Descripción semántica de aplicaciones web mediante microservicios

José Ignacio Fernández-Villamor, Carlos Ángel Iglesias, Mercedes Garijo
Departamento de Ingeniería de Sistemas Telemáticos,
Universidad Politécnica de Madrid
Avenida Complutense nº 30, "Ciudad Universitaria". 28040 - Madrid (España)
jifv@dit.upm.es, cif@dit.upm.es, mga@dit.upm.es

***Resumen*—Las posibilidades de desarrollo de aplicaciones en la web actual están limitadas por la interoperabilidad entre aplicaciones y servicios. La investigación en el campo de la web semántica, servicios web semánticos y mashups busca contribuir a la interoperabilidad y composición de servicios y aplicaciones, aunque muchas alternativas no han logrado suficiente adopción debido a su complejidad.**

En este artículo se propone un método de clasificación de servicios ligero para el estilo arquitectónico REST, presentándose la idea de Microservicios. Frente a anteriores enfoques de servicios web semánticos que buscan la descripción de cualquier tipo de servicio web, en el método propuesto se pretende poder definir descripciones subóptimas sencillas y naturales, llamadas descripciones de microservicios. Las descripciones de microservicios consisten en una lista de términos que representan características de un servicio. Los beneficios de desarrollar una aplicación web mediante microservicios son la generación automática de documentación de la interfaz de programación, la posibilidad de automatizar las pruebas de la aplicación, o la capacidad de descubrir, ejecutar y componer servicios de aplicaciones web.

***Palabras Clave*—REST, aplicaciones web, web programable, servicios web**

I. Introducción

La web actual posee un creciente número de aplicaciones y servicios que cubren diferentes dominios de interés. Actualmente, los usuarios disfrutan de un amplio abanico de aplicaciones, desde aplicaciones de comercio electrónico a blogs, noticias o redes sociales. Las posibilidades de la web actual están limitadas únicamente por la interoperabilidad entre aplicaciones. El número de casos de uso de Internet se multiplicaría si las aplicaciones pudiesen componerse arbitrariamente y ser ejecutadas automáticamente para satisfacer la meta de un usuario. Este enfoque de construir una web programable ha llevado a la investigación en múltiples áreas, como servicios web semánticos y mashups o combinaciones de aplicaciones. Muchos de estos enfoques han sido exitosos desde un punto de vista investigador, pero no han sido capaces de lograr gran adopción debido a su alta complejidad y a su falta de integración con la arquitectura de Internet.

En este artículo se propone un método ligero de anotación y clasificación de servicios que sigue el estilo arquitectónico REST y que posibilita descubrimiento, ejecución, composición, documentación y pruebas automáticas de aplicaciones web. Dichas descripciones se realizan en un lenguaje natural, buscándose la facilidad de llegar a descripciones subóptimas sencillas de servicios, sin pretender cubrir todo el espacio de servicios existente en Internet.

En el artículo, en primer lugar se describe el estado del arte de tecnologías similares y estudios relacionados, así como la problemática existente. En segundo lugar, proponemos nuestro método de clasificación analizando sus características y casos de uso. Finalmente, se resumen las principales conclusiones del trabajo de investigación realizado.

II. Trabajos relacionados

Hay diversos enfoques que contribuyen a la mejora de la integración de información y la composición de servicios.

La arquitectura de servicios web [4] y los estándares de servicios web semánticos que se definen sobre ella (como OWL-S [14], WSMO [11], METEOR-S [10] o WSDL-S [13]) proporcionan un enfoque pesado al descubrimiento, ejecución y composición de servicios. Algunas alternativas han sido empleadas por empresas en sus buses de servicios, mientras que las alternativas semánticas han sido principalmente utilizadas en contextos de investigación. En general, puede decirse que esta arquitectura no ha gozado de gran adopción en Internet. Su falta de integración con el estilo arquitectónico REST y la alta complejidad de los estándares pueden ser razones de este fracaso.

Para mejorar la integración con la arquitectura de Internet, se comenzaron a utilizar enfoques REST en aplicaciones web, implementándose interfaces de programación separadas para las funcionalidades más comunes de las aplicaciones. Para mejorar la automatización y la interoperabilidad, se han propuesto múltiples alternativas con un enfoque REST.

