JCIS 2013

IX Jornadas de Ciencia e Ingeniería de Servicios (JCIS)

Madrid, 18 y 19 de Septiembre de 2013

Editores:

Vicente Pelechano
José Albors
Marcos López Sanz

Actas de las "IX Jornadas de Ciencia e Ingeniería de Servicios (JCIS)"
Madrid, 18 y 19 de septiembre 2013
Editores: Vicente Pelechano, José Albors, Marcos López Sanz

ISBN: 978-84-695-8351-7

Prólogo

El presente volumen contiene los artículos seleccionados para presentación en las *IX Jornadas de Ciencias e Ingeniería de Servicios* (JCIS 2013) celebrado en Septiembre de 2013 en Madrid, España. Este año las Jornadas se celebran al amparo del IV Congreso Español de Informática (CEDI 2013).

El principal objetivo de las Jornadas es proporcionar un foro de discusión e intercambio de conocimiento y experiencias en el ámbito de la Ciencia de Servicios. El interés no sólo se centra en los nuevos avances científicos, sino también en las tecnologías existentes en torno a la computación orientada a servicios y los procesos de negocio, las nuevas prácticas de ingeniería de servicios y las lecciones aprendidas por medio de experiencias reales. Las JCIS, que celebran este año su octava edición, son el resultado de la integración de las Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB) y el Taller sobre Procesos de Negocio e Ingeniería de Servicios (PNIS). Seguir contando después de nueve años, y en el contexto de crisis en el que nos encontramos actualmente, con un foro de encuentro que nos permita intercambiar conocimiento y experiencias entre grupos de investigación de distintas Universidades españolas y profesionales de la Administración Pública y de la Industria, es sin duda un triunfo de nuestra comunidad que debemos y queremos destacar.

En esta edición se han recibido 18 contribuciones para su revisión, de las cuales 3 eran artículos ya publicados en congresos y revistas de reconocido prestigio. Todas las contribuciones fueron revisadas por, al menos, tres miembros del comité de programa. Como resultado de este proceso de revisión, se seleccionaron 15 trabajos para su presentación en las jornadas, 3 de los cuales corresponden a trabajos ya publicados. La presentación de los trabajos se ha organizado en cuatro sesiones temáticas distribuidas a lo largo de los dos días que durarán las Jornadas. Los trabajos se agrupado en los siguientes temas: "Procesos de Negocio", "Servicios Inteligentes", "Métodos y Servicios Intensivos de Conocimiento" y "Verificación y Semántica de Servicios".

Nos gustaría agradecer a todos aquellos que de un modo u otro han contribuido a la organización de estas Jornadas. En primer lugar, a todos los autores de los artículos enviados a JCIS 2013, y a los miembros del Comité de Programa por su disponibilidad y dedicación a la hora realizar las revisiones. Agradecer además a nuestros colaboradores: ATI, Novática, y la Red Científico-Tecnológica en Ciencias de los Servicios financiada por el Ministerio de Economía y Competitividad.

Finalmente, agradecemos a la Sociedad de Ingeniería del Software y Tecnologías de Desarrollo del Software (SISTEDES) y a la organización por parte de los miembros del CEDI 2013. Conocemos la dificultad de la organización de este tipo de eventos, y máxime en la situación de crisis económica en la que nos encontramos, por lo que destacamos y agradecemos enormemente vuestro esfuerzo y dedicación en la realización de estas Jornadas.

Gracias a todos y esperamos que disfrutéis de las Jornadas y de vuestra estancia en Madrid.

Madrid, Septiembre 2013

Vicente Pelechano y José Albors
Presidentes del Comité científico

Marcos López Sanz
Presidente del Comité organizador

Comité Científico

PRESIDENTES DEL COMITÉ DE PROGRAMA

Vicente Pelechano (Universidad Politécnica de Valencia)
José Albors (Universidad Politécnica de Valencia)

COORDINADORA DE DIVULGACIÓN DE ARTÍCULOS YA PUBLICADOS

Victoria Torres (Universidad Politécnica de Valencia)

MIEMBROS DEL COMITÉ CIENTÍFICO

Alfonso Ríos (Gnúbila)
Andrea Delgado (InCo)
Antonio Ruiz-Cortés (Universidad de Sevilla)
Antonio Vallecillo (Universidad de Málaga)
Carlos Bobed (Universidad de Zaragoza)
Daniel González Morales (Ag. Canaria de Inv., Innovación y Soc. De la Inf.)
Diego López (Telefónica I+D)
Enrique Beltrán (Software AG)
Félix García (Universidad de Castilla-La Mancha)
Fernando González Ladrón de Guevara (Universidad Politécnica de Valencia)
Francisco Almeida Rodríguez (Universidad de La Laguna)
Francisco Javier Fabra (Universidad de Zaragoza)
Francisco Ruiz (Universidad de Castilla-La Mancha)
Guadalupe Ortiz Bellot (Universidad de Extremadura)
Jaime Cid (Oracle)
Javier Troya (Universidad de Málaga)
Jesús Arias Fisteus (Universidad Carlos III de Madrid)
Jesús Gorroñogoitia (Atos Origin)
Jesús Bermejo (Schneider Electric)
Joan Pastor (Universitat Oberta de Catalunya)
Jordi Marco (Universidad Politécnica de Cataluña)
José Emilio Labra (Universidad de Oviedo)
José Hilario Canós (Universidad Politécnica de Valencia)
Jose Manuel Gómez (iSOCO)
José M. López Cobo (Playence KG)
José Raúl Romero (Universidad de Córdoba)
Juan De Lara (Universidad Autónoma de Madrid)
Juan Hernández (Universidad de Extremadura)
Juan José Moreno Navarro (Universidad Politécnica de Madrid)
Juan Manuel Murillo (Universidad Extremadura)
Juan Pavón (Universidad Complutense de Madrid)
Leire Bastida (Tecnalia)
Manuel Lama (Universidad de Santiago de Compostela)
Manuel Resinas (Universidad de Sevilla)
Marcos López Sanz (Universidad Rey Juan Carlos)
María del Carmen Penadés (Universidad Politécnica de Valencia)
María Valeria De Castro (Universidad Rey Juan Carlos)
María-Ribera Sancho (Universidad Politécnica de Cataluña)
Marta Patiño (Universidad Politécnica de Madrid)
Martín Álvarez Espinar (W3C Spain)
Mercedes Ruiz (Universidad de Cádiz)
Óscar Corcho (Universidad Politécnica de Madrid)
Pedro Alvarez (Universidad de Zaragoza)
Pere Botella (Universidad Politécnica de Cataluña)
Rafael Corchuelo (Universidad de Sevilla)
Silvia Acuña (Universidad Autónoma de Madrid)
Víctor Ayllón (Novayre)

