margen de velocidad en el que se encuentran. Consideramos que un *Point* lleva una velocidad lenta si está por debajo de los 3 m/s, una velocidad media si está entre 3m/s y 11m/s, y una velocidad rápida por encima de los 11m/s. Cuando en una ruta hay un cambio de un margen a otro, marcamos ese punto ya que puede coincidir con el aparcamiento de un coche, una parada de autobús, u otra situación interesante sobre la ruta que sigue el usuario.

Otra característica de interés a lo largo de una ruta son los cambios de dirección. De esta forma podemos caracterizar la ruta completa del usuario sin necesidad de conservar todas las medidas capturadas. Para calcular esta dirección hacemos una transformada aproximada, basándonos en el radio de la Tierra; así obtenemos que cada grado de latitud son aproximadamente 111.111 metros. Seguidamente, transformamos el vector de dirección obtenido en unitario. Una vez realizado el cálculo, lanzamos al flujo de eventos un nuevo evento complejo denominado *Direction* que contiene los atributos del primer *Point* analizado junto con una nueva propiedad que es el vector de dirección unitario. La realización del cálculo se realiza en la función *update* del subscriptor de este evento. Esta función es la que define que hacer cada vez que el patrón al que esté subscrito se cumpla. En este caso se invoca cada vez que agrupamos dos eventos *Point* y nos permite realizar estos cálculos de mayor complejidad.

```
public void update(Map<String, Point> eventMap) {
  Point a1 = (Point) eventMap.get("a1");
    Point a2 = (Point) eventMap.get("a2");
    float a, b, c;
    a = (a1.getLatitud() - a2.getLatitud()) * 111111;
    b = a1.getLongitud() - a2.getLongitud();
    b *= Math.cos(a2.getLatitud() * Math.PI/180) * 111111;
    c = (float) Math.sqrt(a * a + b * b);
    Vec2f result = new Vec2f(a/c, b/c);
    EPL.handle(new Direction(a1,result));
}
```

Finalmente, mediante un patrón que detecta los eventos *Direction* de dos en dos, invocamos el subscriptor del evento *DirectionChange*. En esta función calculamos la diferencia de ángulos entre ambas direcciones. Hemos simplificado los cálculos, ya que los vectores de ambas direcciones son unitarios. Si el ángulo de ambas direcciones es mayor de 36 grados ($\pi/5$), consideramos que ha habido un cambio de dirección en el itinerario y almacenamos las coordenadas GPS en el nuevo evento *DirectionChange*.

```
1 public void update(Map<String, Direction> eventMap) {
2     Direction a1 = (Direction) eventMap.get("a1");
3     Direction a2 = (Direction) eventMap.get("a2");
4     Float angulo = Math.acos(a1.getDir().x * a2.getDir().x +
          a1.getDir().y * a2.getDir().y);
5     if(Math.abs(angulo) > Math.PI/5)
6       EPL.handle(new DirectionChange(a1));
7 }
```

Llegados a este punto, es posible caracterizar los desplazamientos del usuario. Un esquema de los eventos que considerados y sus relaciones puede observarse en la Figura 1. Si necesitásemos obtener más información solo sería necesario

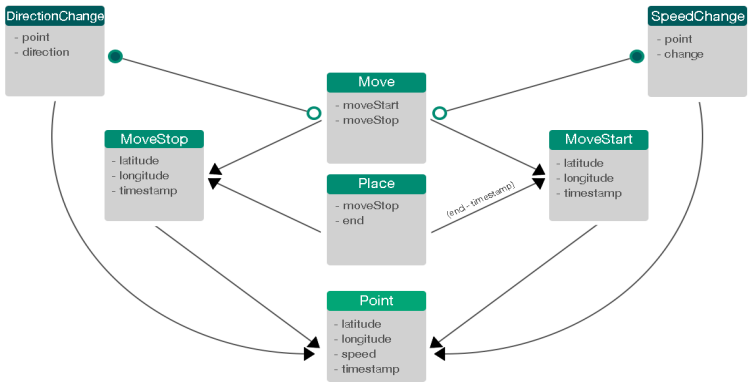

Figura 1: Esquema de los eventos detectados y sus relaciones.

incluir un nuevo patrón en el sistema para que el motor CEP lo tenga en cuenta. Esta es otra de las ventajas de esta tecnología, ya que ofrece una escalabilidad y flexibilidad muy altas para poder adaptar el sistema a nuevas demandas.