SA-REST [12] propone anotar las especificaciones de interfaces de programación (APIs) para permitir descubrimiento, ejecución y composición automática de servicios REST. El enfoque asume que existe una API para funcionalidades comunes de la aplicación junto con una especificación textual disponible públicamente. Tabmién asume que la parte de las aplicaciones que son consumidas por humanos no son procesadas por agentes automáticos. Sobre ello, SA-REST propone anotar las especificaciones textuales de APIs con RDFa [2] para habilitar el procesado automático de APIs. Los esfuerzos de desarrollo siguiendo el enfoque de SA-REST residen en construir una API separada para las funcionalidades más comunes de la aplicación, escribir una especificación textual de la API y anotar esta especificación con RDFa para habilitar el procesado automático. hRESTs [15] es un enfoque similar, que propone un conjunto de microformatos para permitir la anotación de especificaciones de interfaces de programación.

Finalmente, el lenguaje de descripción de aplicaciones web (WADL) [7] es otro enfoque REST para descripción

de interfaces de servicios que, en una forma similar a SA-REST, propone definir un fichero XML que describe una API REST. El fichero WADL describe la API de una forma autocontenida y, a diferencia de SA-REST, no reutiliza la información disponible en la especificación textual de la API.

III. Descripción del problema

A. Los principios de diseño de Internet

La web fue creada con el objetivo de construir un sistema extensible y distribuido de hipermedios siguiendo el estilo arquitectónico REST (*Representational State Transfer*). REST consiste en un conjunto de restricciones que pueden ser aplicadas sobre cualquier otro sistema distribuido de hipermedios. En el caso de Internet, limita la arquitectura a un sistema cliente-servidor cacheable, en capas, sin estado, con facilidades de código bajo demanda y con una interfaz uniforme para sus recursos. Estas prácticas se utilizan en el diseño de la web, y han sido ignoradas ocasionalmente, llevando a desajustes en protocolos y problemas en aplicaciones web [5].

Respecto a los aspectos sociales relacionados con el desarrollo web, los desarrolladores normalmente desean reducir al máximo la complejidad de las aplicaciones. Típicamente, los enfoques en integración de la información y descripción de servicios están basados en gran medida en descripciones formales de servicios. Especificar formalmente el modelo de interfaz y proceso de servicios y APIs es una tarea que suele evitarse por su complejidad, a no ser que proporcione un valor añadido importante a la aplicación. Como conclusión, las iniciativas para hacer la web programable (por ejemplo, los mashups o los servicios web semánticos) necesitan equilibrar los esfuerzos de formalización con los beneficios a nivel de interoperabilidad y automatización.

Por ello, es importante considerar estos aspectos para definir una solución para automatización de servicios que se adapte a la filosofía de Internet y tenga adopción entre desarrolladores.

B. Limitaciones de la interfaz uniforme

Como se ha dicho anteriormente, la web actual sigue el estilo arquitectónico REST. El protocolo HTTP proporciona una forma de acceder a los recursos mediante una interfaz común que está basada, principalmente, en cuatro métodos o verbos: GET, PUT, DELETE y POST. Esto implica que la manipulación de un recurso ha de realizarse utilizando exclusivamente uno de estos verbos.

La seguridad y la idempotencia son propiedades que poseen algunos de estos métodos. Las operaciones seguras son aquellas que no tienen ningún tipo de efecto secundario sobre el estado de la web. Las operaciones idempotentes son aquellas cuyo resultado es el mismo tanto si han sido ejecutadas una vez como si han sido ejecutadas más veces. Una operación segura es, por tanto, idempotente.

El método HTTP GET solicita una representación de un recurso indicado por su identificador o URI, PUT almacena una representación de un recurso proporcionada como parámetro en una determinada URI, DELETE elimina un recurso especificado por su URI y POST procesa una representación dada como subordinada de un recurso dado por su URI. Por ello, GET es una operación segura e idempotente, PUT y DELETE son operaciones idempotentes pero inseguras y POST es una operación insegura y no idempotente.

¿Cómo afecta el estilo arquitectónico REST al desarrollo de aplicaciones web? Las aplicaciones web tienen un dominio o campo de interés (como, por ejemplo, el comercio electrónico o las redes sociales) en el que se involucran ciertas operaciones de negocio y conceptos particulares. Siguiendo el estilo arquitectónico REST, los conceptos deberían modelarse como recursos REST, mientras que las operaciones de negocio deberían modelarse como métodos de la interfaz uniforme.

Siguiendo la semántica de los métodos HTTP, las operaciones de lectura deberían asociarse con operaciones GET, las operaciones de almacenamiento como PUT, las de eliminación como DELETE y cualquier otra operación como POST. Como resultado, no es posible conocer la semántica de una operación que se realiza tras un método, al existir un infinito rango de operaciones de negocio realizable que se asocia a sólo cuatro métodos posibles.

Por ejemplo, en el caso de una aplicación de comercio electrónico, una operación de vaciado del carro de la compra debería implementarse mediante el método POST. Por otro lado, hacerse amigo de otra persona en una red social es otra operación que debería implementarse con un método POST. Ambas operaciones utilizan el mismo método de la interfaz uniforme de REST, pero no tienen el mismo significado ni el mismo efecto.