Comité de Organización

PRESIDENTE DEL COMITÉ ORGANIZADOR
Marcos López Sanz (Univ. Rey Juan Carlos)

COORDINADOR DE LA WEB Y DE PUBLICIDAD
Diana M. Sánchez Fúquene (Univ. Rey Juan Carlos)

MIEMBROS DEL COMITÉ ORGANIZADOR
Verónica Bollati (Univ. Rey Juan Carlos)
Valeria de Castro (Univ. Rey Juan Carlos)
Javier Garzás (Univ. Rey Juan Carlos)
David Granada (Univ. Rey Juan Carlos)
Esperanza Marcos (Univ. Rey Juan Carlos)
Ángel Moreno (Univ. Rey Juan Carlos)
Iván Santiago (Univ. Rey Juan Carlos)
Feliu Trias (Univ. Rey Juan Carlos)
Juan Manuel Vara (Univ. Rey Juan Carlos)
Jenifer Verde (Univ. Rey Juan Carlos)
Manuel Zahera (Fundacion Cotec)

Entidades colaboradoras

Red temática en Ciencias de los Serivicios	GOBIERNO DE ESPAÑA · MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD
Asociación de Técnicos de Informática	Asociación de Técnicos de Informática
Revista de la Asociación ATI	novática — Revista de la Asociación de Técnicos de Informática

Índice de Contenidos

Integración del Internet de las Cosas y las Arquitecturas Orientadas a Servicios: un Caso de Estudio en el Ámbito de la Domótica

Juan Boubeta-Puig, Guadalupe Ortiz e Inmaculada Medina-Bulo

Departamento de Ingeniería Informática, Universidad de Cádiz,
C/Chile 1, 11002 Cádiz, España
{juan.boubeta,guadalupe.ortiz,inmaculada.medina}@uca.es

Resumen El Internet de las Cosas (IoT) describe un escenario en el que los objetos se encuentran unívocamente identificados y conectados a Internet; en este escenario se favorece el control remoto de situaciones críticas o relevantes para un dominio, a través de sensores y actuadores distribuidos geográficamente a nivel mundial. Sin embargo, para poder detectar dichas situaciones es necesario analizar y procesar eficientemente la gran cantidad de información manipulada cada día por estos dispositivos. En este artículo proponemos la integración del IoT y las arquitecturas orientadas a servicios junto con el procesamiento de eventos complejos para facilitar la detección de dichas situaciones en tiempo real en entornos heterogenéos. Este enfoque se ilustra mediante un caso de estudio que detecta y alerta automáticamente de situaciones no deseadas o de riesgo en nuestros hogares, por ejemplo un alto consumo eléctrico, permitiendo así -en este caso- reducir el consumo energético y aumentar la seguridad.

Keywords: Arquitecturas Orientadas a Servicios, Internet de las Cosas, Procesamiento de Eventos Complejos, Arquitecturas Dirigidas por Eventos, Bus de Servicios Empresarial.

1. Introducción

El Internet de las cosas o *Internet of Things* (IoT) es un paradigma emergente que propone el uso de una red de "cosas" u objetos, como sensores y actuadores, interconectados a nivel mundial e identificados unívocamente a través de un esquema de direcciones. De esta forma, cada objeto puede interactuar y cooperar con los demás objetos para alcanzar un objetivo común [3].

Gracias a la interconexión de todos estos objetos a través de Internet, se facilita la monitorización remota de toda la información proporcionada por dichos dispositivos y compartida a través de plataformas IoT. Ante esta situación, sería deseable disponer de sistemas que sean capaces de analizar dicha información de forma remota para detectar automáticamente y en tiempo real situaciones críticas o relevantes para un dominio concreto; por ejemplo, la domótica. En este contexto, se podrían detectar situaciones de interés para el propietario de la

casa, como por ejemplo, un derroche energético al tener encendida la estufa de su casa cuando la temperatura exterior es agradable, y situaciones más peligrosas como la detección de un posible caso de incendio cuando la temperatura de la casa alcance valores extremos.

Para poder detectar estas situaciones, en este artículo proponemos la integración del IoT y las arquitecturas orientadas a servicios dirigidas por eventos o *Event-Driven Service-Oriented Architecture* (ED-SOA o SOA 2.0) [13], junto con el procesamiento de eventos complejos o *Complex Event Processing* (CEP), aunando así las ventajas de cada uno de estos enfoques. Como hemos comentado previamente, el IoT nos proporciona información proveniente de objetos distribuidos geográficamente a nivel mundial. En segundo lugar, SOA es una solución eficiente para la implantación de sistemas en los que la modularidad y las comunicaciones entre terceros son un factor clave, desarrollándose así aplicaciones distribuidas compuestas de componentes (servicios) reutilizables y compartibles. Por otra parte, una arquitectura dirigida por eventos o *Event-Driven Architecture* (EDA) se caracteriza básicamente porque la conexión entre productores y consumidores de información (en forma de eventos) es indirecta y totalmente desacoplada, esto es, los productores no conocen (ni necesitan saber) cuáles serán los consumidores que recibirán los eventos generados por ellos. Finalmente, con vistas a analizar continuamente toda la información que fluye por las ED-SOA; éstas se integran con CEP [9], una tecnología emergente que permite procesar y analizar grandes cantidades de eventos, así como correlacionarlos para detectar y responder en tiempo real y de forma automática a las situaciones que son críticas o relevantes para los procesos de negocio. Para ello, se utilizan unos patrones de eventos que inferirán nuevos eventos más complejos y con un mayor significado semántico, que ayudarán a tomar decisiones para paliar cuanto antes las situaciones acontecidas.

Así pues, combinando IoT, SOA 2.0 y CEP podremos detectar eventos relevantes en sistemas complejos y heterogéneos. Esto requerirá un bus de servicios empresarial o *Enterprise Service Bus* (ESB), que actuará como capa de integración para la transformación, el enriquecimiento y el encaminamiento de mensajes entre servicios de diferentes aplicaciones, entre otros.

