Formalización de una arquitectura de computación móvil basada en Linda

Alejandro Pérez-Vereda Carlos Canal Ernesto Pimentel
apvereda@lcc.uma.es canal@lcc.uma.es ernesto@lcc.uma.es
Universidad de Málaga. España *

Gracias a los avances y el desarrollo que está experimentando el campo de la Internet de las Cosas (*Internet of Things*, IoT), cada día hay más dispositivos a los que se llama inteligentes. Estos dispositivos disponen de sensores y además están dotados de conexión ya sea a la red o mediante alguna otra tecnología. Sin embargo, estos dispositivos, a pesar de lo que indique su nombre, precisan de una configuración e interacción con ellos completamente manual y específica. Lo ideal sería aprovechar esos sensores e "inteligencia" para adaptar su comportamiento automáticamente a las necesidades de los usuarios. En este ámbito, proponemos el diseño de una arquitectura de computación móvil "*People as a Service*", que confiere a los dispositivos móviles (como los *smartphones*) la capacidad de inferir y compartir un perfil virtual del usuario. Más allá, hemos establecido un método de interacción programática para que el dispositivo móvil, ahora con la información e inteligencia necesarias, ejerza de interfaz con estos dispositivos de IoT configurándolos de forma automática y dinámica. En este trabajo proponemos una primera aproximación para formalizar estas interacciones dinámicas mediante un enfoque basado en espacios de tuplas distribuidos y compartidos, basado en Linda.

1. Introducción

El modelo de Internet of Things (IoT) esta basado en la disposición de diversos dispositivos y sensores en nuestro entorno de forma que todos ellos ofrezcan una interfaz tanto para conectarse entre ellos como para obtener la información que recopilan y poder configurarlos [4]. A raíz de este fenómeno nacen lo que llamamos objetos inteligentes, ya que permiten ser adaptados en función de la información disponible en el acto.

Sin embargo, estos objetos precisan de una interacción manual y especifica para cada uno de ellos. Esta falta de autonomía hace tedioso el día a día de los usuarios necesitando configurar cada uno de los objetos que tenga en su entorno de forma individual para que se ajusten a sus necesidades. En un escenario ideal, este proceso debería ser automático y transparente para el usuario, la tecnología debería trabajar para nosotros y no al contrario.

En esta linea, en [5] se propone una arquitectura de referencia de computación móvil llamada *People as a Service* (PeaaS). Esta arquitectura se sustenta sobre un modelo centrado en las personas cambiando su rol de tal manera que actúen como proveedores de servicio. La idea de PeaaS es aprovechar la gran presencia y capacidad computacional y sensorial de los smartphones para otorgarles la habilidad de inferir gestionar y compartir un perfil virtual del usuario con información sobre sus hábitos preferencias y contexto.

El siguiente paso en esta linea de trabajo es el de diseñar un nuevo paradigma de interacción de forma que esta información contenida en los smartphones sea considerada por los dispositivos IoT para que de forma inteligente se adapten de forma dinámica a cada usuario. Para ello en [6] los autores

*Este trabajo ha sido financiado por el Gobierno de España a través de los proyectos TIN2015-67083-R y PGC2018-094905-B-I00 (MINECO/FEDER).

Submitted to:
PROLE 2019

de este articulo presentamos un nuevo marco de programación dinámica que permite interacciones en tiempo real entre el móvil y los dispositivos mediante la ejecución de scripts en el acto. La idea esta basada en la visión que desarrollan en [8], donde se prevé una evolución desde el IoT actual basado en la obtención de información hacia un software flexible que permita la adaptación y configuración de los dispositivos hardware mediante la ejecución de código en tiempo real, convirtiéndolos en verdaderamente programables. De esta forma ambos smartphone y dispositivo podrán aprender y evolucionar en cada interacción de forma automática y transparente creando la idea de tecnología que funciona para las personas, teniendo en cuenta su contexto en cada momento sin necesidad de configuraciones manuales.

