Coordinación de dispositivos en ambientes ubicuos mediante coreografías

Tesis doctoral desarrollada en la Universidad Nacional de San Luis - San Luis - Argentina

Defendida el 17 de Noviembre de 2020

Oscar A. Testa[1], Germán A. Montejano (director)[2], and Oscar Dieste (codirector)[3]

[1]Universidad Nacional de La Pampa, Facultad de Ciencias Exactas y Naturales, La Pampa - Argentina, otesta@exactas.unlpam.edu.ar
[2]Universidad Nacional de San Luis, Facultad de Ciencias Físico, Matemáticas y Naturales, San Luis - Argentina, gmonte@unsl.edu.ar
[3]Universidad Politécnica de Madrid, Escuela Técnica Superior de Ing. Informática, Madrid - España, odieste@fi.ump.es

Resumen

Introducción: Actualmente nos encontramos involucrados en ambientes donde los dispositivos ubicuos forman parte de nuestra vida cotidiana y de nuestras tareas diarias. De forma permanente estamos interactuando con dichos dispositivos y más aún, con los servicios que ellos nos brindan. En casi todos los casos, los dispositivos ubicuos no proporcionan servicios de forma aislada, sino que deben cooperar con otros dispositivos. ***Objetivo:*** El objetivo principal de esta investigación es el de definir un mecanismo de coordinación de dispositivos ubicuos que garantice su interoperabilidad independientemente del modelo y fabricante del mismo; utilizando los estándares de SOA y de coreografías para la composición de servicios. ***Metodología:*** Se ha utilizado como metodología de investigación design science, ya que es la que mejor se adapta a la naturaleza del problema, planteando como uno de sus lineamientos la construcción de artefactos y su posterior evaluación. ***Resultados:*** Se ha obtenido un framework de coordinación de dispositivos a través de la utilización de coreografías, que funciona de manera correcta y dando soporte a las características distintivas de los dispositivos ubicuos. La solución planteada es simple, interoperable y extensible.

1. Introducción

Dispositivos ubicuos son aquellos dispositivos electrónicos que tienen capacidad de procesamiento y comunicación, y pueden ser encontrados en cualquier lugar: la oficina, el auto, la casa, o la misma ropa con la que vestimos [1].

La computación ubicua es un paradigma tecnológico que pretende que las computadoras no se perciban como objetos diferenciados; así como también, que su utilización sea lo más transparente y cómoda posible para las personas, en las diversas situaciones en que existe interacción [1].

Los avances tecnológicos (especialmente de hardware y comunicaciones) han permitido que los dispositivos ubicuos sean al mismo tiempo generadores y consumidores de servicios; es decir, de acuerdo a las capacidades del dispositivo, éste pueda no solo obtener, sino también ofrecer su funcionalidad; lo que se traduce en un ambiente de cooperación entre dispositivos, permitiendo componer funcionalidades más complejas. Por composición, entendemos la forma en que los dispositivos ubicuos se pueden combinar para realizar una tarea determinada. La composición implica que los dispositivos deben comunicarse entre ellos con la finalidad de obtener un servicio con valor agregado, lo que a

su vez conlleva desafíos tales como: tolerancia a fallas, escaso nivel de procesamiento, problemas de conectividad, por mencionar algunos ejemplos [2].

Si bien hoy en día podemos decir que distintos dispositivos se pueden comunicar entre ellos, compartiendo de alguna manera sus servicios, generalmente lo realizan a partir de protocolos propietarios y sin seguir definiciones estándar, provocando que otros dispositivos no puedan ser integrados para la comunicación. Esto representa una importante limitación en la composición de dispositivos ubicuos [2].

La composición de dispositivos ubicuos presenta desafíos adicionales tales como: la heterogeneidad de los mismos, las contingencias de los dispositivos, y la personalización de los mismos (ej: provisión de servicios de acuerdo a las preferencias del usuario). Dado que los dispositivos ubicuos poseen limitaciones de recursos (ej. poca memoria y batería), se deben hacer consideraciones especiales respecto a la eficiencia y rendimiento de la composición de dispositivos. Todas estas dificultades hacen que la composición de dispositivos ubicuos constituya un área de investigación importante, donde los avances no son claros al día de hoy [2].

Esta situación nos motivó a plantear una propuesta que aglutine las especificaciones y estandarizaciones existentes en SOA para la coordinación de dispositivos ubicuos. Si pensamos que cada dispositivo ubicuo es proveedor o consumidor de un servicio, encaja perfectamente en la arquitectura de servicios.