En un trabajo previo [4] ya propusimos detalladamente una arquitectura que combina SOA 2.0 y CEP. En este artículo, implementamos dicha arquitectura utilizando como productores de eventos una plataforma IoT, y como consumidores de eventos un servidor de correos y una base de datos NoSQL. Para llevar a cabo la integración SOA 2.0 y CEP hemos desarrollado un módulo que hace posible el envío de información (en forma de eventos) entre el ESB Mule [11] y el motor CEP Esper [7]. Una de las ventajas de esta arquitectura es que podrá analizar continuamente datos "reales" y heterogéneos generados por distintos sensores ubicados a lo largo de todo el planeta y conectados a plataformas IoT.

Además, se define e implementa un caso de estudio en la domótica para ilustrar la utilidad de nuestra arquitectura. En concreto este caso de estudio consiste en la detección temprana de situaciones relevantes o críticas en hogares distribuidos geográficamente a nivel mundial e interconectados a través de Internet. Para

detectar dichas situaciones, en este artículo también se proponen e implementan unos patrones de eventos complejos en el ámbito de la domótica.

El resto del artículo se estructura de la siguiente forma. En la Sección 2 se definen IoT y CEP. En la Sección 3 se describe la arquitectura para la integración de SOA 2.0 y CEP con IoT. Para ilustrar dicha arquitectura, en la Sección 4 se define e implementa un caso de estudio para la detección de situaciones críticas o relevantes en el ámbito domótico y en la Sección 5 se presentan los resultados obtenidos. A continuación, en la Sección 6 se enumeran algunos trabajos relacionados y, finalmente, se presentan las conclusiones y el trabajo futuro.

2. Conceptos Previos

2.1. Internet de las Cosas

IoT se define como una red formada por objetos interconectados que pueden ser identificados unívocamente y representados en un mundo virtual. Para poder identificar estos objetos será necesario el uso de mecanismos, como identificación por radiofrecuencia. Algunos de los dominios donde puede aplicarse IoT son [3]: transporte y logística, salud y domótica, entre otros.

Una de las plataformas IoT potentes y escalables que existe en la actualidad es Xively [8], que gestiona cada día millones de datos provenientes de miles de personas, organizaciones y compañías a nivel mundial. Esta plataforma Web para construir aplicaciones IoT permite almacenar, compartir y extraer datos en tiempo real ofrecidos por los distintos objetos y dispositivos ubicados a lo largo de todo el planeta.

Xively ofrece la posibilidad de establecer conexiones tanto directas (entre dos dispositivos, objetos o entornos) como conexiones varios a varios. Además, dispone de una buena documentación y una API RESTful, que proporciona la información como flujos de datos en formato XML, CSV y JSON. Gracias a esta API, los distintos dispositivos, como sensores instalados en el hogar, podrán enviar sus datos a esta plataforma IoT, para que puedan ser compartidos y recibidos en tiempo real por las aplicaciones en cuestión.

2.2. Procesamiento de Eventos Complejos

CEP [9] es una tecnología que proporciona un conjunto de técnicas que ayudan a hacer un uso eficiente de las arquitecturas EDA. Se basa en el filtrado de eventos irrelevantes y en el reconocimiento de patrones de los eventos que sí son relevantes.

Un evento puede ser definido como algo que ocurre o que se espera que ocurra; y puede ser generado por distintas fuentes externas, tales como sensores de datos, servicios web, base de datos externas, entre otros. Téngase en cuenta que un evento puede ser clasificado como evento simple o evento complejo. Un evento complejo es una abstracción de otros eventos simples o complejos.

Además, CEP hace posible la detección de nuevos eventos más complejos, conocidos como "situaciones"; es decir, permite la captura y correlación de cualquier tipo de evento con el fin de detectar situaciones de una mayor complejidad semántica e inferir conocimiento valioso para los usuarios finales.

La característica principal de estos eventos complejos procesados mediante tecnología CEP es que pueden ser identificados e informados en tiempo real, reduciendo la latencia en la toma de decisiones; a diferencia del software tradicional de análisis de eventos, que no funciona en tiempo real.

Así pues, CEP es una tecnología fundamental para aplicaciones que deban responder a situaciones que cambien rápidamente y de forma asíncrona, gestionar excepciones o reaccionar rápidamente a situaciones inusuales; y que requieran adaptabilidad y bajo acoplamiento.

3. Arquitectura que Integra IoT, SOA 2.0 y CEP

Como se ha comentado previamente, en este artículo implementamos la arquitectura que ya propusimos en [4] para la integración de CEP y SOA 2.0, y, además, la integramos con una plataforma IoT. En esta sección describimos qué elementos de la arquitectura y sus interconexiones son necesarios para llevar a cabo dicha integración.

El elemento principal de esta arquitectura es el ESB, que permitirá combinar dichos enfoques. Como puede verse en la Figura 1, los productores de eventos son sensores de datos, esto es, dispositivos que monitorizan el entorno para captar información (temperatura, humedad, consumo energético, etc.), que posteriormente es transmitida a las plataformas IoT por los controladores que se encuentran integrados en estos sensores. Por tanto, esta información quedará disponible a todas aquellas aplicaciones, entornos y usuarios que estén autorizados para acceder a dichas plataformas.

El ESB será el que se encargue de recibir la información de la plataforma IoT utilizando, para ello, el *endpoint* HTTP. Cabe destacar que se utiliza este *endpoint* HTTP, en lugar del adaptador TCP/IP propuesto en nuestro artículo previo, ya que en este caso los datos de los sensores son proporcionados por los distintos servicios Web de las plataformas IoT.

Una vez se reciban los datos de los sensores, el ESB se encargará de normalizarlos y transformarlos al formato de eventos adecuado para, posteriormente, enviarlos al motor CEP. Para llevar a cabo este envío, en este trabajo se ha implementado el módulo CEP, esto es, el módulo que hace posible que se puedan enviar datos desde el ESB al motor CEP, y viceversa. Cabe destacar que aunque en principio Mule proporciona un módulo para integrarse con Esper [12], dicho módulo está desactualizado y no funciona en ninguna de las versiones actuales. Es por ello que hemos desarrollado nuestro propio módulo.