4. Resultados

Tras haber implementado los patrones explicados anteriormente hemos realizado diversas pruebas en distintos escenarios utilizando la aplicación para registrar los recorridos de un usuario. Hecho esto, analizaremos los eventos detectados por el motor CEP para validar la corrección del sistema e inspirarnos para nuevas ampliaciones y mejoras que podamos realizar sobre este.

Hemos establecido casos de prueba que recojan tipos de desplazamiento en rutas diferentes y utilizando medios de transporte distintos, desde andando, bicicleta o autobús. Empezaremos con el caso de prueba más sencillo, y en adelante se irán mostrando escenarios de pruebas cada vez más complejos.

Figura 2: Recorrido caminando.

El primer escenario que hemos escogido se trata de un desplazamiento andando. En este se muestra cómo el usuario se desplaza desde el punto A hacia el punto B. Como se aprecia en la Figura 2, todos los puntos son de color verde, indicativo de velocidad baja (ya que va andando). Al caminar a cielo abierto, no se produce ninguna pérdida de la señal de GPS y el motor identifica perfectamente la mayoría de los cambios de dirección, representados en azul.

Figura 3: Recorrido en bicicleta.

El segundo caso de prueba que hemos escogido se trata de un desplazamiento en bicicleta (Figura 3). Es una ruta que sale desde la Escuela Técnica Superior de Informática (punto A) y va al centro de la ciudad (punto B). En esta ocasión ya podemos observar todos los márgenes de velocidad que hemos establecido, verde para velocidades lentas, amarilla para velocidades medias, y roja para altas. Al ser un desplazamiento en bicicleta, la gran mayoría de los puntos están en amarillo salvo algunas cuestas hacia abajo (en rojo) y lo que puede ser alguna parada en semáforos, que se registra como color verde.

Por último, realizamos el mismo recorrido pero esta vez en autobús urbano. Así, se identifican tres tramos dentro de esta prueba; el primero caminando desde A hasta la parada de autobús en B, el segundo en autobús hasta el punto C, y por último un breve trayecto andando hasta el destino en el punto D. Como puede verse en la Figura 4, el primer tramo de desplazamiento aparece en verde, indicativo de que el usuario va andando. Si nos fijamos detenidamente, pueden observarse algunas pérdidas de la señal en A que se pueden deber al desplazamiento por el interior del edificio, también se inicia y acaba un desplazamiento

dentro del edificio (par de puntos blanco y negro) debido a este mismo fenómeno.

Figura 4: Recorrido en autobús con detención intermedia en la parada

El segundo tramo del desplazamiento transcurre con mayor normalidad. En esta ocasión, los puntos son rojos por la velocidad alta del autobús, pero al llegar a la parada del destino (C), los puntos se vuelven color verde indicando que el usuario se ha bajado del autobús y finaliza el desplazamiento andando. El cálculo utilizado para detectar cambios de dirección parece funcionar correctamente.

5. Conclusiones y Trabajos Futuros

En este trabajo hemos implementado una nueva versión del motor de inferencia para la arquitectura PeaaS usando la tecnología CEP para el análisis de información en tiempo real. Así, construimos un perfil virtual de los usuarios que sirva como interfaz para adaptar a sus necesidades los servicios de las ciudades inteligentes. Combinando esta información con datos IoT acerca de los medios de transporte público, podremos realizar recomendaciones al usuario, mejorar los planes de estos servicios y promover, por ejemplo, el uso de transporte alternativo al vehículo particular, reduciendo así las emisiones de CO2.