Para poder llevar adelante este tipo de investigación, y de acuerdo a la naturaleza misma de la solución que intentamos encontrar, nos basamos en la metodología de investigación Design Science, ya que la misma se basa en la construcción y evaluación de un artefacto con la intención de dar solución a problemas debidamente identificados [3].

En este caso, nuestro objetivo principal es la definición de un mecanismo de coordinación de dispositivos ubicuos que garantice su interoperabilidad independientemente del modelo y fabricante del mismo; utilizando los estándares de SOA y de coreografías para la composición de serivicos. Para poder alcanzar este objetivo principal hemos planteado otros objetivos más específicos, de los cuales se desprende la principal contribución de este trabajo de Tesis.

La contribución consiste entonces, en la construcción de un framework de ejecución de coreografías en ambientes pervasivos a través de la utilización de dispositivos ubicuos. Este framework, por lo tanto, al estar basado en SOA es estandarizado, lo que conlleva a que se evite el problema en la heterogeneidad de dispositivos, además de permitir la utilización de características más avanzadas (dependiendo de la potencia y capacidad que posea el dispositivo) como es el caso de WS-Transaction, WS-Security, etc. Además, permitirá la interconectividad entre dispositivos ubicuos y servidores arbitrarios para llevar adelante la composición.

2. Motivación

Los mecanismos de composición de servicios, como las orquestaciones y coreografías, son aspectos bien conocidos en SOA (Service Oriented Architecture) que permiten construir sistemas de negocio complejos y aplicaciones a partir de una gran cantidad de servicios heterogéneos, simples y distribuidos. Estos conceptos podrían ser aplicables a ambientes ubicuos, en especial las coreografías, las cuales pueden fácilmente interpretarse en términos de dispositivos y ambientes ubicuos. Sin embargo, en determinados ambientes donde los servicios son dinámicos, móviles, menos fiables y dependientes del dispositivo, los mecanismos de composición establecidos para servicios web no son directamente aplicables [16].

Adicionalmente, la composición de múltiples dispositivos ubicuos presenta nuevos desafíos que no son compatibles con la composición de servicios web. En particular, los mecanismos de composición en ambientes masivos [1] necesitan hacer frente a distintas contingencias que pueden ocurrir con estos elementos, así como también contemplar la heterogeneidad de los mismos. Estos dispositivos tienen distintas limitantes como son la cantidad de memoria disponible, la durabilidad de la batería, la disponibilidad de acuerdo a la red del lugar donde se encuentren en un momento determinado, etc.

[1] Por ejemplo el de dispositivos móviles

En ambientes ubicuos, la disponibilidad y confiabilidad de los dispositivos no puede ser garantizada. No obstante, a pesar de todas las dificultades, existen ventajas que podría brindar la adaptación de los conceptos de SOA para la composición de dispositivos ubicuos, como son la estandarización tanto de los protocolos como de los mecanismos de comunicación entre los dispositivos, y la compatibilidad con otras plataformas de servicios ya existentes.

Las similaridades entre la composición de servicios web y la coordinación de dispositivos ubicuos es sorprendente. Si pensamos que cada dispositivo ubicuo en un ambiente pervasivo [2] es proveedor, o, consumidor de un servicio, la coordinación de dispositivos encaja perfectamente con la composición de servicios. Es también sorprendente que esta similitud no haya sido apenas explorada con anterioridad, salvo en el trabajo de Sheng [2] se hace mención a ello.

A finales de la década de los 80 y principios de los 90, Mark Weiser introdujo el término de computación ubicua (también utilizó el acrónimo "ubicomp") [1]. La teoría de Weiser postula que la computación ubicua tiene como propósito mejorar el uso de las computadoras, haciéndolas disponibles en el entorno físico, a través de una multiplicidad de elementos. El problema de esta teoría fue que los protocolos de comunicación existentes, no se alineaban con ella y debieron ser mejorados, en especial para permitir la movilidad de los dispositivos [1].

Se han realizado diversos proyectos de entornos ubicuos como **Aura** [4] de la Universidad de Carnegie Mellon, cuyo objetivo principal es el de proveer a cada usuario con un halo invisible de servicios de información, más allá del lugar que se encuentre; **Gaia** [5] de la Universidad de Illinois donde se plantean que los espacios físicos se convierten en espacios activos a partir de la utilización de dispositivos ubicuos y proponen un sistema operativo para manejar todos estos elementos en conjunto; **Oxigen** [6] perteneciente al MIT (Massachusetts Institute of Technology) donde se trabaja para que los dispositivos sean incorporados en la vida humana y cotidiana de manera natural e imperceptible.