Por ello, se necesita información adicional para que agentes autónomos sepan la operación precisa que un método HTTP realiza, sin importar si esta información puede ser inferida del contexto o si se especifica mediante anotaciones. Esta es una desventaja de utilizar una interfaz uniforme en el estilo arquitectónico REST.

IV. Microservicios

En este artículo proponemos un enfoque para la descripción de servicios en aplicaciones web que sigue el estilo arquitectónico REST y que pretende tener un equilibrio entre efuerzos de desarrollo y el valor añadido proporcionado.

Otros enfoques proponen construir servicios paralelos a la parte de las aplicaciones web que es utilizada por los usuarios humanos. Para reducir esfuerzos de desarrollo, proponemos anotar las aplicaciones web de forma que los usuarios humanos y los agentes automáticos puedan seguir el mismo flujo de trabajo al utilizar una aplicación, en la línea de iniciativas como RDFa y POSH [8].

La filosofía que hay detrás de este enfoque reside en proporcionar a los clientes automáticos los mismos medios que los usuarios humanos para utilizar las aplicaciones web. Al utilizar una aplicación web, los usuarios humanos interaccionan con ella en respuesta a las salidas que proporciona la aplicación, siguiendo enlaces y enviando formularios de acuerdo a un comportamiento esperado de la aplicación que se deduce del contenido de las páginas web, tanto a nivel textual como contextual.

Iniciativas como los microformatos contribuyen a proporcionar una forma de describir los contenidos de las representaciones de los recursos web. En este enfoque, sin embargo, falta una manera de describir las interacciones con los recursos más allá de la semántica del método HTTP utilizado. Considerando un servicio en una aplicación como una combinación de un

recurso y un método HTTP, se propone a continuación la idea de microservicios para resolver esta problemática y así posibilitar la utilización de aplicaciones web por parte de agentes automáticos.

La implementación de una aplicación con microservicios permite varias funcionalidades adicionales, tales como:

- Descubrimiento de servicios automático por parte de motores de búsqueda.
- Descubrimiento, ejecución y composición automática de servicios por agentes inteligentes y combinadores de aplicaciones o *mashups*.
- Generación automática de documentación de servicios.
- Prueba automática de servicios.
- Asistencia en la navegación web, como rellenado de formularios.

A. Introducción a los microservicios

La web actual posee multitud de aplicaciones con gran cantidad de servicios diferentes a disposición de los usuarios. Existen múltiples iniciativas para describir servicios, como OWL-S o WSMO. Estas iniciativas intentan ofrecer un método para describir cualquier servicio posible, obteniéndose como resultado estándares muy complejos de utilizar. En este artículo proponemos un enfoque diferente, en el que se proporciona un conjunto cerrado de elementos para describir servicios, intentando hacer posible definir una descripción subóptima para cualquier tipo de servicio. Dado que está inspirado en la idea de Microformatos, llamamos Microservicios [6] a estos servicios subóptimos que pueden describirse. En otras palabras, suponiendo un espacio multidimensional en el que existen los diferentes servicios, los microservicios pretenden ser servicios con máxima similaridad con servicios de un mismo tipo.

Siguiendo el enfoque de descripción de servicios de WSMO, la semántica de un servicio se define a través de un conjunto de precondiciones, asunciones, postcondiciones y efectos [11] que, en general, son un conjunto de condiciones o restricciones requeridas y que se aplican ante la ejecución satisfactoria del servicio. Estas condiciones describen, por ejemplo, efectos secundarios, presencia de parámetros de entrada, o la semántica de la salida del servicio.

Las restricciones de un servicio pueden agruparse de acuerdo a las características de alto nivel que pretenden describir. Por ejemplo, si se necesita como parámetro una lista de palabras clave en un servicio de búsqueda, este hecho puede describirse con una precondición que exija la presencia de una variable de entrada `keywords` y una postcondición que implique que la salida debe incluir las palabras clave proporcionadas en `keywords`. Ambas condiciones buscan añadir una característica de filtrado por palabras clave al servicio, dando lugar a un elemento descriptivo reutilizable para otros servicios. El conjunto de características disponibles da lugar a la "caja de herramientas" que los desarrolladores web pueden utilizar para describir un microservicio.

En las siguientes subsecciones se definirá el concepto de microservicios en detalle mediante un modelo y se presentarán los intermediarios de microservicios.

B. Modelo de descripción de microservicios

Las descripciones de microservicios son descripciones ligeras para aplicaciones web siguiendo el estilo arquitectónico REST. En esta sección, el concepto de microservicio se clarifica definiendo un modelo de descripción de microservicios.