Este motor CEP es el software que se encargará de recibir los datos de los sensores en formato de eventos (denominados eventos simples), analizarlos y correlacionarlos para detectar las situaciones críticas o relevantes que hayan sido previamente definidas en el motor. Para definirlas, se utilizan los llamados

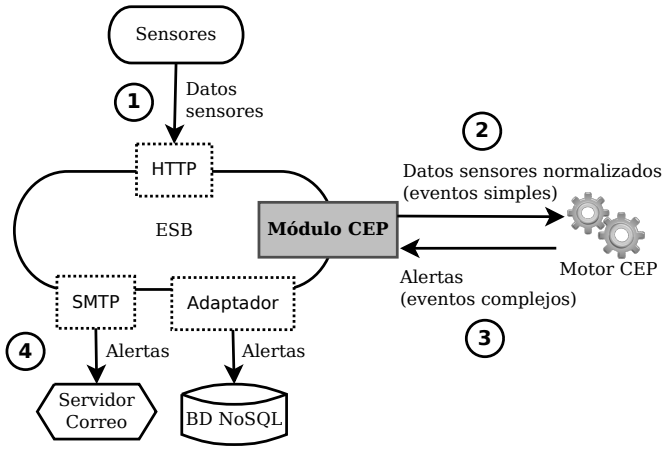

Figura 1. Arquitectura para la integración de IoT, SOA 2.0 y CEP.

patrones de eventos, esto es, las "plantillas" donde se especifican las condiciones que deben cumplirse para que sean detectadas dichas situaciones.

Cada vez que se detecte un patrón de eventos, se creará un evento complejo que describirá la situación acontecida. Entonces, el motor CEP enviará automáticamente este evento complejo o "alarma" al ESB.

Una vez que las "alarmas" hayan sido recibidas por el ESB, el último paso consistirá en enviarlas a todos aquellos sistemas, entornos y usuarios interesados en recibirlas. En esta ocasión, se ha optado por enviar las alarmas por correo electrónico usando el protocolo SMTP, así como almacenarlas en bases de datos NoSQL (Not only SQL) [2], un tipo de base de datos emergente basada en relaciones clave/valor, fácilmente escalables horizontalmente y eficientes para la manipulación de grandes cantidades de datos. Así pues, por un lado, los usuarios interesados en tipos de alarmas concretas las recibirán por correo electrónico y, por otro lado, serán almacenadas para disponer de un histórico de los nuevos eventos complejos creados en cada momento.

4. Caso de Estudio

En los últimos años está creciendo el número de casas inteligentes que disponen de sensores de datos para monitorizar las condiciones del entorno en el que se encuentran. Estos sensores pueden medir con cierta frecuencia la temperatura y humedad externa o interna de un hogar, o el consumo energético o de gas, entre otros. Además, existe una tendencia de compartir esta información recogida por dichos sensores en plataformas IoT de acceso libre.

Este escenario cobra interés a la hora de monitorizar estos datos en tiempo real para detectar situaciones relevantes o críticas que sucedan en nuestros hogares; tratanto de paliar cuanto antes sus consecuencias. Por ejemplo, sería

deseable que nos avisasen cuando realizamos un consumo irresponsable de electricidad, ya que de esta forma podríamos poner más cuidado en cómo usamos nuestros electrodomésticos.

Por ello, en este caso de estudio se ha aplicado la arquitectura que integra IoT, SOA 2.0 y CEP, introducida en la Sección 3, al campo de la domótica.

4.1. Descripción de las Fuentes de Datos

Como se ha comentado previamente, utilizaremos datos sobre domótica proporcionados por Xively. En este trabajo se ha realizado un estudio de cuáles de los flujos de datos disponibles en dicha plataforma sobre domótica son más idóneos para la monitorización, teniendo en cuenta el número de sensores disponibles por hogar, así como la frecuencia de actualización de los datos. En la Tabla 1 se muestran los nombres de los flujos, el país donde se encuentran los sensores asociados a cada flujo, las URL de los flujos utilizados, así como la frecuencia de actualización de datos de cada uno de ellos.

Nº	Nombre del flujo	País	URL	Actualización
F1	*Residential information*	Holanda	`https://xively.com/feeds/62988`	1 min
F2	*HAC Center*	Polonia	`https://xively.com/feeds/103216`	1 min
F3	*Current Cost Bridge*	España	`https://xively.com/feeds/89125`	5 min
F4	*Hirsch House Data*	EE.UU.	`https://xively.com/feeds/24319`	5 min

Tabla 1. Flujos de Xively sobre domótica utilizados en este caso de estudio.

Debe tenerse en cuenta que en cada uno de estos flujos se depositan los valores tomados de los distintos sensores que se encuentren en el hogar asociado a ese flujo. Por tanto, el nombre y el número de tipos de datos proporcionados por cada flujo es variable e incluso algunos de ellos se presentan en distintas unidades. Por ejemplo, la temperatura en el flujo *Hirsch House Data* está en escala Fahrenheit, mientras que en el resto en escala Celsius; y el consumo energético se toma en KW (kilovatios) en lugar de W.

Por este motivo, ha sido necesario normalizar los datos de estos cuatro flujos (F1 - F4), como se muestra en la Tabla 2. En esta tabla se especifica el nombre que le hemos asignado a cada dato junto con su tipo, una descripción y si se encuentra disponible en el flujo (marcado con una "x" en caso afirmativo). Estos datos serán almacenados en objetos de una clase denominada `EventoHogar`.

4.2. Patrones de Eventos Aplicados a la Domótica

A continuación, describimos la definición de los patrones de eventos complejos que permiten detectar situaciones críticas o relevantes en los hogares. Estos patrones se han implementado en el lenguaje EPL (*Event Processing Language*) de Esper por varios motivos. En primer lugar, la curva de aprendizaje no es elevada,

Dato	Tipo	Descripción	F1	F2	F3	F4
Hogar	*String*	Nombre del flujo.	x	x	x	x
Sensor	*String*	URL del flujo.	x	x	x	x
Localización	*String*	Nombre de la ciudad, país.	x	x	x	x
Latitud	*float*	Latitud de la localización.	x	x	x	x
Longitud	*float*	Longitud de la localización.	x	x	x	x
TiempoRegistro	*String*	Fecha y hora de registro del dato.	x	x	x	x
ConsumoEnergético	*float*	Consumo energético (W)	x	x	x	x
TemperaturaInterior	*float*	Temperatura interior del hogar (^{o}C)	x	x	x	x
TemperaturaExterior	*float*	Temperatura exterior (^{o}C)		x		x
HumedadInterior	*float*	Humedad interior del hogar (%)	x	x		
HumedadExterior	*float*	Humedad exterior (%)		x		x
ConexionTV	*boolean*	Estado de TV (encendido/apagado)			x	x

Tabla 2. Formato de los datos de flujos normalizados.

ya que su sintaxis se aproxima bastante a la del lenguaje SQL ampliamente conocido a nivel mundial. Por otro lado, EPL soporta de forma nativa varios tipos de formato de eventos: objetos Java/.NET, *maps* y documentos XML. Además, permite la personalización del lenguaje, así como del motor Esper, escrito en Java y de código abierto.