Si bien estos proyectos han sido un avance para la integración de los dispositivos, no se ha logrado hacer que los mismos trabajen de forma independiente y colaborativa para la obtención o realización de una tarea específica, sin necesidad de ser coordinados a través de un nodo central.

Para la coordinación de dispositivos (o sistemas) ubicuos existen algunos mecanismos, como por ejemplo los frameworks de desarrollo. Si bien estos frameworks han presentado un avance respecto de la integración de los dispositivos ubicuos con otros elementos de computación, no son suficientes para que los mismos puedan interactuar de manera independiente.

Existen actualmente también trabajos de investigación relacionados con la coordinación de dispositivos ubicuos en ambientes pervasivos, aunque los mismos no han logrado trabajar sobre el tema de esta tesis en profundidad.

Todas estas características hacen que la composición de dispositivos ubicuos se configure en un área de investigación muy importante donde los avances han sido limitados al día de hoy.

3. Metodología de Investigación

La computación orientada a servicios, y en particular los servicios web, proporcionan mecanismos para la composición de servicios. Las orquestaciones, por ejemplo, son mecanismos bien conocidos que permiten construir sistemas de negocio complejos a partir de una gran cantidad de servicios heterogéneos, simples y distribuidos.

Nuestro principal objetivo de investigación es *Definir un mecanismo de coordinación de dispositivos ubicuos que garantice su interoperabilidad independientemente del modelo y fabricante del mismo; utilizando los estándares de SOA y de coreografías para la composición de servicios.* Si pensamos que cada dispositivo ubicuo en un ambiente pervasivo es proveedor, o consumidor de un servicio, la coordinación de dispositivos parece encajar perfectamente con la composición de servicios. Sin embargo, en contextos tales como el de Internet de las cosas (IoT de las siglas del inglés Internet of

[2] Ambientes pervasivos son entornos poblados por varios dispositivos (sensores, actuadores, etc) y aplicaciones de software integrados de forma transparente [1]

Things), donde los servicios son dinámicos, móviles, menos fiables y dependientes del dispositivo, los mecanismos de composición establecidos para servicios web no son directamente aplicables [16].

Por otra parte, la composición en ambientes pervasivos, como es el caso de las redes de sensores, dispositivos *wearables*, etc., necesita hacer frente a las distintas contingencias que pueden ocurrir con estos elementos, sin descuidar la heterogeneidad de los mismos. No debemos olvidar, que los dispositivos en mención tienen distintas limitantes como son la cantidad de memoria disponible, la durabilidad de la batería, la disponibilidad de acuerdo a la red del lugar donde se encuentre en un momento determinado. Por tanto, en ambientes pervasivos, la disponibilidad y confiabilidad de los dispositivos no puede ser garantizada.

Si bien las semejanzas entre la tecnología SOA y su aplicabilidad a sistemas ubicuos es muy importante, como mencionábamos, la mera traslación de los conceptos no alcanza. Para ello es necesario adaptar los conceptos de SOA para poder dar lugar a soportar las características propias de los dispositivos ubicuos. Es sorprendente que esta similitud no haya sido apenas explorada con anterioridad. Únicamente en el trabajo de Sheng [2] se hace mención a la necesidad de más investigación en esta área. En miras de alcanzar este propósito, nos hemos planteado las siguientes preguntas de investigación:

1. ¿Es SOA capaz de actuar como mecanismo de coordinación de sistemas ubicuos?.
2. ¿Es posible desarrollar un framework que permita incorporar SOA a dispositivos ubicuos?.
3. ¿Es posible demostrar la aplicabilidad de SOA en ambientes ubicuos mediante una prueba de concepto?.

De acuerdo a la naturaleza del problema y en base al objetivo planteado en el presente trabajo de investigación, la metodología seleccionada para llevar adelante el proyecto es "Design Science". Design Science crea y evalúa artefactos de tecnologías de la información con la intención de dar solución a problemas debidamente identificados, tal como es expresado en [3]. Los artefactos que se crean o evalúan a partir de Design Science van desde software, lógica formal y matemática rigurosa, hasta descripciones informales en lenguaje natural.

En la Figura 1 podemos apreciar las fases que plantea la metodología Design Science asociados a las preguntas de investigación planteadas.