Una vez desarrolladas algunas pruebas de concepto con la idea planteada de forma general, observamos la necesidad de realizar una formalización sobre este marco de interacción programática que establezca las bases para conexiones seguras y acceso controlado al perfil virtual, garantizando la validez del sistema en todas las interacciones llevadas a cabo. En este trabajo introducimos esta formalización que estará basada en un modelo de múltiples espacios de tuplas compartidos, inspirado en Linda, destacando las características y consideraciones a tener en cuenta.

El trabajo se estructura introduciendo en la Sección 2 los conceptos básicos del marco de programación considerado y un ejemplo de escenario. En la Sección 2 se proponen los elementos que sustentan la formalización propuesta. Por último, se presentan las conclusiones en la Sección 4.

2. Fundamentos de la propuesta

A continuación se exponen las características fundamentales de esta propuesta, comenzando por una visión sobre el marco a formalizar que ayude a focalizar la linea de este trabajo emergente, y proporcionando un escenario que ilustre la motivación de nuestro enfoque.

2.1. Marco de computación móvil

De aquí en adelante, para simplicidad y compresión de la propuesta, consideraremos *beacons* como dispositivos IoT que representan de forma general la noción que pretendemos modelar sobre dispositivos ubicuos, contenedores de scripts e información capaces de establecer una conexión mediante *Bluetooth Low Energy* (BLE) con los teléfonos inteligentes. De esta forma, los beacons transmiten de forma continua una señal que contiene la dirección URL a un script, que el dispositivo móvil descargará y ejecutará en el acto en cuanto reciba una de estas señales. El script contendrá instrucciones para la actualización de la información existente en el perfil virtual y para la configuración remota y especifica del dispositivo en cuestión (el beacon en este caso). En un escenario real, los beacons serán cualquier dispositivo inteligente, que no necesite el apoyo de un servidor para almacenar su propio script y que ofrezca una interfaz de configuración.

El marco de trabajo, por lo tanto, provee las herramientas necesarias para, en primer lugar, la aportación y/o actualización dinámica y automática de los perfiles virtuales y, en segundo lugar, para permitir a los propios fabricantes hardware definir la inteligencia que tendrán sus dispositivos en función de la capacidad de adaptación que otorguen a estos mediante su propio script, pudiendo acceder a la información de cada usuario en el momento de la conexión sin necesidad de establecer contratos con las grandes multinacionales de información como Google.

La arquitectura del marco de trabajo se muestra en la Figura 1. En primer lugar, los beacons emiten una señal BLE con la URL al script (1). El smartphone que detecte esta señal, descarga el código del servidor (2), y lo ejecuta para actualizar el perfil virtual (3). Para terminar, se ejecutará la parte del script

que contiene operaciones para la propia actualización del dispositivo definidas por su fabricante a través de su interfaz especifica, de forma que en nuestro ejemplo con beacons, se subiría al servidor una nueva versión del script (4).

