# MICROSERVICES: LIGHTWEIGHT SERVICE DESCRIPTIONS FOR REST ARCHITECTURAL STYLE

José Ignacio Fernández-Villamor, Carlos Á. Iglesias, Mercedes Garijo
*Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid*
*{jifv, cif, mga}@dit.upm.es*

Keywords: REST, service description, semantic web, mashups, web applications

Abstract: Current web has a vast number of applications available that offer users a wide domain of services. Most services, however, cannot be machine processed, which limits service composition for application and mashup development. Research on Semantic Web Services contributes to the improvement of interoperability and composition of applications and services. Many approaches cover service description by following paradigms such as Web Services and REST architectural style, allowing describing any kind of service for its use by an automatic agent, but sometimes using these solutions can be a time-consuming task.
This paper introduces Microservices, a lightweight service classification framework for REST architectural style. Microservices do not attempt to describe every possible service, but to provide a way to describe a set of services in a simple way. Microservice descriptions consist of a set of terms that represent service features. After describing features semantically, microservices framework allows generating detailed service descriptions, which allows reusing common feature descriptions across different services. A use case that adapts heterogeneous search services to produce a standard interface using microservices is described.

## 1 INTRODUCTION

The web has an increasing number of applications and services that cover different domains and fields. Users can enjoy a wide range of applications, from e-commerce to blogging, media or social networking. The possibilities of the current web are only limited by the interoperability between applications. Internet versatility would increase if applications could be arbitrarily composed and automatically executed to fulfil a user's goal. This approach of building a programmable web has led to research in several areas, such as semantic web services (Zhou et al., 2006) or mashups (Yu et al., 2008). Many approaches have been successful from a research point of view without having reached wide adoption. The effort of describing services in web applications can be too big and too often web developers do not perform that task.

This paper introduces microservices, a framework that attempts to simplify service description, and its application on a use case. With microservices framework, service descriptions can be generated out of simple service feature lists, enabling services in web applications to be consumed by automatic agents.

This paper is organized as follows. First, a review of previous related approaches is done and the background of the problem is summarized. Second, our approach to service description is described. Third, a use case that employs microservices to adapt heterogeneous search services is outlined. Finally, the main conclusions of this research work are summarized.

## 2 BACKGROUND

Current web has applications with plenty of services available, which are of many different kinds. Approaches to describe services semantically allow automatic agents to execute and compose services automatically. Some approaches follow the Web Services Architecture, such as OWL-S (World Wide Web Consortium, 2004), WSMO (Roman et al., 2005), METEOR-S (Patil et al., 2004) or WSDL-S (World Wide Web Consortium, 2005). To favour the integration with Internet's architecture, RESTful services started to be employed in web applications, which caused RESTful alternatives to semantic web services to appear. SA-REST (Sheth et al., 2007) or hRESTs (Wright State University, 2008) are approaches that provide languages to describe RESTful APIs specifications. Similarly, Web Application Description Language (Hadley, 2006) proposes describing RESTful APIS by defining a WADL, XML-based, file.

All these approaches provide means to describe every kind of service and process the respective descriptions to automate tasks such as execution or composition. They are abstract and flexible enough to allow describing every possible service. Whenever a new service is deployed, a description has to be built in order to make the service available for automatic agents to process it. Usually, this implies describing service's inputs and outputs semantically, as well as defining the precise operation the service performs, which can be a time-consuming task that is sometimes not carried out.

## 3 MICROSERVICES

Microservices framework attempts to simplify the process of defining service descriptions to push automatic service consumption in the semantic web. In this framework, the description task attempts to be improved by enabling reusability accross service descriptions.

A microservice description is a list of terms. Each term represents a feature that the service has. Examples of features can be "*the service performs a retrieval operation*", "*the service requires user authentication*", "*the service performs a storage operation*", "*the service has an image as input*", or "*the service outputs a set of resources*". By describing features semantically, feature descriptions can be combined to produce a semantic service description. This allows reusing feature descriptions in services that are different but have one or more features in common. This feature-based approach is shown in figure 1, and has four elements of interest:

- A microservice description, which consists of a list of terms. Each term represents a feature that the service has.
- An extended service description, which is defined as a textual description and a set of preconditions and postconditions that define the operation that is performed by the service.
- A transformation function that transforms a microservice description into an extended service description.
- A set of term definitions that define the semantics of terms and thus describe service features.

