Hacia el uso de modelos en la selección e integración de Web APIs

Rolando Rodríguez[1], Irene Garrigós[2] y Jose-Norberto Mazón[2]

WaKe research
[1]Dept. of Computer Science, University of Matanzas "Camilo Cienfuegos", Cuba
rolando.rodriguez@umcc.cu
[2]Dpto. Lenguajes y Sistemas Informáticos
Universidad de Alicante, España
{igarrigos,jnmazon}@dlsi.ua.es

Resumen. Este trabajo constituye una propuesta de ingeniería dirigida por modelos para dotar a desarrolladores Web de mecanismos para tomar decisiones informadas a la hora de seleccionar e integrar Web APIs. El objetivo de este trabajo es la definición de características cualitativas que intervienen en la integración de las Web APIs, y la definición de medidas basadas en análisis estructural que permitan la comparación entre de las mismas a través de sus operaciones. La propuesta emplea técnicas de metamodelado para lograr homogeneizar la representación de las Web APIs y así poder establecer mediciones independientemente de cómo se hayan implementado. Con el fin de realizar estas mediciones se aplican técnicas de análisis estructural, obteniendo criterios en los que se deben apoyar los desarrolladores para seleccionar las mejores combinaciones de Web APIs.

Palabras clave: datos abiertos, análisis estructural, metamodelo, ingeniería web

1 Introducción

Tras la creciente proliferación de los Servicios Web se han reunido esfuerzos por construir directorios de internet que aglutinan gran cantidad de referencias a Web APIs con el objetivo de facilitar y promover la reutilización de sistemas. El ejemplo paradigmático es ProgrammableWeb[1]. La forma en que se organizan las referencias y los mecanismos de búsqueda de este directorio no potencian la toma de decisiones satisfactorias cuando se trata de seleccionar Web APIs que sean fácilmente integrables entre sí. Esta deficiencia se debe a que las sugerencias que se obtienen como resultado de una búsqueda pueden filtrarse u ordenarse simplemente por atributos de las Web APIs o por criterios de popularidad. No obstante, ninguno de estos elementos permite determinar el esfuerzo de integración. La integración tiene lugar cuando existe una afinidad o compatibilidad entre operaciones de Web APIs. Para estudiar la integración es necesario analizar los parámetros con los que funcionan estas operaciones, su estructura, su tipo de dato y su significado. Comparando dos operaciones en base a estos elementos se puede llegar a conclusiones sobre la dificultad que puede aparecer en el momento de integrarlas. Esta dificultad radica en dos factores fundamentales: (i) la sustitución y (ii) el acoplamiento. Si se pretende erradicar una operación de un Mashup es útil sustituirla por otra que sea similarmente funcional y aporte mejores prestaciones. Por otro lado, si ya se tienen integradas varias operaciones y se necesita una nueva funcionalidad, es posible encontrarla en otra operación que acople bien en la aplicación. Y en caso de no aparecer una candidata que se acople directamente, es posible que una tercera operación pueda servir de intermediaria para acoplar a otras dos. Se habla de acoplamiento aludiendo a la capacidad que tiene una operación de proveer datos con los que otra pueda funcionar. En la medida en que menos transformaciones y menos volumen de codificación sean necesarios, menor se hace el esfuerzo de integración. En este trabajo se propone una

[1] **www.programmableweb.com**. Directorio líder de Web APIs y mashups de Internet

adfa, p. 1, 2011.

representación homogénea de las Web APIs por medio de un metamodelo y se definen medidas que permiten cuantificar la compatibilidad de sus operaciones en escenarios donde se deba realizar la sustitución o el acoplamiento de las mismas. Para el caso del acoplamiento por medio de operaciones intermediarias se propone el empleo de técnicas del -análisis estructural-, las cuales van a permitir determinar criterios para tomar decisiones informadas sobre la integración de las Web APIs desde una perspectiva más global.