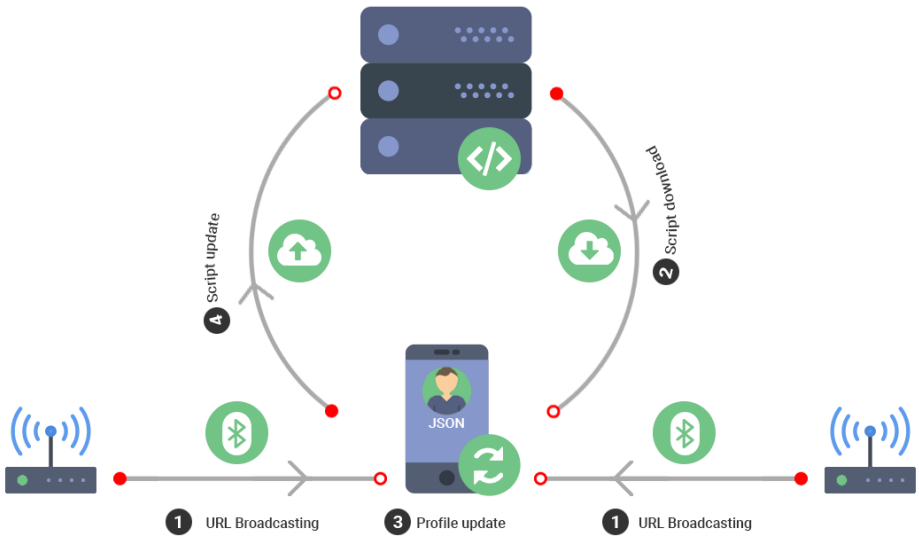

Figura 1: Framework de programación dinámica

2.2. Ejemplo de escenario

Para ayudar a entender el funcionamiento del sistema con el framework de programación dinámica proponemos un posible escenario de uso cotidiano el cual se vería afectado por el despliegue de esta tecnología.

Supongamos una situación habitual como una reunión en el trabajo. Normalmente esta tendrá lugar en una sala de reuniones equipada con un sistema de aire acondicionado inteligente. Si tenemos en cuenta esta tecnología y hacemos uso de nuestro marco, ya no habrá necesidad de que nadie configure la temperatura a la que está el aire acondicionado funcionando. Cada vez que un participante de la reunión entre en la sala, su smartphone descargará el script provisto por el fabricante del aire acondicionado a través de este. Este script se ejecutará en el smartphone de cada participante de modo que obtenga de su perfil virtual información sobre su temperatura deseada y posteriormente envíe al aire acondicionado esta información configurándolo para que funcione a gusto de todos los participantes según el protocolo que el fabricante del aparato considere apropiado para satisfacer las preferencias de todos. Así, el aire acondicionado funcionará tan inteligentemente como su fabricante estime oportuno, haciendo uso de información acerca de las personas involucradas en tiempo real y sin necesidad de establecer contratos con las grandes compañías que dispongan de esta información sobre los usuarios.

3. Hacia una formalización de PeaaS

Para la gestión de los perfiles virtuales partiremos de nuestra experiencia anterior con el modelo PeaaS ya mencionado. El perfil virtual de un smartphone puede ser accedido e incluso actualizado mediante procesos en el smartphone que ejecuten las operaciones adecuadas permitidas por la API de gestión de perfiles virtuales. Esta API es el único punto de acceso al perfil permitiendo controlar cada operación realizada sobre él. se trata de una API local, únicamente accesible desde el propio teléfono. Para formalizar esta idea, consideramos Linda [1], un lenguaje de coordinación [3] que consiste en un conjunto de primitivas de comunicación inter-agente que pueden ser añadidas virtualmente a un lenguaje de programación. Las primitivas de Linda permiten a los procesos leer, borrar y escribir tuplas en un espacio de tuplas compartido. Los espacios de tuplas serán nuestra representación de los perfiles virtuales compartidos entre los dispositivos involucrados en una interacción.

A diferencia de otras propuestas que modelan movilidad y computación distribuida, nuestro enfoque pone el foco en controlar que la información hospedada en los dispotivos móviles personales solo son accesibles según los permisos establecidos por la persona propietaria y que, además, los servicios a los que se puede acceder desde esos dispositivos pueden configurarse a través de la interacción con ellos. De este modo, por ejemplo, en [7] se propone una extensión de Linda, denominada LIME, que se basa en compartir de forma transitoria espacios de tuplas. Aunque la propuesta modela de forma adecuada la comunicación P2P, no se adecúa a uno de los requisitos fundamentales del paradigma PeaaS, que exige privacidad de las tuplas que representan perfiles virtuales. En otras propuestas, como KLAIM, descrita en [2], la extensión permite la aplicación de las primitivas de Linda sobre diferentes espacios de tuplas distribuidos. En este caso, lo relevante es la interacción independiente con distintos espacios de tuplas. Pero, de nuevo, no se contempla la posibilidad de tener parte de estos espacios de tuplas protegidos, ni expresar la modificación controlada de los servicios como consecuencia de la interacción los dispositivos.