De aquí en adelante consideraremos un servicio como una interacción con un recurso de una aplicación web, es decir:

Definición 1. Un *servicio* es una dupla (r, m) que tiene un recurso r y un método m.

Por tanto, un recurso puede tener tantos servicios asociados como métodos HTTP existen. Siguiendo esta definición de servicio, se define el concepto de microservicio.

Definición 2. Un *microservicio* es un servicio que está descrito por una descripción de un microservicio y por su descripción extendida asociada.

Los conceptos de descripción de microservicio y descripción extendida se definen a continuación.

Definición 3. Una *descripción de microservicio* es un conjunto de términos t_1, t_2, ..., t_n. Cada término t_i representa una característica que tiene el servicio.

La descripción de un microservicio puede ser transformada en una descripción extendida mediante una función de transformación. Una descripción extendida de un servicio se define siguiendo el enfoque WSMO de descripción de servicios.

Definición 4. Una *descripción extendida de servicio* s es un conjunto de precondiciones, postcondiciones y descripciones textuales que describen un servicio. Es decir, $s \in \mathcal{P}(\mathcal{PRE} \cup \mathcal{POS} \cup \mathcal{TEX})$, siendo $\mathcal{PRE}$, $\mathcal{POS}$, y $\mathcal{TEX}$ los conjuntos de precondiciones, postcondiciones y descripciones textuales, respectivamente, y $\mathcal{P}(X)$ el conjunto potencia de X.

La función de transformación utiliza definiciones de términos para construir la descripción extendida.

Definición 5. Una *definición de un término* d es un conjunto de precondiciones, postcondiciones y descripciones textuales que se asocian a un conjunto de términos. Es decir, $d \in \mathcal{P}(\mathcal{T}) \times \mathcal{P}(\mathcal{PRE} \cup \mathcal{POS} \cup \mathcal{TEX})$, siendo $\mathcal{T}$ el conjunto de términos y $\mathcal{P}(\mathcal{T})$ su conjunto potencia.

Las precondiciones, postcondiciones y descripciones textuales en las definiciones cuyos términos están presentes en la descripción del microservicio se utilizan por la función de transformación para producir la descripción extendida del servicio.

Definición 6. Una *función de transformación* F es una función que combina definiciones de términos en una descripción de un microservicio para producir una descripción extendida. Es decir, $F : \mathcal{M} \rightarrow \mathcal{S}$, con $F(m) = \bigcup_i C_i \mid (T_i, C_i) \in \mathcal{D} \wedge T_i \subseteq m$, siendo $\mathcal{D}$, $\mathcal{M}$ y $\mathcal{S}$ el conjunto de definiciones, el conjunto de descripciones de servicios y el conjunto de descripciones extendidas de servicios, respectivamente.

Por tanto, la función de transformación construye una descripción extendida de un servicio de acuerdo con un vocabulario de términos. Las capacidades descriptivas de las

descripciones de servicios imponen los límites sobre qué es un microservicio y qué no lo es. Por ello, dado un determinado vocabulario de términos, algunos servicios serán microservicios al poder ser descritos mediante descripciones de microservicios, y otros no.

Un ejemplo de descripción de microservicio puede ser `keyword-filtered multiple picture get`, que consta de cuatro términos, y describe un servicio de búsqueda de imágenes por palabras clave. Su descripción extendida se muestra en la figura 1. Puede observarse que la descripción extendida posee una descripción textual, precondiciones y postcondiciones, posibilitando documentación automática, pruebas automáticas o ejecución automática, entre otras funcionalidades, como se detallará en la sección V.

Dicha descripción extendida se obtiene, de acuerdo con la definición 6, agrupando precondiciones, postcondiciones y descripciones textuales de aquellas definiciones de términos cuyos términos aparecen en la descripción del servicio. La descripción del ejemplo se construye mediante la combinación de las definiciones `keyword-filtered`, `multiple get`, `picture get` y `get`.

Este ejemplo ilustra la necesidad de que las condiciones impuestas por un término puedan ser dependientes de otros términos presentes en la descripción. Por ejemplo, el término `picture` impone una postcondición ("devolver una imagen") al aplicarse sobre `get`, pero impone una precondición ("una imagen debe proporcionarse como parámetro") al aplicarse sobre `post`.

En resumen, los microservicios proporcionan una descripción dual de servicios: (i) una descripción estándar, que consiste en una lista de términos que representan características de alto nivel y (ii) una descripción extendida, que proporciona una especificación formal del servicio. Ambas descripciones son procesables por agentes automáticos o humanos, mientras que utilizar una u otra depende del nivel de expresividad necesario para la tarea requerida.