2 Trabajo relacionado

Para el apoyo a la toma de decisión en la selección de Web APIs pueden emplearse técnicas basadas en criterios de calidad, precisamente un trabajo interesante sobre esta base se propone en [1]. La propuesta calcula diversas medidas de calidad, centrándose en la precisión de los datos devueltos, la vigencia y la fiabilidad de las Web APIs. Sin embargo consideran la similitud de forma global y no se estudia esta medida en las operaciones, por lo que la selección se hace siguiendo criterios demasiado generales. En dicha propuesta no se hace hincapié en la heterogeneidad asumiendo únicamente como Web APIs a los servicios basados en *REST*[2]. Otro trabajo interesante es [2] que estudia varios contextos de uso de las Web APIs con el fin de poder realizar una selección de manera personalizada. Esta última se basa en la aplicación de teoría de grafos para capturar las relaciones entre Web APIs y sus usuarios, teniendo en cuenta las preferencias y la satisfacción de los usuarios. El diseño, implementación y evaluación de Mashups personalizados se describe también en [3] donde se propone por medio de la creación de una base de datos con información referente a cómo se han usado ciertas Web APIs y su aplicación para la creación de Mashups con información geográfica. En [4] se propone el desarrollo de Mashups para el análisis de datos a través de la definición de patrones de diseño basados en entornos similares a hojas de cálculo. En cuanto al desarrollo de un modelo de programación que facilite el desarrollo de Mashups, en [5] se define Swashup DSL, que posibilita la integración de servicios Web en aplicaciones Web. Por otra parte, en [6] se define un modelo formal para el desarrollo de Mashups cuyo componente básico es el denominado "mashlet". Un "mashlet" puede tener varias funcionalidades: consultar fuentes de datos, importar otros "mashlets", usar servicios Web externos, etc. Este modelo facilita la composición dinámica de mashlets, su interacción y reutilización con el fin de capturar el comportamiento de un Mashup. Otro trabajo que sigue una propuesta dirigida por modelos es [7] donde se define un metamodelo para el desarrollo de Mashups. Desafortunadamente no proveen mecanismos para la selección de Web APIs. Finalmente, en [8] se define un modelo llamado Mashlight centrado no sólo en la selección y composición de Web APIs para el desarrollo de Mashups para la Web sino también en la composición de "widgets" para aplicaciones móviles. Todas estas propuestas indican la importancia que tiene el desarrollo de Mashups en el campo de la investigación en ingeniería del software.

3 Uso de modelos para la selección e integración de Web APIs

Durante años, la ingeniería dirigida por modelos [9,10] ha proveído a los desarrolladores de una forma de estandarizar y documentar de una forma visual cada etapa de desarrollo del software (Ejemplo: UML3 en ingeniería de software y ER4 en el modelado de bases de datos). Utilizando modelos de Web APIs se conseguiría una forma homogénea y formal de representarlas, así como el nivel de automatización de todas las fases de desarrollo de Mashup se puede elevar [11], y se puede afrontar desde una perspectiva más formal las cuestiones de interoperabilidad [12].

---

2 http://es.wikipedia.org/wiki/REST

[3] UML: Lenguaje Unificado de Modelado. http://es.wikipedia.org/wiki/UML

[4] ER: Modelo Entidad-Relación. http://es.wikipedia.org/wiki/Modelo_entidad-relación

3.1 Metamodelo

Esta propuesta formaliza las Web APIs a través de un metamodelo que contiene los componentes que se describen a continuación y se muestran en la Figura 1.