En nuestra propuesta de extensión, modelamos el comportamiento interno de cada dispositivo con una extensión de las primitivas originales de Linda (mediante la incorporación de una acción para la adición remota de tuplas, pero no de forma indiscriminada) y la definición de una semántica operacional propia basada en reglas de transición que gobiernen la interacción basada en la descarga de scripts entre dispositivos.

A continuación definimos una noción de perfil virtual, que caracteriza la información que reside dentro de cada dispositivo.

Definition 1 *Un* perfil virtual *P es un multiconjunto de entidades, donde cada una es una 4-tupla $t = (n,s,p,v)$ compuesta por $n \in$ Name que representa el nombre de la entidad, $s \in$ Type que define su tipo, $p \in$ Privacy que proporciona el nivel de privacidad, y el valor $v \in$ Value, con una estructura que dependerá del tipo de entidad. Denotaremos por T el conjunto de tuplas, y por $\mathscr{P}$ el conjunto de perfiles virtuales.*

De este modo, un perfil virtual recoge información dentro de un dispositivo con una identificación, de un tipo determinado, con cierto nivel de privacidad y con un valor que puede ser modificado, pero siempre atendiendo a los privilegios de acceso delimitados por el nivel de privacidad que tenga asignado. La forma en que estos valores pueden ser modificados, o incluso las tuplas añadidas o eliminadas de un determinado perfil, estará condicionada por la ejecución de determinados scripts dentro del dispositivo. Precisamente, consideramos una extensión de lenguaje Linda para representar esos scripts. En efecto, el lenguaje $\mathscr{L}$ definido a continuación, proporcionará una forma adecuada de modelar scripts que pueden descargarse de un servidor y ejecutarse en un dispositivo inteligente. La sintaxis de $\mathscr{L}$ se puede definir

formalmente como sigue:

$$\begin{array}{rcl} S \in \mathscr{L} & ::= & 0 \mid \alpha.S \mid S+S \mid S \parallel S \mid S(\tilde{t}) \\ \alpha \in Act & ::= & rd(t) \mid nrd(t) \mid in(t) \mid out(d,t) \end{array}$$

donde 0 denota el proceso vacío, $d \in D$ un identificador de dispositivo, y t denota una tupla. El proceso $S(\tilde{t})$ denota una llamada a procedimiento donde la definición del mismo vendrá dada por $S(\tilde{x})$ (siendo $\tilde{x}$ una secuencia de variables instaciadas por la secuencia de tuplas $\tilde{t}$). Las acciones primitivas consideradas en $\mathscr{L}$ se corresponden con las habituales de Linda, de forma que $rd(t)$ permite progresar cuand la tupla t está presente en el espacio de tuplas y $nrd(t)$ cuando no hay ninguna tupla que encaje con t en el espacio de tuplas. La acción $in(t)$ es similar a $rd(t)$ pero, además, elimina la tupla t del espacio de tuplas. Por último, la operación $out(d,t)$ extiende la operación *out* de Linda, de una forma similar a como se hace en KLAIM, permitiendo la adición de tuplas en espacios distribuidos.

Para modelar los tipos de dispositivos involucrados en la arquitectura propuesta consideramos un conjunto de identificadores de dispositivo D. Formalmente, un dispositivo $d \in D$ estará representado por un par formado por un perfil virtual y un proceso, correspondiente a la ejecución de uno o varios scripts externos.

Definition 2 *Un* dispositivo $d \in D$ *se caracteriza por un par* $\langle P : S \rangle_d$*, que incluye un perfil virtual* P *y un script* $S \in \mathscr{L}$.

Cada dispositivo d dispondrá también de una definición de un script asociado (que denotaremos por $S_d(\tilde{x})$, el cual será descargable por cualquier otro dispositivo con capacidad de computación. Obsérvese que los scripts pueden ser instanciados (la secuencia $\tilde{x}$ puede tomar valores distintos) de forma distinta cuando son descargados por uno u otro dispositivo inteligente.