En otros casos de prueba realizados, tuvimos dificultades de detectar algunos medios de transporte, como el metro, ya que perdíamos la señal GPS. El próximo paso sería caracterizar este tipo de situaciones, así como la de la espera en una parada de autobús, apoyándonos en la información sobre transportes públicos disponible en el portal de datos abiertos del ayuntamiento de Málaga (http://datosabiertos.malaga.eu), lo que podría ayudarnos a determinar con mayor precisión el medio de transporte que utiliza el usuario en cada desplazamiento. Queda también para próximas iteraciones el conseguir un mejor filtrado de los datos de forma que podamos reducir la aparición de falsas rutas y pérdidas de señal indicando puntos fuera de la ruta. Todos los casos de prueba son fácilmente reproducibles y la aplicación se encuentra disponible en GitHub[1].

[1] Aplicación en GitHub: https://github.com/wifloso/AnalisisRutasTFG. Queremos agradecer a Carlos Salguero Tejada su contribución a este trabajo durante la realización de su TFG.

Referencias

1. Ayala-Gómez, F., Daróczy, B.Z., Mathioudakis, M., Benczúr, A., Gionis, A.: Where could we go? recommendations for groups in location-based social networks. In: Proceedings of the 2017 ACM on Web Science Conference. ACM Press (2017)
2. Berrocal, J., Boubeta-Puig, J., Canal, C., Garcia-Alonso, J., Murillo, J.M., Ortiz, G.: Construyendo perfiles virtuales mediante el procesamiento de eventos complejos. In: Jornadas de Ciencia e Ingeniería de Servicios JCIS, Sistedes 2016 (2016)
3. Birth, O., Hoffmann, M., Strassberger, M., Roor, R., Schlichter, J.: Concept for an intermodal traveller information system with real-time data using complex event processing. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems. pp. 2293–2298. IEEE (2015)
4. Cardone, G., Foschini, L., Bellavista, P., Corradi, A., Borcea, C., Talasila, M., Curtmola, R.: Fostering participaction in smart cities: a geo-social crowdsensing platform. IEEE Communications Magazine **51**(6), 112–119 (2013)
5. Cohen, B.: What exactly is a smart city. Co. Exist **19** (2012)
6. Dunkel, J., Bruns, R., Stipković, S.: Event-based smartphone sensor processing for ambient assisted living. In: 2013 IEEE Eleventh international symposium on autonomous decentralized systems (ISADS). pp. 1–6. IEEE (2013)
7. Eggum, M.: Asper - Esper for Android. https://github.com/mobile-event-processing/Asper, accessed: 2019-04-01
8. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: a mobile-centric model for providing collective sociological profiles. IEEE software **31**(2), 48–53 (2014)
9. Kondamudi, P.R., Protano, B., Alhoori, H.: Pokémon go: Impact on yelp restaurant reviews. In: Proceedings of the 2017 ACM on Web Science Conference. pp. 393–394. ACM (2017)
10. Luckham, D.C.: Event processing for business: organizing the real-time enterprise. John Wiley & Sons (2011)
11. Manville, C., Cochrane, G., Cave, J., Millard, J., Pederson, J.K., Thaarup, R.K., Liebe, A., Wissner, M., Massink, R., Kotterink, B.: Mapping smart cities in the eu (2014)
12. Nam, T., Pardo, T.A.: Conceptualizing smart city with dimensions of technology, people, and institutions. In: Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times. pp. 282–291. ACM (2011)
13. Pérez Lozano, P., Pérez Vereda, A., Murillo, J.M., Canal-Velasco, J.C., et al.: Safewalks: aplicación móvil de supervisión de pacientes de alzheimer. In: Jornadas de Ciencia e Ingeniería de Servicios JCIS, Sistedes 2015 (2015)
14. Sheth, A.: Computing for human experience: Semantics-empowered sensors, services, and social computing on the ubiquitous web. IEEE Internet Computing **14**(1), 88–91 (2010)
15. Wang, F.Y., Carley, K.M., Zeng, D., Mao, W.: Social computing: From social informatics to social intelligence. IEEE Intelligent systems **22**(2), 79–83 (2007)