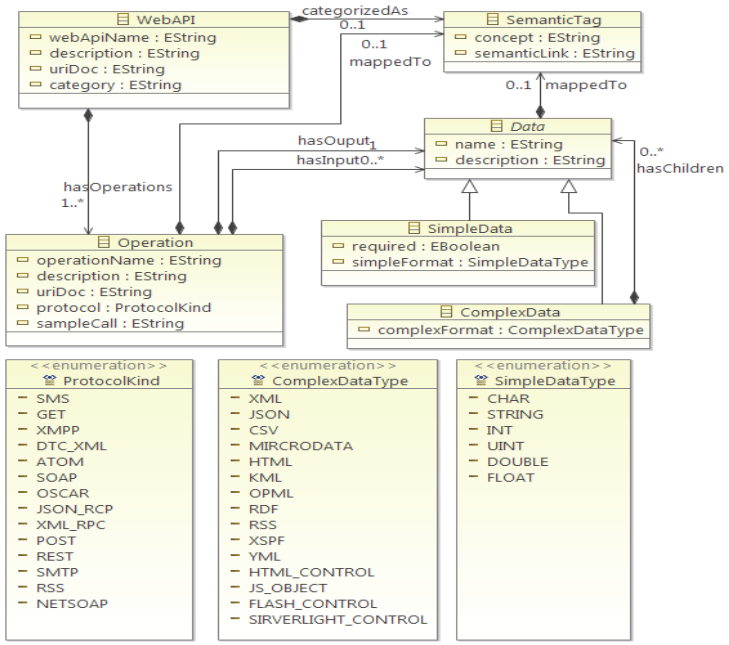

Figura 1 Metamodelo de Web API

- **Metaclase *Web API***: Clase contenedora principal, en la cual se define el *WebAPIName* (nombre de la Web API), *description*, *uriDoc* (dirección URL de la documentación oficial), *category* (categoría general que engloba el dominio o contexto), *hasOperation* (representa la contenencia de una o varias operaciones).
- **Metaclase *Operation***: Acciones o rutinas que implementa una Web API. Cada *Operation* tiene un *operationName* y *description* (atributos descriptivos). Cada operación tiene tambien una *uriDoc*. Intercambian sus datos a través de un protocolo por ello el atributo *protocol* que es una de las opciones que están en el enumerador *ProtocolKind*. Una *operation* puede tener o no datos de entrada, y siempre contiene al menos una salida. Tanto los datos de entrada como de salida pueden ser desde datos simples como texto, números, hasta tan complejos como una representación de un objeto en formato XML, JSON, etc., o incluso un control de usuario.
- **Metaclase *Data***: Clase abstracta que representa los datos de entrada o salida de una operación. Sus atributos *name* y *description* son puramente descriptivos. De esta clase heredan *SimpleData* y *ComplexData*.
- **Metaclase *SimpleData***: Datos simples, valores primitivos que tienen un único significado, por ejemplo: datos de tipo *string*, *double*, *char*, *int* etc. El atributo *required* toma valores de verdadero o falso y se utiliza para representar parámetros de entrada obligatorios en la llamada de una operación. En un parámetro de salida este atributo siempre será verdadero. El atributo *simpleFormat* representa el tipo de formato de este dato, cuyo valor es una opción del enumerador *SimpleDataType*.
- **Metaclase *ComplexData***: Datos complejos de entrada o salida de operaciones. Se consideran complejos aquellos datos que puedan contener datos simples o complejos por ejemplo un esque-

ma XML, un objeto, un control de usuario, etc. El atributo *complexFormat* representa los tipos de formato del dato y toma un valor entre las opciones del enumerador *ComplexDataType*.

- **Metaclase *SemanticTag***: Representa etiquetas semánticas en el metamodelo (correspondencia entre un elemento del metamodelo y un recurso que lo represente semánticamente). *Concept (*literal que enuncia el significado o concepto), *semanticLink (*identificador del recurso donde está la definición el concepto). Cada operación puede estar mapeada o no a un único *SemanticTag* mediante la relación *mappedTo*, cada *Data* puede estar mapeado o no a un único *SemanticTag* mediante la relación *mappedTo*. La Web API puede a su vez estar enlazada o no a un único *SemanticTag* mediante la relación *categorizedAs*.

3.2 Medidas de Integración de Web APIs

Para medir el esfuerzo de integración entre operaciones de Web APIs fue necesario definir dos medidas que son aplicables a escenarios de sustitución o de acoplamiento. Estas están enfocadas a cuantificar la correspondencia entre los significados de los datos con los que trabajan las operaciones. Para ello fue necesario hacer uso del coeficiente de afinidad definido en [12]. Este coeficiente se comporta de forma unidireccional y asimétrica por lo que no se puede pensar en operaciones matemáticas conmutativas. Este comportamiento va a estar presente también en las medidas **Similitud Funcional** y **Acoplamiento Funcional** que se proponen en este trabajo. Para expresar esta direccionalidad se emplean los términos origen y destino.

3.2.1 Similitud Funcional

Medida que va a posibilitar cuantificar la capacidad que tiene una operación de ser sustituida por otra. Se manifiesta cuando dos operaciones tienen propósitos afines (aplicando el coeficiente de afinidad definido en [12]) y sus datos de entrada y salida son afines respectivamente. Esta medida se calcula como la sumatoria de tres componentes. El primero es la afinidad que existe entre los datos de entrada de la operación destino y los de la operación origen. El segundo corresponde con la afinidad de los datos de salida de ambas operaciones. Finalmente, el tercero representa la afinidad entre los propósitos de ambas operaciones (origen y destino). La sumatoria se normaliza con el fin de que el resultado tome valores en el rango [0,1].