Consumo Energético Irresponsable Este patrón permite detectar situaciones en las que dentro de un hogar se está realizando un alto consumo energético (mayor a 1500 vatios) en un intervalo de tiempo determinado (30 minutos). La implementación de este patrón en EPL es la siguiente:

```
@Name("ConsumoEnergeticoIrresponsable")
insert into ConsumoEnergeticoIrresponsable
select e.tiempoRegistro as tiempoRegistro, e.hogar as hogar,
  e.localizacion as localizacion, e.latitud as latitud,
  e.longitud as longitud,
  e.consumoEnergetico as consumoEnergetico
from pattern [every e = EventoHogar(consumoEnergetico > 1500)
  ].win:time_batch(30 min)
```

En primer lugar, se define el patrón de eventos complejos mediante la cláusula `from pattern`. Se toma de entre todos los eventos de tipo `EventoHogar` aquellos que cumplan la condición en la que el consumo energético en ese instante sea mayor a 1500 W y, a continuación, se le aplica el operador `every` para seleccionar cada uno de estos eventos que representan un alto consumo energético. Para poder seleccionar a posteriori los atributos de este evento, es necesario asignarle un alias (en este caso denominado `e`).

Como puede observarse, a este patrón se le ha aplicado además una ventana temporal `win:time_batch(30 min)`, esto quiere decir que aunque se detecte varias veces el patrón de consumo energético irresponsable durante el intervalo

de tiempo de 30 minutos, sólo se notificará que el patrón ha sido detectado al finalizar los 30 minutos establecidos.

Finalmente, para cada uno de los eventos que cumple la condición impuesta en el patrón, se seleccionan los atributos siguientes: el tiempo de registro, el hogar donde se ha detectado la situación, la localización junto con la latitud y longitud, y el consumo energético. Con estos atributos se crea un nuevo evento complejo de tipo `ConsumoEnergeticoIrresponsable(tiempoRegistro, hogar, localizacion, latitud, longitud, consumoEnergetico)` y se inserta en un nuevo flujo de eventos de Esper con la misma denominación que el evento complejo.

Uso Estufa Irresponsable Este patrón trata de detectar situaciones en las que en un hogar se esté utilizando la calefacción cuando realmente no es necesario y, sin embargo, esté provocando un alto consumo energético. En este patrón, consideramos que no hace falta utilizar la estufa cuando la temperatura exterior sea menor de 19ºC y la temperatura interior sea mayor que 21ºC. Estas condiciones ponen de manifiesto que se tiene la calefacción encendida, ya que la temperatura interior es mayor que la exterior, pero a una temperatura más alta de la que realmente se aconseja en pro de un consumo responsable. La implementación de este patrón en EPL es la siguiente:

```
@Name("UsoEstufaIrresponsable")
insert into UsoEstufaIrresponsable
select e.tiempoRegistro as tiempoRegistro,
  e.hogar as hogar, e.localizacion as localizacion,
  e.latitud as latitud, e.longitud as longitud,
  e.temperaturaExt as temperaturaExt,
  e.temperaturaInt as temperaturaInt
from pattern [every e = EventoHogar(temperaturaExt < 19
  and temperaturaInt > 21)].win:time_batch(30 min)
```

Incendio Este patrón detecta posibles casos de incendio. Para ello, se comprueba que si una vez analizada la temperatura de un hogar en el minuto siguiente se observa que la temperatura ha aumentado en más de 20ºC, entonces se puede deducir que se ha producido un incendio en la casa. Lógicamente, es improbable que en tan solo un minuto la temperatura pueda oscilar en más de 20ºC ni siquiera utilizando una estufa para calentar el ambiente; salvo que se produzca algún incendio. La implementación de este patrón en EPL es la siguiente:

```
@Name("Incendio")
insert into Incendio
select e2.tiempoRegistro as tiempoRegistro,
  e2.hogar as hogar, e2.localizacion as localizacion,
  e2.latitud as latitud, e.longitud as longitud,
  e1.temperaturaInt as temperaturaIntInicial,
```

```
  e2.temperaturaInt as temperaturaIntFinal
from pattern [every (e1 = EventoHogar -> (timer:interval(1 min)
  and e2 = EventoHogar(hogar = e1.hogar and (temperaturaInt -
  e1.temperaturaInt) > 20)))]
```

Corte Eléctrico Este patrón avisará cuando se produzca un corte del suministro eléctrico. Para ello, tendrá que cumplirse que en un momento determinado el consumo energético sea mayor que 0 vatios y, seguidamente, el consumo energético en ese mismo hogar sea igual a 0. Téngase en cuenta que suponemos que el consumo energético "normal" en un hogar nunca será 0, ya que siempre existirá algún electrodoméstico enchufado a la toma de corriente, como puede ser el caso del frigorífico. La implementación de este patrón en EPL es la siguiente:

```
@Name("CorteElectrico")
insert into CorteElectrico
select e2.tiempoRegistro as tiempoRegistro, e2.hogar as hogar,
  e2.localizacion as localizacion, e2.latitud as latitud,
  e.longitud as longitud,
  e1.consumoEnergetico as consumoEnergeticoInicial,
  e2.consumoEnergetico as consumoEnergeticoFinal
from pattern [every (e1 = EventoHogar(consumoEnergetico > 0)
  -> e2 = EventoHogar(hogar = e1.hogar and consumoEnergetico = 0))]
```