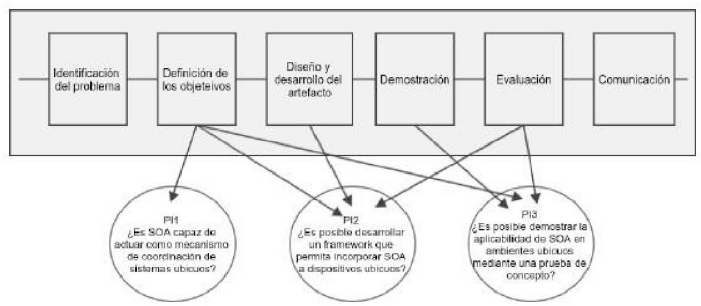

Figura 1: Metodología de investigación

4. Contribución

La principal contribución de este trabajo es el framework de ejecución de coreografías, incluyendo a los dispositivos ubicuos como parte importante de él, funciona de manera correcta y dando soporte a las características propias de los dispositivos ubicuos. La solución planteada es simple, interoperable y extensible. Hemos logrado trasladar, a un espacio o entorno donde las soluciones realizadas eran ad-hoc, conceptos de una teoría que existía únicamente en SOA, cuyo propósito es el de realizar sistemas

interoperables de forma estandarizada y transparente. Esto permite que los dispositivos ubicuos puedan cooperar a la hora de realizar sus tareas con ordenadores, teléfonos móviles, etc. que implementen la interfaz de SOA de forma transparente.

El framework de ejecución de coreografías[3] se construyó en base a los lenguajes de programación PHP y C++. Estos lenguajes fueron seleccionados en función de los dispositivos que se utilizaron para la prueba de concepto. La base de la ejecución de coreografías se centra en una serie de clases que, en principio, leen la descripción en WS-CDL (del inglés Web Service Choreography Description Language) de la coreografía, y en base al dispositivo que lo ejecuta; determina en primer lugar en qué posición de la ejecución de la coreografía se encuentra para luego poder determinar cuál o cuáles son los pasos siguientes en la ejecución. Una vez determinados los pasos que se deben ejecutar, se hacen las invocaciones a otros dispositivos, teniendo en cuenta las actividades descriptas en la definición de la coreografía en el lenguaje WS-CDL. Esta ejecución se hace de manera controlada, en el sentido de examinar que no se produzcan contingencias provenientes de las características de los dispositivos (desapariciones, latencia en la respuesta por falta de capacidad de procesamiento, etc.). En base a esta verificación se realizan las tareas correctivas correspondientes, según se han expresado en la definición de la coreografía. La estructura de ejecución del framework la podemos apreciar en la Figura 2. Allí se presentan los pasos que se llevan adelante dentro o desde un dispositivo que compone la coreografía, donde en primer lugar se determina cuál o cuáles son los pasos siguientes y los va ejecutando según corresponda. Como vemos la secuencia de ejecución es simple, clara y aplicable a cualquier dispositivo.

Para comprender mejor la estructura del framework desarrollado, mostraremos como ejemplo, un código escrito en el lenguaje de especificación de coreografías WS-CDL donde se aprecia la interacción entre dos dispositivos que forman parte de la coreografía [4]. Desde el primer dispositivo denominado VehiculoAccidentadoRole se produce una comunicación con otro dispositivo BalizaRole, éste último a su vez se relaciona con otro dispositivo CentralBalizaRole. Esta coordinación se pueden apreciar en las dos interacciones definidas. Si observamos en la Figura 2, el paso *Determinar siguiente* hace la tarea de fijarse en la definición XML de la coreografía a qué dispositivo debe llamar, en el ejemplo que mostramos debería leer la interacción que corresponde y fijarse cuál es el dispositivo que figura como *toRole*; si esto fuese el caso de la primer interacción del listado, debería invocar o coordinar con el dispositivo que se denomina BalizaRole.

```
<interaction name="reportarAccidente" operation="informarIncidente" >
  <participate relationshipType="tns:Vehiculo_Baliza" fromRole="tns:VehiculoAccidentadoRole" toRole="
      tns:BalizaRole" />
  <exchange action="request" name="informarIncidente" informationType="tns:avisoIncidenteType">
    <send variable="cdl:getVariable(tns:DatosIncidente,VehiculoAccidentadoRole)"/>
    <receive variable="cdl:getVariable(tns:DatosIncidente,BalizaRole)"/>
  </exchange>
</interaction>

<interaction name="publicarAccidente" operation="publicarIncidente">
  <participate relationshipType="tns:Baliza_CentralBaliza" fromRole="tns:BalizaRole" toRole="
      tns:CentralBalizasRole" />
  <exchange action="request" name="informarIncidente" informationType="tns:avisoIncidenteType">
    <send variable="cdl:getVariable(tns:DatosIncidente,BalizaRole)"/>
    <receive variable="cdl:getVariable(tns:DatosIncidente,CentralBalizasRole)"/>
  </exchange>
</interaction>
```