3.2.2 Acoplamiento Funcional

Medida que posibilita cuantificar la capacidad que tiene la operación origen de proveer datos con los que la operación destino pueda funcionar correctamente. Se calcula sumando las correspondencias entre los datos de salida de la operación origen y los de entrada de la operación destino y normalizando el resultado para que se mantenga también en el rango [0,1]. Se toman en cuenta solamente los parámetros de entrada de la operación destino que son requeridos. El acoplamiento puede manifestarse incluso entre operaciones de la misma Web API. Esto es algo que se toma muy en cuenta a la hora de sugerir favorablemente una operación u otra cuando existe empates de esta medida. Se consideró producente darle relevancia a los pares de operaciones pertenecientes a la misma Web API ante pares de operaciones de Web APIs diferentes pero para ello es necesario representar el repositorio como una estructura más compleja. Esto se desarrolla en la próxima sección.

3.3 Más allá de las medidas individuales de integración: análisis estructural

La definición de las medidas descritas en 3.2.1y 3.2.2 se pueden emplear para estudiar enfoques individuales de integración. Para poder analizar todo el repositorio de modo que se pueda sugerir y

guiar la selección desde una perspectiva global, se propone la representación del mismo en forma de un grafo donde cada operación es un nodo y sus arcos son los valores calculados de similitud y acoplamiento funcionales para cada par de operaciones. La representación física de la estructura se realiza empleando ficheros que contienen matrices de adyacencia individuales para cada medida. La matriz de acoplamientos se construye en dos ejecuciones, una para calcular el acoplamiento individual entre todos los pares de operaciones de la red y la otra para favorecer positivamente a los enlaces de acoplamiento entre operación de la misma Web API cuando existen empates con operaciones de Web APIs ajenas. Una vez construidas estas matrices se aplican técnicas del análisis estructural [13, 14] que van a devenir en nuevas matrices y vectores con información relevante del comportamiento de la similitud y el acoplamiento a gran escala en todo el repositorio. La aplicación de esta disciplina permite introducir nuevos indicadores tales como: la -Centralidad de Grado-, la -Distancia Geodésica- y la -Centralidad de Intermediación-; los cuales van a permitir determinar enlaces de acoplamiento indirectos por mediación de operaciones intermediarias, operaciones que potencialmente se encuentran en la mayor cantidad de secuencias de acoplamiento calculadas entre todas las operaciones no acoplables directamente en la red, entre otros criterios.

4 Implementación inicial

La implementación de la propuesta se realizó en el entorno *Eclipse*[5] que permitió la definición del metamodelo, la creación de los modelos de Web APIs, la implementación de las clases necesarias para la representación del grafo y el cálculo de los indicadores. Se empleó como recurso semántico *WordNet*[6] pero la implementación está basada en patrones de diseño que permiten fácilmente adaptar a posteriori otros recursos más potentes como por ejemplo *DBPedia*[7]. También se hizo uso de UCINET[8] que es otro recurso externo a *Eclipse* mediante el cual a través de su interfaz de líneas de comando se puede automatizar el cálculo de los indicadores de redes. Para construir un modelo de Web API, se pueden emplear técnicas automatizadas, supervisadas o de forma manual. En esta investigación no se hace hincapié en la automatización [15] de su proceso de creación, por lo que los modelos usados se han construido de forma manual, siendo una pequeña muestra que sirvió para evaluar la propuesta.

5 Conclusiones y trabajo futuro

Se puede concluir a modo general que en esta propuesta, la integración de las Web APIs se estudia más afondo al nivel de las operaciones considerando estas como la estructura básica. Para poder cuantificar la forma en que las operaciones de Web APIs se pueden combinar se han definido las medidas de similitud funcional, acoplamiento funcional. Medidas que permiten calcular la facilidad de acoplar un par individual de operaciones y también establecer un criterio de similitud o equivalencia funcional entre ellas. Se ha considerado más provechoso sugerir operaciones de la misma Web APIs; siempre que sea posible acoplarlas; antes de mezclar operaciones de Web APIs diferentes; consideración que se tiene en cuenta para la definición e implementación de las matrices de acoplamiento propuestas. Además se ha estudiado cómo, aplicando medidas e indicadores del análi-

[5] www.eclipse.org
[6] **WordNet**: Base de datos léxica con relaciones semánticas (http://wordnet.princeton.edu/)
[7] http://www.dbpedia.com/
[8] https://sites.google.com/site/ucinetsoftware/ Es un analizador estructural que provee interfaces para el manejo de matrices de redes complejas tanto interfaces graficas como líneas de comando.

sis estructural de redes, se puede obtener información sobre las operaciones intermediarias que potencialmente pueden posibilitar que otras dos operaciones se puedan integrar indirectamente.