OlvidoApagarTV Este patrón trata de detectar situaciones en las que olvidamos apagar nuestros televisores cuando, por ejemplo, no estamos en casa. Para ello, se establece como condición a detectar si la televisión permanece encendida ininterrumpidamente durante 6 horas. En esta ocasión, suponemos que los usuarios no pasan más de 6 horas seguidas delante del televisor. En caso contrario, habría que aumentar el número de horas impuestas en este patrón. La implementación de este patrón en EPL es la siguiente:

```
@Name("OlvidoApagarTV")
insert into OlvidoApagarTV
select e1.tiempoRegistro as tiempoRegistroInicial,
  e2.tiempoRegistro as tiempoRegistroFinal,
  e2.hogar as hogar, e2.localizacion as localizacion,
  e2.latitud as latitud, e.longitud as longitud,
  e1.conexionTV as conexionTVInicial,
  e2.conexionTV as conexionTVFinal
from pattern [every (e1 = EventoHogar(conexionTV = true)
  -> (timer:interval(6 hours) and not e2 = EventoHogar(
  hogar = e1.hogar and conexionTV = false)))]
```

4.3. Integrando IoT, SOA 2.0 y CEP

En esta subsección vamos a explicar cómo se ha implementado la arquitectura que integra IoT, SOA 2.0 y CEP, utilizando la plataforma Xively como productor

de eventos, Esper como motor de procesamiento de eventos complejos, el ESB Mule, y como consumidores de eventos Gmail y la base de datos MongoDB.

Para ello, se han creado 2 flujos en Mule (véase la Figura 2). El primero de ellos, denominado *Domótica*, se encarga de todo el proceso necesario desde la toma de datos de Xively hasta su envío al motor Esper. El segundo flujo, denominado *ConsumirEventosComplejos*, recibirá los eventos complejos ("alarmas") detectados por los patrones de eventos definidos en el motor Esper y los enviará a las partes interesadas. Nótese que la figura no muestra este segundo flujo al completo por problemas de espacio.

A continuación, describimos la implementación del flujo Domótica. En primer lugar, se ha utilizado el ámbito *composite source* que permite al flujo Mule obtener datos desde distintas fuentes externas, incorporando para cada uno de los cuatro flujos de Xively descritos anteriormente un *endpoint* HTTP. Este *endpoint* llevará asociado el nombre de usuario y contraseña de la cuenta creada en Xively así como un elemento global HTTP *polling* que se encargará de acceder a Xively para recoger datos con una frecuencia de 1 o 5 minutos dependiendo del flujo (véase Tabla 1). Seguidamente, la información que obtenemos en formato JSON es convertida a objetos Java; estos objetos serán recibidos por el transformador que hemos implementado para normalizar sus datos y convertirlos a formato de eventos (véase Tabla 2). Finalmente, estos eventos serán enviados al motor Esper, utilizando el módulo que hemos implementado para integrar Mule con Esper, denominado *Mule -> Esper*, permitiéndose así el envío de información en ambas direcciones. Además, en este módulo se implementarán los patrones de eventos a detectar (véase Sección 4.2).

Conforme se vayan detectando eventos complejos en el motor Esper, se irán enviando al componente *VM* del flujo *ConsumirEventosComplejos*. Seguidamente, el control de flujo *Choice* detectará cuál es el tipo de evento complejo que se ha recibido y, entonces, se almacenará en una colección específica de la base de datos para este tipo de eventos complejos y se enviará un aviso por correo electrónico sólo a los destinatarios interesados.

5. Resultados del Caso de Estudio

En esta sección se presentan los resultados obtenidos al ejecutar el sistema implementado. Las pruebas se han realizado desde las 2:00 h. (UTC) hasta las 17:00 h. (UTC) del 18 de abril de 2013; en total una ejecución de 15 horas.

En primer lugar, la Tabla 3 detalla el número de eventos complejos ("alarmas") que ha sido detectado en cada uno de los hogares. Como puede comprobarse, sólo se han detectado alarmas de tipo *ConsumoEnergeticoIrresponsable* y *UsoEstufaIrresponsable* en los hogares *Residential information* y *HAC Center*, respectivamente. Por tanto, las alarmas que se han detectado hacen referencia a situaciones producidas por un uso irresponsable tanto del consumo energético en general como del uso de estufas. Así pues, sería necesario adoptar medidas de responsabilidad del consumo energético en dichos hogares.