A continuación mostramos cómo se implementan los pasos de la ejecución de la coreografía, descriptos en la figura 2, pero ya en el código propiamente dicho. En la figura 3 se pueden apreciar los diagramas de clases en C++; éstas clases son las encargadas de llevar adelante la ejecución de la coreografía. En este diagrama podemos apreciar también que se encuentran ya implementadas las clases que dan manejo a una transacción dentro de la ejecución.

Para poder hacer uso del framework para un caso específico se deben incluir las librerías que implementan las clases dentro del código específico de la coreografía que se desea llevar adelante. Además, se deben generar clases que heredan de las clases preexistentes en el framework. A continuación se muestra un template de una clase que implemente y haga uso del framework. Más específicamente,

[3]El código fuente del framework desarrollado, puede ser accedido a través de esta dirección: `https://github.com/GRISE-UPM/ml_server_rest`.

[4]Solamente se muestra un trozo de la totalidad de la especificación XML, a los fines prácticos

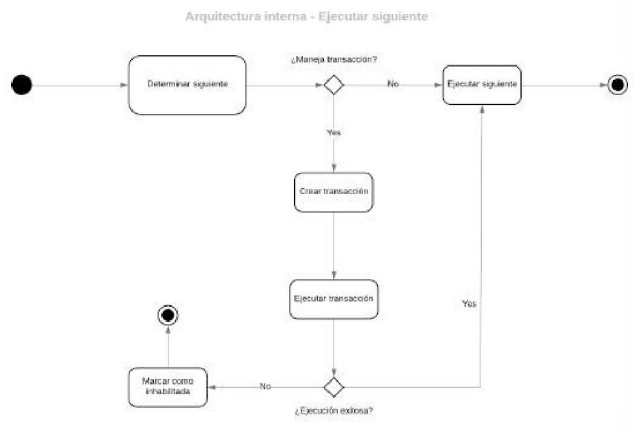

Figura 2: Diagrama de arquitectura de ejecución

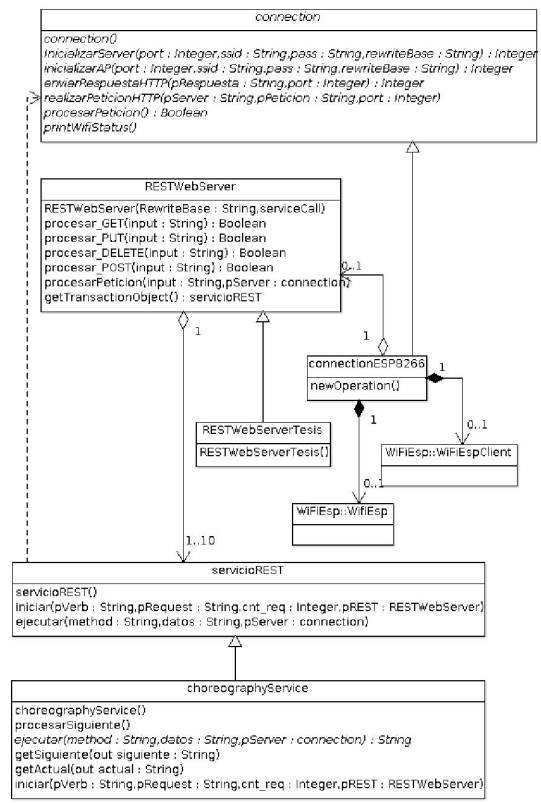

Figura 3: Diagrama de clases para ejecución de coreografías

Se deben incluir las librerías connection y RESTWebServer, y se deben implementar las clases constructoras, iniciarlas y ejecutarlas. En este caso mostramos el código que implementa el dispositivo BalizaRole, el cual es parte de las interacciones mostradas en la definición previa.

```
#include "ClaseX.h"
#include <connection.h>
#include <RESTWebServer.h>

BalizaRole::BalizaRole() : choreographyService(){
    chor_act_role = "BalizaRole";

    return;
}

int BalizaRole::iniciar(){
  return choreographyService::iniciar();
}

void BalizaRole::ejecutar(){
  if (strcmp(method,"informarIncidente") == 0){
    if (this->_verb != 'O'){
      pServer->enviarRespuestaHTTP("405_Method_not_allowed","");
      return;
    }
    String respuesta = this->informarIncidente(method,datosIncidente);
    strcpy(sRespuesta,"{\"resultado\":\"");
    strcat(sRespuesta,respuesta.c_str());
    strcat(sRespuesta,"\",\"token\":");
    strcat(sRespuesta,_token);
    strcat(sRespuesta,"}");
    pServer->enviarRespuestaHTTP("200_OK",sRespuesta);

    // LLamo a que se ejecute lo que tenga que seguir de la coreografia
    choreographyService::ejecutar();
}

}
```