Como trabajo futuro se pretende aplicar otras relaciones semánticas que representen similitud e implicación entre conceptos para mejorar el coeficiente de afinidad propuesto por Bianchini en [12]. Por ejemplo, se estudiará cómo utilizar DBPedia para etiquetar semánticamente las componentes de las Web APIs. Otro trabajo futuro es considerar, en relaciones de implicación, el nivel de granularidad entre significados, como una variable que puede influir luego en los valores de similitud y acoplamiento. Por otra parte, el metamodelo se debe ampliar para incluir elementos que consideren el propio Mashup como un conjunto integrado de Web APIs, así como el contexto de funcionamiento de las Web APIs, la compañía o usuario que se la ha desarrollado para poderlas agrupar, y algunas restricciones tecnológicas por ejemplo:

- Cuando una operación de una Web API provee funciones que son específicas para una zona geográfica no tiene sentido integrarla con otra que también tenga restricciones para de funcionamiento para otra zona geográfica. Tampoco tiene sentido compararlas porque no se puede sustituir una por otra.
- Cuando una operación de una Web API tiene una restricción tecnológica como que solo puede obtener sus datos de entrada del lado servidor, no puede integrarse con Web APIs cuyo funcionamiento dependa del lado cliente.

También se tiene la proyección de aplicar otros indicadores estructurales para obtener más conclusiones respecto al repositorio. Definir mecanismos para obtener criterios de selección en contextos y escenarios más complejos. Por ejemplo, la determinación de *CLIQUES* (Grupos de estructuras en el grafo que sean fuertemente conexas) puede interpretarse como grupo de operaciones de Web APIs que son fuertemente acoplables. Se pueden obtener información relevante de indicadores sobre el repositorio en general como la densidad, la integralidad o la accesibilidad.

Referencias

1. Marco Comerio, Matteo Palmonari, Flavio De Paoli, Carlo Batini Luca Panziera, "Quality-driven Extraction, Fusion and Matchmaking of Semantic Web API Descriptions.," J. Web Eng., vol. 11, no. 3, pp. 247-268, 2012.
2. Milan Dojchinovski, Kuchar Jaroslav, Vitvar Tomas, and Zaremba Maciej, "Personalised Graph-based Selection of Web APIs".
3. Vania Dimitrova, Karim Djemame Minh Dang Thang, "Personalised Mashups: Opportunities and Challenges for User Modelling," User Modeling, pp. 415-419, 2007.
4. Boualem Benatallah, Julien Vayssière, Régis Saint-Paul, Fabio Casati Woralak Kongdenfha, "Rapid development of spreadsheet-based web mashups," WWW, pp. 851-860, 2009.
5. Hernán Wilkinson, Nirmit Desai, Stefan Tai E. Michael Maximilien, "Domain-Specific Language for Web APIs and Services Mashups," ICSOC, pp. 13-26, 2007.
6. Ohad Greenshpan, Tova Milo Serge Abiteboul, "Modeling the mashup space," WIDM, pp. 87-94, 2008.
7. Vincent Tietz, Jan Reimann, Christian Liebing, Michèl Pohle, Klaus Meißner Stefan Pietschmann, "A metamodel for context-aware component-based mashup applications," iiWAS, pp. 413-420, 2010.
8. Sam Guinea Luciano Baresi, "Consumer Mashups with Mashlight," ServiceWave, pp. 112-123, 2010.
9. Juan Carlos Molina Oscar Pastor, "Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modeling," 2007.
10. Kendall Scott, Axel Uhl, Dirk Weise Stephen Mellor, "Principles of Model Driven Architecture," MDA Distilled, 2004.
11. R Baumgartner, G Gottlob, and M Herzog, "Scalable web data extraction for online," vol. 2, 2009.

12. Valeria De Antonellis, and Michele Melchiori Devis Bianchini, "Semantics-Enabled Web API Organization and Recommendation," p. 10, 2012.
13. L Freeman, "Centrality in social networks: I. Conceptual classification".: Social Networks, 1979.
14. L Freeman, The gatekeeper, pair-dependency, and structural centrality.: Quality and Quantity, 1980.
15. Rolando Rodríguez Ortega et al., "Extracting models from Web API documentation," LNCS vol. 7703, 2012.