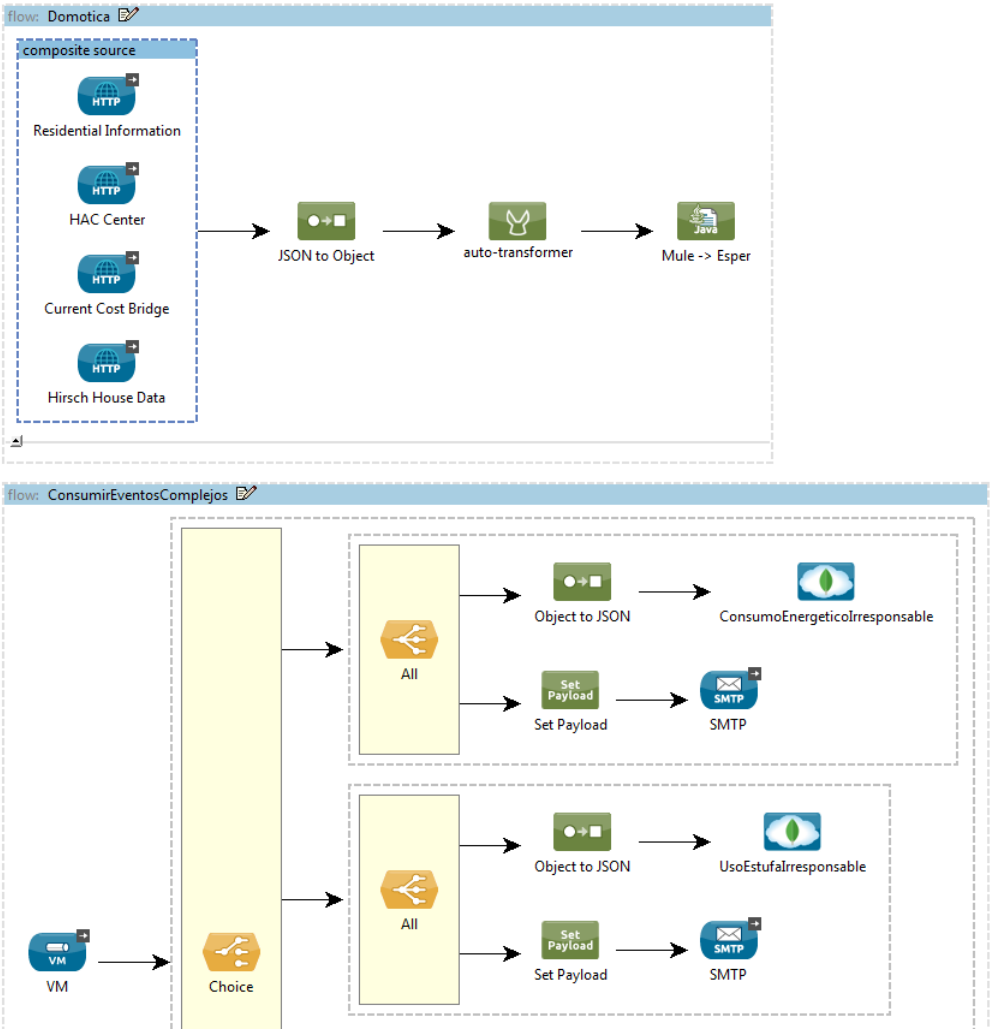

Figura 2. Implementación de la arquitectura que integra IoT, SOA 2.0 y CEP aplicada a la domótica.

En cuanto al número de eventos complejos detectados de tipo *Incendio*, *CorteElectrico* y *OlvidoApagarTV* cuyo valor es cero, habría que destacar que es un resultado "normal" y esperado, ya que la probabilidad de que ocurra, por ejemplo un incendio, es baja.

Nombre de Patrón de Eventos	Nº eventos detectados	Hogar
ConsumoEnergeticoIrresponsable	31	*Residential information*
UsoEstufaIrresponsable	11	*HAC Center*
Incendio	0	
CorteElectrico	0	
OlvidoApagarTV	0	

Tabla 3. Número de eventos complejos detectados y su localización.

En la Figura 3 se muestra el valor del consumo energético en los instantes en los que ha sido detectado el patrón *ConsumoEnergeticoIrresponsable*. Como puede observarse, conforme va finalizando el día el consumo energético en este hogar va decreciendo. Ante este escenario y tras recibir todas estas alertas, el propietario de la casa podría estudiar por qué existe un consumo tan elevado en las primeras horas del día, determinando qué medidas adoptar para reducirlo. De todas formas, aunque el consumo decrece en las últimas horas, sigue siendo un consumo elevado para una residencia en condiciones normales.

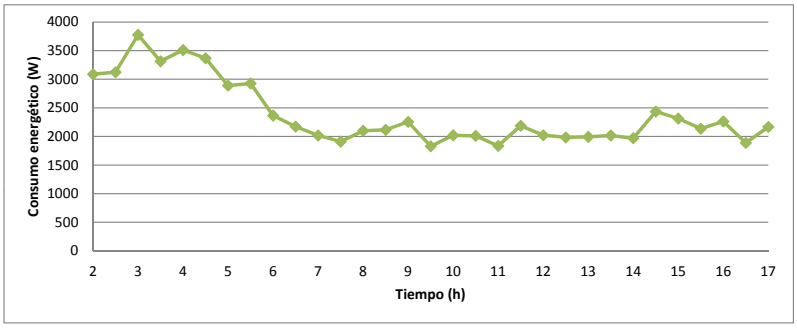

Figura 3. Medida del consumo energético en cada uno de los 31 eventos complejos *ConsumoEnergeticoIrresponsable* detectados.

Por otra parte, la Figura 4 especifica las temperaturas exterior e interior en cada uno de los 11 eventos complejos *UsoEstufaIrresponsable* detectados. Se puede apreciar que se está utilizando la estufa durante toda la noche hasta que seguramente los habitantes del hogar se despiertan a las 7.00 h, teniendo siempre la temperatura interior por encima de los valores recomendados para

un consumo responsable. Por tanto, se debería recomendar la regulación de la estufa para calentar el hogar a una temperatura alrededor de los 20 o 21 ºC.

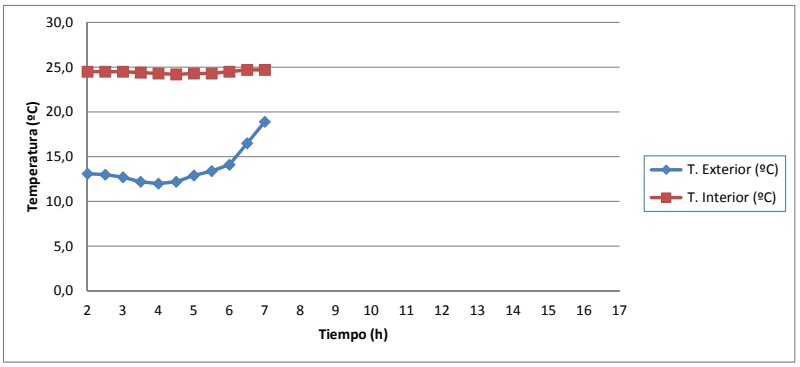

Figura 4. Medida de las temperaturas exterior e interior en cada uno de los 11 eventos complejos *UsoEstufaIrresponsable* detectados.

Tras haber realizado estos experimentos, podemos afirmar que nuestro sistema es capaz de detectar automáticamente situaciones críticas o relevantes que ocurran en hogares distribuidos geográficamente a nivel mundial. Además, nos encontramos ante un sistema escalable, ya que podrán incorporarse, si así se desea, nuevos flujos de datos provenientes de sensores de otros hogares.

6. Trabajos Relacionados

En los últimos años CEP se ha convertido en una de las tecnologías fundamentales para la detección automática y en tiempo real de situaciones relevantes o críticas en distintos contextos: fraudes, compra y venta de acciones, tráfico, logística y salud, entre otros.