C. Descripciones en lenguaje natural

El enfoque utilizado para las descripciones de microservicios encaja de forma natural con el lenguaje hablado. Cada término de una descripción añade restricciones y condiciones de la misma forma a cómo se hace en lenguaje natural. Por ello, eligiendo nombres para los términos a un alto nivel de abstracción permite obtener una descripción prácticamente en lenguaje natural.

Los métodos HTTP están representados por sus correspondientes términos. `get`, `post`, `put` y `delete` son términos que implican la utilización del respectivo método HTTP en el microservicio, añadiendo las convenientes precondiciones y postcondiciones.

Para incrementar la semántica de los métodos HTTP, pueden añadirse más términos para construir una descripción con más significado. Por ejemplo, el microservicio `picture post` implica que la entidad enviada es una imagen, mientras que `multiple get` implica que el recurso solicitado es una lista de recursos en un formato de lista acordado, como XOXO [9]. Una variante de este último ejemplo sería el microservicio `keyword-filtered multiple picture get`, que describiría un servicio similar a Flickr en el que se devuelven múltiples imágenes filtradas por unas palabras clave. Finalmente, `picture private sharing post` describiría un servicio en el que una imagen es enviada para su compartición con un conjunto privado de usuarios de una determinada red social.

Puede observarse que todas las descripciones son amigables y naturales, lo que hace que su construcción sea fácil y sencilla dado un determinado vocabulario de términos con el que trabajar.

D. Microservicios intermediarios

Con esta filosofía, rara vez habrá un microservicio que encaje perfectamente con otro servicio. En los casos en que no haya encaje, o bien el servicio se debe adaptar a la descripción del microservicio, o se ha de definir un intermediario que utilice el servicio pero se ajuste a la descripción del microservicio. En esta sección se tratará este último caso, la utilización de intermediarios o *proxies* de microservicios.

Definición 7. Un *microservicio intermediario* es un microservicio que (i) ejecuta otro servicio y (ii) tiene una descripción de microservicio que es la descripción subóptima del servicio externo.

Al utilizar un microservicio intermediario, el cliente del servicio debe conocer el servicio que se ejecuta realmente. Por ejemplo, http://www.flickr.com/search permite buscar imágenes en el dominio http://www.flickr.com. Si se implementa un microservicio intermediario con ese servicio y se despliega en http://ejemplo.com/flickr/search, el cliente podría asumir erróneamente que el intermediario busca imágenes en el dominio http://ejemplo.com. Por ello, la descripción del microservicio para el intermediario ha de ser complementada con el servicio que realmente se ejecuta para permitir una correcta interpretación del funcionamiento del intermediario.

Definición 8. Una *descripción de microservicio intermediario* es una descripción de microservicio seguida de una URI al servicio que el intermediario ejecuta.

Por tanto, el microservicio intermediario del ejemplo anterior podría describirse como `keyword-filtered multiple picture get http://www.flickr.com/search`. Informalmente, podría entenderse como "http://ejemplo.com/flickr/search permite ejecutar http://www.flickr.com/search como un microservicio `keyword-filtered multiple picture get`".

En resumen, los microservicios intermediarios permiten reutilizar servicios y utilizarlos como microservicios sin modificar la implementación original. Esto permite una transición suave hacia una aplicación basada totalmente en microservicios en un sistema ya desplegado.

V. Utilización de microservicios

La utilización de microservicios en una aplicación web y su descripción permiten una serie de funcionalidades añadidas que se resumen en esta sección.

A. Generación de documentación

La documentación acerca de la utilización de un microservicio se genera automáticamente a partir de la descripción del microservicio. Dicha documentación es la descripción extendida del microservicio, la cual se produce mediante la