El template mostrado corresponde a la programación en lenguaje C++; no obstante, la programación en PHP es muy similar a la presentada, incluso al ser un lenguaje de más alto nivel, su implementación es mucho más simple.

Para validar la funcionalidad del framework de coordinación, se ha seleccionado una prueba de concepto basada en un escenario de trabajo específico; no obstante, creemos que esta solución podría ser extrapolada hacia prácticamente cualquier ambiente de trabajo.

Se planteó un ambiente inteligente relacionado directamente con los problemas referentes al tráfico vehicular a lo largo de las autopistas, rutas o carreteras. Simulamos vehículos que transitan por una carretera, los cuales se comunican con balizas distribuidas a lo largo de la vía. Esta comunicación puede ser bi direccional, cuando el vehículo informa de un problema encontrado o bien que la baliza da aviso de posibles problemas a lo largo de la carretera. A su vez las balizas pueden comunicarse con centrales de coordinación o de emergencia según corresponda. Una aplicación real de este escenario podría ser que un ómnibus con pasajeros se traslada desde una ciudad a otra a través de una carretera normal (con dos carriles), el conductor del vehículo cuenta con un sensor que detecta la posibilidad de que esté por sufrir un ataque cardíaco, o más aún, que ya lo esté sufriendo en ese momento (este dispositivo puede ser una placa de desarrollo Arduino). A partir de ese instante, el vehículo le avisa al conductor (ya sea visualmente como auditivamente), a su vez intenta comunicarse con una baliza (la cual puede ser representada por una placa Arduino) de la carretera para que la ayuda llegue lo antes posible al lugar del evento. También podría disparar alertas a los vehículos cercanos, para que los mismos puedan tomar acciones preventivas, además de avisar al pasaje del ómnibus y detener la marcha en caso de ser necesario.

Los dispositivos utilizado para la construcción de la prueba de concepto son los siguientes: Equipo servidor, Equipo Laptop, Equipo RaspberryPi B+ y Placas de desarrollo Arduino Mega 2560 y Arduino Nano V3.

El diagrama de secuencia de la Fig. 4 representa la ejecución de la coreografía, en base a la prueba de concepto definida. Esta figura se genera automáticamente desde el mismo framework ya que por cada ejecución que se hace de un caso de prueba, se genera un código de identificación único que es informado en la ejecución para luego poder obtener el gráfico correspondiente. En el diagrama se pueden observar, como etiquetas en la secuencia de mensajes, el formato de información que viaja desde un dispositivo hacia otro, el cual está definido dentro de la definición de la coreografía realizada en WS-CDL. De acuerdo al extracto en XML que mostramos previamente, las dos interacciones que allí presentamos

se ven representadas en este diagrama como las interacciones marcadas con las numeraciones 1, 2, 3, 7 y 10 respectivamente.

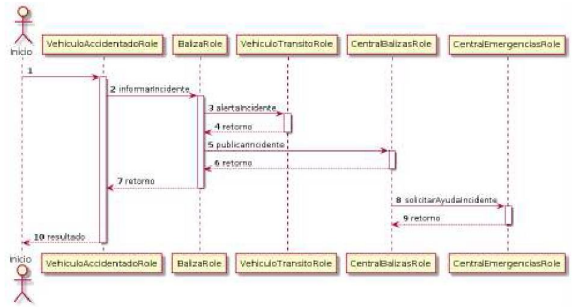

Figura 4: Diagrama de secuencia de ejecución de la coreografía

Este escenario que se plantea puede ser fácilmente abstraído a otras situaciones como puede ser un sistema de alarma hogareño, coordinación de una flota de distribución de mercadería, provisión de servicios a pasajeros en tránsito dentro de un aeropuerto, etc.

5. Conclusiones y Trabajo Futuro

La solución planteada, esto es, la utilización del estándar WSCDL de coreografías para SOA, junto con el framework desarrollado, ha resuelto satisfactoriamente el problema planteado.