Existen algunos trabajos que proponen la integración de CEP con IoT y SOA 2.0. Da et al. [6] proponen una arquitectura SOA que soporta servicios personalizados centrados en el usuario altamente escalables para IoT, haciendo uso de un ESB y el motor de reglas Drools. Por otro lado, Walczak et al. [14] describen un enfoque para comunicación máquina a máquina y procesamiento de datos en aplicaciones del Internet del futuro. En este caso, también utilizan el motor de reglas Drools. En nuestra arquitectura implementada utilizamos el motor CEP Esper, en lugar de un motor de reglas, ya que como afirman Chandy y Schulte [5] son más rápidos y eficientes. Según EsperTech, la empresa desarrolladora de Esper, este motor puede procesar en torno a 500.000 eventos por segundo en una estación de trabajo, y entre 70.000 y 200.000 eventos por segundo en un portátil.

Por otra parte, existen trabajos que integran IoT y SOA, aplicándolo al ámbito de la domótica, pero no usan CEP como tecnología de procesamiento de información. Por ejemplo, Miori y Russo [10] implementan ontologías específicas que automáticamente toman información de contexto y utilizan una representación del entorno basada en conocimiento. Ye y Huang [15] proponen un *framework* para casas inteligentes basado en *cloud* que proporcionan distintos servicios Web para detectar las situaciones relevantes que acontezcan dentro de los hogares. Nuestra arquitectura es más flexible que éstas, debido a que el motor CEP proporciona un lenguaje de patrones de eventos que facilita la incorporación de nuevos tipos de alarmas (eventos complejos) a detectar, en el caso de que sea necesario.

7. Conclusiones y Trabajo Futuro

En este trabajo se ha implementado la arquitectura propuesta en [4] que integra CEP y SOA 2.0, y, además, se ha integrado con plataformas IoT, utilizando un ESB como nexo de unión. Para llevar a cabo dicha integración hemos desarrollado un módulo que hace posible el envío de información (en forma de eventos) entre el ESB Mule y el motor CEP Esper.

Gracias a este enfoque, los datos "reales" (no simulados) emitidos por los distintos objetos y dispositivos interconectados a nivel mundial por plataformas IoT pueden ser analizados y correlacionados de forma remota y en tiempo real, haciendo posible la detección de situaciones que son críticas o relevantes para un dominio en concreto. Para que el motor CEP pueda llevar a cabo estas detecciones, es necesario definir e implementar previamente unos patrones de eventos en los que se indican las condiciones que deben cumplirse. Además, este motor alerta de estas situaciones al ESB, quien se encarga de notificárselas a las partes interesadas.

Para comprobar la utilidad de esta arquitectura, se ha descrito e implementado un caso de estudio en el que se ha aplicado dicha arquitectura al ámbito de la domótica. En concreto, se ha utilizado como productor de eventos la plataforma IoT Xively, Mule como bus de servicios empresarial, Esper como motor de procesamiento de eventos; y el servidor de correos Gmail y la base de datos NoSQL MongoDB como consumidores de eventos. Además, se han implementado unos patrones de eventos complejos para detectar posibles situaciones de interés en los hogares como son el consumo energético y el uso de estufas de manera irresponsable, así como casos de incendios, cortes eléctricos u olvidos de apagar el televisor cuando no estamos en casa.

Los resultados confirman que este sistema es capaz de detectar automáticamente situaciones críticas o relevantes que acontecen en distintos hogares situados a lo largo de todo el planeta. Aunque en este caso de estudio sólo se han tomado datos de sensores localizados en 4 hogares situados en Estados Unidos, Polonia, Holanda y España, esta arquitectura puede integrarse con otros sensores situados en hogares de otros países y continentes, debido a su alta escalabilidad.

En nuestro trabajo futuro más próximo extenderemos esta arquitectura integrándola con otras plataformas IoT como, por ejemplo, ThingSpeak [1]. Además, se usará en otros dominios de aplicación, lo que requerirá la definición de nuevos patrones de eventos complejos.

Agradecimientos. Este trabajo está financiado por el Ministerio de Ciencia e Innovación (proyecto TIN2011-27242). Los autores agradecen la colaboración de Rafael Bernárdez en la definición del caso de estudio. Además, los autores agradecen a Víctor Ayllón y Juan Manuel Reina, fundadores de la empresa Novayre, las discusiones y conocimientos intercambiados sobre la aplicación de CEP en entornos SOA.

Referencias

1. Internet of Things - ThingSpeak (2013), `https://www.thingspeak.com/`
2. NoSQL databases (2013), `http://nosql-database.org/`
3. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A Survey. Computer Networks 54(15), 2787–2805 (octubre 2010)
4. Boubeta Puig, J., Ortiz, G., Medina Bulo, I.: Procesamiento de Eventos Complejos en Entornos SOA: Caso de Estudio para la Detección Temprana de Epidemias. In: Actas de las VII Jornadas de Ciencia e Ingeniería de Servicios. pp. 63–76. Servizo de publicacións da Universidade da Coruña, A Coruña, Spain (septiembre 2011)
5. Chandy, K.M., Schulte, W.R.: Event Processing: Designing IT Systems for Agile Companies. Estados Unidos (2009)
6. Da, Z., Bo, C., Yang, Z., Junliang, C.: Future Service Provision: Towards a Flexible Hybrid Service Supporting Platform. In: Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific. pp. 226 –233 (diciembre 2010)
7. EsperTech Inc.: Esper - Complex Event Processing (2013), `http://esper.codehaus.org/`
8. LogMeIn: Xively - Public Cloud for the Internet of Things (2013), `https://xively.com/`
9. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Estados Unidos (2001)
10. Miori, V., Russo, D.: Anticipating Health Hazards through an Ontology-Based, IoT Domotic Environment. In: The 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS). pp. 745–750 (2012)
11. MuleSoft: Mule ESB (2013), `http://www.mulesoft.org/`
12. MuleSoft: Mule Esper Module (2013), `http://mulesoft.github.io/mule-module-esper/mule/esper.html`
13. Taylor, H., Yochem, A., Phillips, L., Martinez, F.: Event-Driven Architecture: How SOA Enables the Real-Time Enterprise. Addison-Wesley, Indiana, USA (2009)
14. Walczak, D., Wrzos, M., Radziuk, A., Lewandowski, B., Mazurek, C.: Machine-to-Machine Communication and Data Processing Approach in Future Internet applications. In: The 8th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP). pp. 1–5 (julio 2012)
15. Ye, X., Huang, J.: A Framework for Cloud-based Smart Home. In: The International Conference on Computer Science and Network Technology (ICCSNT). vol. 2, pp. 894–897 (2011)