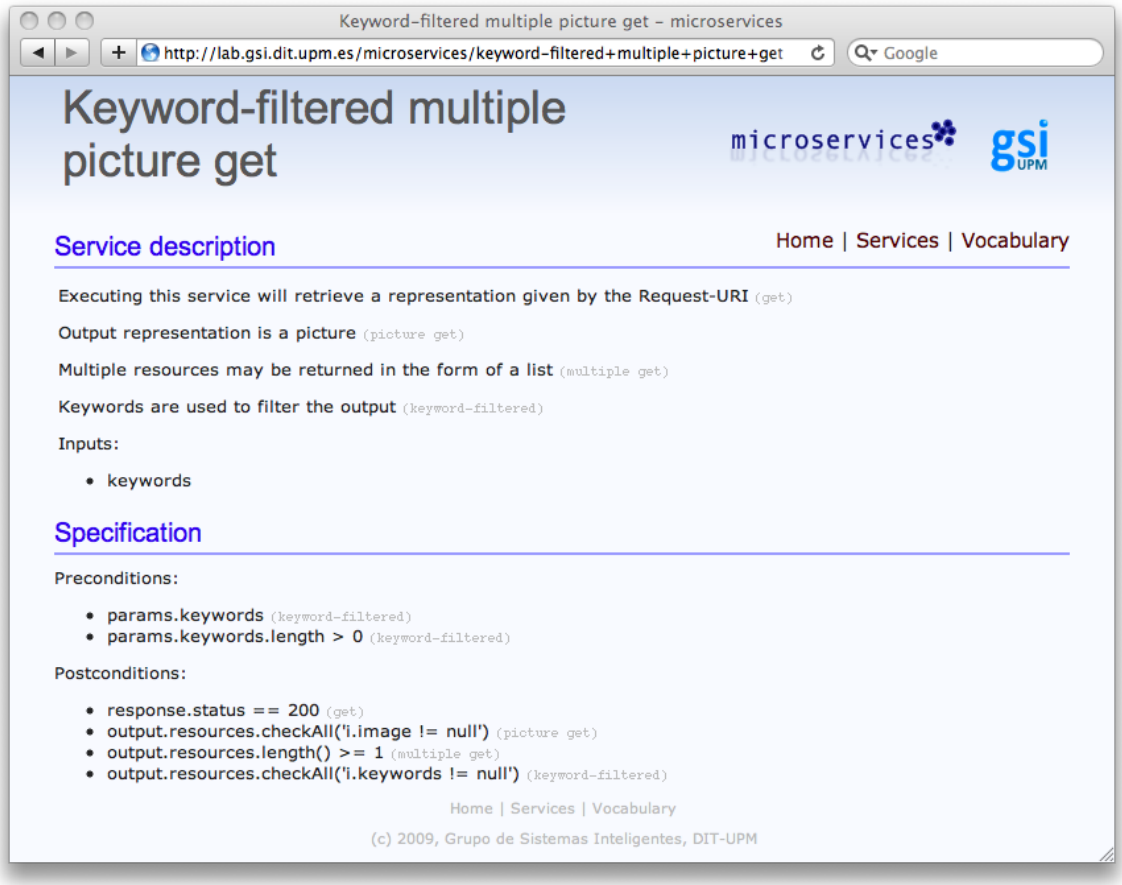

Fig. 1. Ejemplo de descripción extendida de microservicio de búsqueda de imágenes por palabras clave

función de transformación. Dicha descripción consta de dos partes:

- Descripción textual: La descripción textual se obtiene mediante la concatenación de las descripciones de cada uno de los términos.
- Descripción formal: Son las precondiciones y postcondiciones del servicio, que se obtienen mediante la unión de las condiciones que aporta cada término de la descripción. Se expresan mediante un lenguaje de acuerdo a un determinado modelo de sistema, que contiene elementos como petición, respuesta, etc.

La descripción textual es comprensible por un agente humano, mientras que la descripción formal es apta para agentes automáticos y también para humanos en caso de que se desee clarificar detalles del funcionamiento del servicio.

B. Descubrimiento automático

Considerando una aplicación web cuyas interacciones con los recursos son microservicios, pueden incluirse anotaciones en las representaciones de los recursos de la aplicación para que un agente automático pueda conocer las interacciones que se llevan a cabo en dicha aplicación.

Un esquema posible para permitir descubrimiento automático sería incluir la descripción del microservicio en un parámetro de la etiqueta de formulario HTML, como en el siguiente ejemplo en el que se enriquece un formulario de un microservicio de búsqueda:

```
<form
 class="keyword-filtered multiple get"
 action="/search"
 method="get">
    <input name=keywords value="">
    <input name=btnG type=submit value="Search">
</form>
```

Esta modificación permite que un agente rastreador que recorra la web siguiendo hipervínculos pueda conocer la semántica del formulario. De esta manera, utilizando la descripciones de los microservicios un agente automático puede construir una base de datos de microservicios disponibles en la web y realizar filtrados por términos para buscar, por ejemplo, microservicios de búsqueda (`multiple get`), de imágenes (`picture`), etc.

Otra opción más pesada consiste en utilizar la descripción extendida, de forma análoga a la utilización de servicios web semánticos WSMO. Al diseñar un agente inteligente que descubra microservicios a partir de su descripción extendida, se le debería proporcionar la capacidad de procesar las precondiciones y postcondiciones de la descripción para inferir qué microservicios satisfacen sus metas determinadas. Ésta sería la opción más compleja al tener que utilizarse el modelo de sistema empleado en las condiciones de la descripción extendida, en lugar de los términos de la descripción normal.