El principal problema que la presente tesis ha afrontado, el cual está descrito en detalle en la Sección de Planteamiento del problema, consistía en que la composición de dispositivos se realizaba a través de protocolos propietarios y sin seguir definiciones estándares. Esto ha provocado que dispositivos de distintos proveedores no puedan ser utilizados en una composición, presentando limitaciones importantes a la hora de realizar composiciones con estos dispositivos.

La solución planteada permite realizar composiciones con dispositivos ubicuos de una manera abierta, basada en estándares y escalable. Esto implica que diversos dispositivos, de diversos fabricantes, y con distintas capacidades, pueden ser incorporados fácilmente al framework y, por consiguiente, participar en coreografías con otros dispositivos de forma sencilla.

El framework desarrollado no posee gran complejidad. Es bastante simple desde el punto de vista de diseño, y no demasiado exigente en términos de procesamiento y memoria. Por ello, creemos que la solución planteada puede aplicarse en sectores comerciales como, por ejemplo, el de sistemas de seguridad hogareña, donde se podría utilizar el framework desarrollado para implementar sistemas más sofisticados que los actuales, con la posibilidad de manejar una mayor cantidad de dispositivos, realizando tareas más complejas que la generación de una simple señal de activación, como ocurre hoy en día. Otro sector en donde podría aplicarse el framework desarrollado es en la atención domiciliaria de pacientes con enfermedades que deban ser monitorizados a distancia, incluso con posibilidad de poder aplicar medicación en caso de ser necesario.

La solución planteada también es aplicable a IoT, donde los dispositivos no solamente estarían conectados entre sí, sino que también podrían utilizar otros protocolos de comunicación más allá de TCP/IP.

La solución propuesta permite, adicionalmente que los dispositivos ubicuos pueden interactuar no sólo con otros dispositivos ubicuos, sino también con aplicaciones basadas en SOA. De esta manera, los dispositivos pueden ser integrados en sistemas existentes ampliando enormemente sus capacidades.

Finalmente, la solución propuesta no es, ni mucho menos, completa. Es posible, y deseable, realizar toda una serie de mejoras y ampliaciones que describimos en a continuación.

De la mano con los hallazgos alcanzados, se han abierto tanto nuevas líneas de investigación como también posibilidades de producción de software de uso industrial. A continuación reportamos las futuras opciones de investigación y desarrollo que esta tesis ha propiciado.

5.1. Líneas futuras de investigación

- Descubrimiento de servicios: Durante la presente investigación se ha trabajado sobre servicios y dispositivos ubicuos estáticos, es decir, preestablecidos dentro de la coreografía y presentes en la red de comunicaciones antes del inicio de cualquier interacción. Resultaría una línea de investigación muy interesante la de poder coordinar a través de coreografías dispositivos ubicuos descubiertos "on the fly".

- Introducción de capas de seguridad: Una línea de investigación que también resultaría en un notable progreso a este trabajo de tesis es la relacionada con la introducción dentro del framework de coordinación de las capas de seguridad existentes en servicios web como es el caso de WS-SECURITY. La inclusión de esta capa de seguridad significa un desafío importante ya que se debe estudiar de qué manera es posible incorporarla dentro de dispositivos con poca capacidad de procesamiento y escasa memoria.

- Implementación de transacciones de acuerdo a estándares: Otra línea de investigación, relacionada en cierta forma con la anterior, es la de implementar la totalidad de los estándares de transacciones disponibles en SOA dentro del framework de coordinación propuesto en esta tesis. En esta investigación se realizó la implementación de una transacción distribuida, pero sin ajustarse en su totalidad a los estándares WS-TRANSACTION y WS-COORDINATION.

- Convergencia con microservicios: Hemos mencionado durante la sección de Discusión que existe una relación bastante estrecha entre esta tesis y la tecnología de microservicios. En microservicios se discute sobre la forma en que se pueden coordinar los servicios para llevar adelante tareas en conjunto, donde una de las formas presentadas es la de coreografías. Por lo tanto, una futura línea de investigación consistiría en la convergencia entre los conceptos emanados de este trabajo de Tesis y la tecnología de microservicios.

- Convergencia con IoT: Como hemos indicado en la sección de Discusión, IoT utiliza únicamente el protocolo TCP/IP para la interconectividad de los distintos componentes. También hemos mencionado que es una limitación que no se ha afrontado en el framework desarrollado en esta tesis. Por lo tanto, una futura línea de investigación consistiría en la convergencia entre ambas tecnologías ampliándolas a la utilización de otros protocolos como puede ser Bluetooth o RFID, por mencionar dos ejemplos destacados.