Establecemos una diferencia entre dispositivo y objeto inteligente. El primero ofrece capacidad computacional para la ejecución de scripts, mientras que el segundo solo proveerá los scripts. Comúnmente, un objeto inteligente será cualquier artefacto hardware de IoT que pueda interactuar con los smartphones, como los beacons utilizados para exponer la arquitectura del sistema. Ambos tipos de dispositivos (dispositivos y objetos inteligentes) poseen perfiles virtuales, sin embargo, entendemos que el proceso consiste en utilizar la información disponible en los perfiles virtuales de los dispositivos inteligentes para modificar el perfil virtual de los objetos inteligentes, a modo de actualización de su configuración.

En nuestra propuesta, las operaciones en ejecución en el smartphone pueden ser parte de un proceso interno del mismo o ser parte de un script que ha descargado de un objeto inteligente adyacente. Esto supone que tanto la descarga como la ejecución de las operaciones de los smartphones también deben estar formalizadas para asegurar la integridad, privacidad y seguridad en las interacciones y el propio perfil virtual. Por ello, además de disponer de la API de gestión de perfiles virtuales estableceremos una serie de requisitos y condiciones que propicien el buen funcionamiento del sistema como la necesidad de cercanía al objeto inteligente, que este aparezca registrado y proporcione un certificado firmado con alguna autoridad confiable o, por supuesto, que el código descargado cumpla con los requisitos de la API.

Con objeto de definir cómo interactúan los dispositivos consideraremos configuraciones compuestas por una composición paralela de dispositivos como:

$$\langle P_1 : S_1 \rangle_{d_1} \mid \langle P_2 : S_2 \rangle_{d_2} \mid \cdots \mid \langle P_n : S_n \rangle_{d_n}$$

donde los P_i ($i = 1..n$) son perfiles virtuales de dispositivos y objetos inteligentes, los S_i son scripts que se ejecutan en dispositivos inteligentes (en el caso de objetos inteligentes, entenderemos que $S_i = 0$,

$$\text{(SYNC)}\quad \frac{accept(P,e)}{\langle P:S\rangle_d \mid \langle Q:T\rangle_e \xrightarrow{\tau} \langle P:S \parallel S_e(Q)\rangle_d \mid \langle Q:T\rangle_e}$$

$$\text{(PAR)}\quad \frac{D_1 \xrightarrow{\alpha} D_1'}{D_1 \mid D_2 \xrightarrow{\alpha} D_1' \mid D_2}$$

Cuadro 1: Transition rules

y donde los d_i representan los identificadores de los dispositivos. Hemos de tener en cuenta que cada script S_i también será una composición paralela de procesos en $\mathscr{L}$, por lo que consideramos dos niveles de paralelismo en la propuesta, uno intra-dispositivo y otro inter-dispositivo. La semántica operaciones debe tener en cuenta esta distinción.

Por un lado, hemos de definir un sistema de transiciones etiquetado que rija la semántica operacional del comportamiento dentro de cada dispositivo:

$$\dot{\longrightarrow} \subseteq \mathscr{D} \times \Lambda \times \mathscr{D}$$

siendo $\mathscr{D} = \mathscr{P} \times \mathscr{L} \times \mathscr{D}$ y $\Lambda = \{t, \bar{t}, \underline{t} : t \in T\} \cup \{\tau\}$. La definición de $\dot{\longrightarrow}$ se puede obtener como extensión de Linda.

Un segundo nivel (el que da cuenta de la interacción entre dispositivos) será a su vez una extensión de $\dot{\longrightarrow}$, incluyendo al menos una regla que modele la descarga de scripts de un dispositivo a otro, y otra que modele la interacción concurrente entre ellos. El Cuadro 1 muestra estas dos reglas de transición.

La regla SYNC muestra la forma en que un dispositivo d descarga el script contenido en el dispositivo e, y que viene dado por el proceso Linda $S_c(\tilde{x})$, que se instancia de forma conveniente con información extraída del perfil Q del dispositivo de origen (es decir, $S_c(Q)$. El proceso de instanciación solo depende del estado del servidor que contenga el script, que puede haber sido modificado por otros dispositivos que hayan interactuado previamente con ese servidor, pero no depende del dispositivo que lo está descargando en cada momento. Es decir, no se necesita ningún gestor de scripts que realice modificaciones de los mismos en el dispositivo receptor.