Consultar la presencia de ciertos términos en la descripción de un microservicio simplifica en gran medida una consulta de encaje de servicios frente al enfoque más formal de procesar las precondiciones y postcondiciones del servicio. En función

de la tarea requerida, puede ser más apropiado utilizar una u otra alternativa.

C. Ejecución automática

Si las interacciones posibles en una aplicación web son microservicios, esto permite ejecutar la aplicación por parte de un agente automático dada la información proporcionada de la semántica de dichas interacciones. A este respecto, al igual que en el caso del descubrimiento, existen dos niveles de semántica utilizables, por la dualidad existente en las descripciones de microservicios.

Nuevamente, la opción más pesada consiste en utilizar la descripción extendida de los microservicios. En este caso, un agente inteligente capaz de ejecutar un microservicio a partir de su descripción extendida procesaría las condiciones de la descripción para ejecutar el servicio de acuerdo a los detalles de formato o parámetros que se muestran en las condiciones.

Utilizando únicamente la descripción normal basada en lista de términos, la implementación de un agente que consuma un microservicio debería conocer de antemano el acuerdo de implementación de servicio que conlleva cada término de la descripción.

Así, por ejemplo, si un término `keyword-filtered` implica que las palabras clave deben estar separadas por espacios, éste hecho podría: (i) incluirse de forma directa en la implementación del agente, de acuerdo al enfoque más ligero que sólo considera las descripciones normales o (ii) inferirse de las postcondiciones en el caso de utilizar la descripción extendida.

La implementación de un consumidor genérico de servicios que infiera cada hecho de las condiciones existentes en una descripción es una tarea compleja que puede no aportar el suficiente valor añadido en muchos casos. Por ello, en general la utilización de la descripción extendida para inferir formatos, parámetros o efectos para la ejecución de un microservicio no debería ser necesaria.

D. Pruebas automáticas

La descripción extendida de un microservicio posibilita la validación de casos de prueba de forma automática. Para:

- un servicio s con precondiciones $pre(s)$ y postcondiciones $pos(s)$,
- un conjunto de elementos del sistema (como parámetros o recursos web) $e_1...e_n$,
- dos estados del sistema s_{bef} (antes de la ejecución del servicio) y s_{aft} (después de la ejecución),

se obtiene que:

$$pre(e_1(s_{bef})...e_n(s_{bef})) \Rightarrow pos(e_1(s_{aft})...e_n(s_{aft})) \quad (1)$$

Por otro lado, se define un caso de prueba como sigue:

Definición 9. Un *caso de prueba* es un conjunto de elementos del sistema en un estado determinado $e_1(s_1)..e_i(s_1)$.

Y, teniendo en cuenta la expresión 1, se verifica que, para un caso de prueba T:

$$\forall e \in T, pre(e(s_{bef})) \Rightarrow pos(e(s_{aft})) \quad (2)$$

Es decir, dado un conjunto de casos de prueba, con la descripción extendida de un microservicio se puede determinar el subconjunto de casos de prueba que ejecutan el microservicio de forma correcta, gracias a las precondiciones. Además, se dispone de los criterios de validación de cada caso de prueba, gracias a las postcondiciones.

Así, la prueba de servicios se reduce a la definición de un conjunto de casos de prueba, entendidos como un simple conjunto de parámetros de entrada y de recursos web, automatizándose la verificación de la validez de dichos casos de prueba y la definición de las condiciones de validación para la salida.

E. Extensibilidad

Como se ha dicho, la idea de microservicios es simplificar las descripciones de servicios. Para ello, el vocabulario de términos con el que trabajar debe mantenerse reducido por simplicidad y para favorecer la reutilización, pero tan amplio como sea posible para permitir la descripción de muchos servicios. A continuación se enumeran algunos principios para mantener un buen vocabulario de términos para su utilización en descripciones de microservicios:

- Un término debe representar una característica de alto nivel de abstracción. Es discutible cuándo una característica es de alto nivel o no. Sin embargo, una posible regla puede ser que un término debería tener el mismo nivel de abstracción que el vocabulario empleado en las aplicaciones web, en las que se manejan términos como blogs, compartición, productos, amigos, etc.
- El nombre de un término debería ser fácilmente comprensible por un humano, y encajar de la mejor forma posible en una frase en lenguaje natural.
- Aparte de los métodos HTTP, un término no debería representar una acción. Los métodos HTTP son el único conjunto de acciones permitidas en la web por la interfaz uniforme del estilo arquitectónico REST. Estas acciones pueden ser extendidas mediante términos que representen roles, como `private`, `keyword-filtered` o `sharing`.
- Un término no debería representar al mismo tiempo un rol y un tipo de datos. Por ejemplo, podrían definirse dos términos como `comment-upload` y `comment-param` en lugar de simplemente `comment`. En el primer caso se amplía gratuitamente el vocabulario, mientras que en el segundo caso se aplica una dependencia con los términos `get` y `post` para definir un único término, reduciendo el vocabulario y simplificando el descubrimiento de microservicios.