5.2. Líneas futuras de desarrollo

- Implementación del framwork de ejecución de coreografías sobre dispositivos ubicuos con calidad industrial: Para poder llevar adelante la investigación fue necesario implementar un framework de coordinación de dispositivos ubicuos tal y como exige la metodología de "Design science" elegida. Si bien ello fue suficiente para los fines de la presente investigación, el prototipo desarrollado no alcanza la calidad necesaria para ser aplicado efectivamente sobre problemas de coordinación de dispositivos ubicuos en la industria.

  Aunque sí deja las bases sentadas para que el desarrollo siga evolucionando hasta poder producir un framework sólido y de características industriales para que el mismo pueda ser utilizado en entornos reales.

- Ampliación de las pilas de protocolos: El objetivo en este caso sería mejorar la conectividad de los dispositivos que forman parte de una coreografía. Para ello se debería extender el framework para utilizar otros protocolos de comunicaciones, como puede ser el caso de la utilización de protocolos Bluetooth, RFID, etc. Esta mejora permitiría ejecutar coreografías no sólo con una mejor diversidad de dispositivos ubicuos sino también probablemente de forma más versátil, aprovechando las posibilidades que dichos protocolos ofrecen, ej: la comunicación a través de redes de sensores.

Referencias

[1] M. Weiser, "Hot topics-ubiquitous computing," *Computer*, vol. 26, pp. 71–72, Oct 1993.

[2] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, no. 0, pp. 218–238, 2014.

[3] S. T. March and G. F. Smith, "Design and Natural Science Research on Information Technology," *Decis. Support Syst.*, vol. 15, pp. 251–266, Dec. 1995.

[4] J. Harkes, T. Farbacher, and N. Miller, "Project Aura Distraction-free Ubiquitous Computing." Available at `http://www.cs.cmu.edu/~./aura/people.html`, Last visited: Aug 15th, 2002.

[5] R. H. Campbell, D. M. Mickunas, D. Reed, and K. Nahrstedt, "Active Spaces for Ubiquitous Computing." Available at `http://gaia.cs.illinois.edu/`, Last visited: Aug 15th, 2002.

[6] M. L. f. C. Science and M. A. I. Laboratory, "Pervasive Human-Centered Computing." Available at `http://oxygen.csail.mit.edu/`, Last visited: Aug 15th, 2002.

[7] "Amigo Project." Available at `http://gforge.inria.fr/projects/amigo/`.

[8] M. Viroli, "On competitive self-composition in pervasive services," *Science of Computer Programming*, vol. 78, no. 5, pp. 556–568, 2013. Special section: Principles and Practice of Programming in Java 2009/2010 & Special section: Self-Organizing Coordination.

[9] S. Najar, M. K. Pinheiro, and C. Souveyet, "A New Approach for Service Discovery and Prediction on Pervasive Information System," *Procedia Computer Science*, vol. 32, pp. 421–428, 2014. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014).

[10] S. W. Loke, "Supporting ubiquitous sensor-cloudlets and context-cloudlets: Programming compositions of context-aware systems for mobile users," *Future Generation Computer Systems*, vol. 28, no. 4, pp. 619–632, 2012.

[11] F. Palmieri, "Scalable service discovery in ubiquitous and pervasive computing architectures: A percolation-driven approach," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 693–703, 2013. Special Section: Recent Developments in High Performance Computing and Security.

[12] S. Cherrier, Y. M. Ghamri-Doudane, S. Lohier, and G. Roussel, "Services collaboration in wireless sensor and actuator networks: Orchestration versus choreography," in *2012 IEEE Symposium on Computers and Communications (ISCC)*, pp. 000411–000418, July 2012.

[13] C. Duhart, P. Sauvage, and C. Bertelle, "Emma: A resource oriented framework for service choreography over wireless sensor and actor networks," *International Journal of Wireless Information Networks*, 06 2015.

[14] L. Mostarda, S. Marinovic, and N. Dulay, "Distributed orchestration of pervasive services," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 166–173, April 2010.

[15] Z. Zhou, D. Zhao, L. Liu, and P. C. Hung, "Energy-aware composition for wireless sensor networks as a service," *Future Generation Computer Systems*, vol. 80, pp. 299 – 310, 2018.

[16] G. Cassar, P. Barnaghi, W. Wang, S. De, and K. Moessner, "Composition of services in pervasive environments: A Divide and Conquer approach," in *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pp. 000226–000232, July 2013.