Para que la sincronización y la descarga del script sea factible, deben darse las condiciones de seguridad y conectividad necesarias. Estas condiciones son abstraídas por la condición *accept*, que depende del estado del perfil del dispositivo receptor de la descarga y de cuál sea la credibilidad (a través, por ejemplo, de una autoridad de certificación) del dispositivo que proporciona el script. Esta función de aceptación también permitirá limitar las descargas de un script que se puedan realizar de un determinado dispositivo, evitando una sincronización continuada y sin límites del mismo dispositivo. La regla PAR modela la interacción paralela usual entre componentes cuando cualquiera de ellas puede progresar independientemente.

4. Conclusiones

En trabajos anteriores se ha abogado por el uso de arquitecturas de computación móvil para una transición en la tecnología que considere la información sobre las personas para adaptar sus servicios a estas. En esta línea, proponemos una arquitectura móvil para inferir y compartir perfiles sociológicos virtuales de los usuarios. El propósito de esto es desarrollar una interfaz transparente y automática que se

encargue de personalizar las interacciones de los usuarios con los dispositivos de su entorno. Las interacciones que proponemos serán llevadas a cabo en un framework de programación dinámica que permita la configuración de los dispositivos en cada interacción de acuerdo con las preferencias y necesidades de los usuarios. En esta ocasión mostramos nuestro trabajo en curso (aún en un estado muy preliminar) para tratar de formalizar estas interacciones programáticas de forma que sea posible satisfacer todos sus requerimientos de paralelismo, robustez, sincronización, seguridad y privacidad. Para ello basaremos nuestra formalización en Linda siendo un lenguaje de coordinación muy aceptado e idóneo para la naturaleza de nuestro sistema. Es necesario establecer ciertas peculiaridades como la convivencia de dispositivos de diversa naturaleza y la sincronización entre ellos durante las interacciones. Una vez completada la formalización de este enfoque, tenemos previsto analizar diversas propiedades habituales en los sistemas de transición que nos permitan analizar sistemas que sigan la filosofía PeaaS. En particular, la bisimilitud asociada a $\stackrel{\cdot}{\longrightarrow}$ debería ser una congruencia con objeto de permitir análisis composicional. Así mismo, formalizaremos diversos escenarios para ilustrar estas posibilidades de análisis.

Referencias

[1] Carriero, N., Gelernter, D.: Linda in context. Commun. ACM **32**(4), 444–458 (Apr 1989). https://doi.org/10.1145/63334.63337, http://doi.acm.org/10.1145/63334.63337

[2] De Nicola, R., Ferrari, G.L., Pugliese, R.: Klaim: a kernel language for agents interaction and mobility. IEEE Transactions on Software Engineering **24**(5), 315–330 (May 1998). https://doi.org/10.1109/32.685256

[3] Gelernter, D., Carriero, N.: Coordination languages and their significance. Commun. ACM **35**(2), 96– (Feb 1992). https://doi.org/10.1145/129630.376083, http://doi.acm.org/10.1145/129630.376083

[4] Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): A vision, architectural elements, and future directions. Future generation computer systems **29**(7), 1645–1660 (2013)

[5] 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)

[6] Pérez Vereda, A., Flores-Martín, D., Canal, C., Murillo, J.M.: Un framework de programación dinámica para iot. In: Jornadas de Ciencia e Ingeniería de Servicios JCIS, Sistedes 2018 (2018)

[7] Picco, G.P., Murphy, A.L., Roman, G.C.: Lime: Linda meets mobility. In: Proceedings of the 21st International Conference on Software Engineering. pp. 368–377. ICSE '99, ACM, New York, NY, USA (1999). https://doi.org/10.1145/302405.302659, http://doi.acm.org/10.1145/302405.302659

[8] Taivalsaari, A., Mikkonen, T.: A roadmap to the programmable world: software challenges in the IoT era. IEEE Software **34**(1), 72–80 (2017)