VI. Trabajos relacionados

Existen algunos trabajos de investigación que se centran en la descripción de formularios HTML. RDF-Forms [3] busca "incorporar a la Web Semántica capacidades similares a los formularios HTML". Así, propone esquemas que representan a cada uno de los métodos HTTP, anotando sus entradas semánticamente, de forma opuesta al enfoque de microservicios, en el que las entradas quedan anotadas por las postcondiciones del servicio.

OpenSearch [1] es una alternativa para el modelado de servicios de búsqueda para compartir resultados de búsqueda. Al contrario que los microservicios, OpenSearch está enfocado a un tipo muy concreto de servicio, lo que permite especificar con un alto nivel de detalle este tipo de servicios de búsqueda.

Su filosofía es también diferente, dado que en OpenSearch existe un descriptor separado, definido para cada servicio de búsqueda, en el que se especifican características como el tipo de resultados, idioma o codificaciones.

VII. Conclusiones

En este artículo se han revisado enfoques para la automatización de servicios en la web actual. Por regla general, dichas alternativas han sufrido carencias como baja integración con REST, el estilo arquitectónico de Internet, o bien alta complejidad, lo que ha reducido su adopción generalizada.

Como alternativa, se ha propuesto el concepto de microservicios en aplicaciones web para cubrir de forma subóptima la tarea de descripción de servicios. Como se ha visto, la descripción de microservicios está al nivel de sencillez del lenguaje natural, al tiempo que permiten la automatización de tareas como documentación, pruebas, descubrimiento y ejecución, beneficiando a campos como los agentes inteligentes o la combinación de aplicaciones.

Como trabajos futuros se plantean tareas en la línea de facilitar la adopción de este esquema de descripción semántica en aplicaciones web ya existentes. Un primer paso es definir reglas para la clasificación automática de servicios en sus microservicios subóptimos. Para ello, habría que analizar conjuntos de servicios y ver cómo se manifiestan las diversas características que presentan. Por ejemplo, la existencia de viñetas o listas podrían ser indicadores de un formato de salida de lista de recursos, representado por el término `multiple`. Análogamente, un servicio cuya salida contiene las palabras clave incluidas como entrada es susceptible de ser filtrado por palabras clave y contener el término, etc. El resultado sería un conjunto de reglas heurísticas de términos para clasificación automática de servicios en sus microservicios más semejantes. Esta clasificación automática sería complementable con la generación automática de intermediarios.

Agradecimientos

Este trabajo de investigación ha sido financiado por la Comisión Europea bajo el proyecto de I+D ROMULUS (FP7-ICT-2007-1) y por el Gobierno Español bajo el proyecto de I+D Java sobre Ruedas (FIT-350401-2007-8).

Referencias

[1] A9.com, inc. OpenSearch specification. http://www.opensearch.org/Specifications/OpenSearch/1.1, 2005.

[2] B. Adida and M. Birbeck. RDFa Primer - Bridging the Human and Data Webs. http://www.w3.org/TR/xhtml-rdfa-primer/, 2008.

[3] M. Baker. RDF Forms. http://www.markbaker.ca/2003/05/RDF-Forms/, 2005.

[4] W. Consortium. Web Services Architecture. http://www.w3.org/TR/ws-arch/, 2004.

[5] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, 2000.

[6] Grupo de Sistemas Inteligentes. Microservices. http://lab.gsi.dit.upm.es/microservices, 2009.

[7] M. J. Hadley. Web application description language. https://wadl.dev.java.net/wadl20061109.pdf, 2006.

[8] Microformats community. Plain old semantic html (posh). http://microformats.org/wiki/posh, 2007.

[9] Microformats community. Microformats. http://microformats.org/, 2008.

[10] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. METEOR-S Web service Annotation Framework. In *Proceeding of the World Wide Web Conference*, 2004.

[11] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology, Applied Ontology. IOS Press.

[12] A. P. Sheth, K. Gomadam, and J. Lathem. SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups. In *IEEE Computer Society*, 2007.

[13] World Wide Web Consortium. Web Service Semantics - WSDL-S. http://www.w3.org/Submission/WSDL-S/.

[14] World Wide Web Consortium. OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/, 2004.

[15] Wright State University. HTML Microformat for Describing RESTful Web Services and APIs. http://knoesis.wright.edu/research/srl/projects/hRESTs/#hRESTs, 2008.