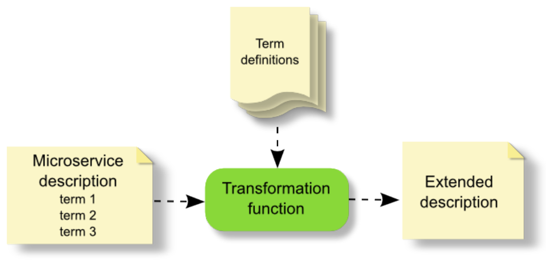


Figure 1: Microservice description framework

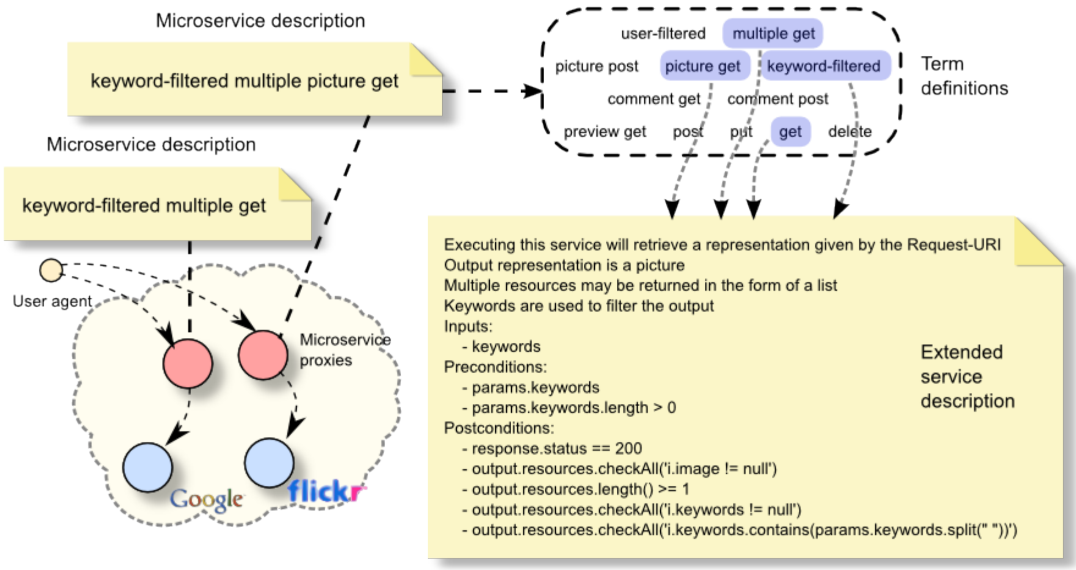
An example of microservice description is `keyword-filtered multiple picture get`, which describes a search service of pictures that are filtered by keywords. Four terms are used in order to define the microservice: `keyword-filtered`, `multiple`, `picture`, and `get`. Because of following REST architectural style, at least a term that represents the underlying HTTP method used has to be included in the description.

An extended service description is built by combining the appropriate term definitions. Definitions can be tied to more than one term. For example, there can be a definition for `get`, a definition for `picture`, but also a definition for altogether `picture` and `get`.

The need for definitions that involve sets of terms can be easily noticed. Term `picture` should set a postcondition (i.e. "*outputting a picture*") when used with term `get`, but should set a precondition (i.e. "*a picture has to be provided as parameter*") when used with term `post`. This can be achieved by setting definitions for `picture` and `get` and for `picture` and `post`.

In this case, the extended microservice description for the previous example is built by combining the definition for `keyword-filtered`, the definition for `multiple` and `get`, the definition for `picture` and `get`, and the definition for `get` by grouping their respective preconditions, postconditions and textual descriptions. This service's extended description can be seen in figure 2.

As a feature-based description framework, the microservice approach has several advantages over manually defining a service description: (i) it allows the reuse of feature descriptions among different services, and (ii) it reduces the description task to selecting a set of features that describes the considered service, given a vocabulary of terms.



Executing this service will retrieve a representation given by the Request-URI
Output representation is a picture
Multiple resources may be returned in the form of a list
Keywords are used to filter the output
Inputs:
- keywords
Preconditions:
- params.keywords
- params.keywords.length > 0
Postconditions:
- response.status == 200
- output.resources.checkAll('i.image != null')
- output.resources.length() >= 1
- output.resources.checkAll('i.keywords != null')
- output.resources.checkAll('i.keywords.contains(params.keywords.split(" "))')

Figure 2: Sample use case scenario

## 4 USE CASE: DESCRIBING SEARCH SERVICES

An environment for the creation of microservices through the definition of microservice proxies to other external services has been developed[1]. This environment allows wrapping an already existing service into a proxy that exposes an interface that can be described with a microservice description. Microservice proxies are defined by specifying a preadapter, which adapts service inputs, and a postadapter, which adapts the output format. This environment is used in this use case to wrap Google and Flickr search services.

In order to describe the proxies to these services, terms `keyword-filtered`, `picture`, `multiple`, and `get`, and their associated term definitions as shown in table 1, are defined.

The result of wrapping these search services as microservice proxies, as shown in figure 2, is:

- The described microservices share most of their terms in their descriptions, which allows reutilization of the semantic descriptions.
- Both services share the same interface. Sample searches can be tried at Google's proxy[2] and Flickr's proxy[3]. All of them accept a set of keywords in input `keywords` and have the same output format. As Flickr's proxy searches pictures, it additionally includes an `image` field in its results.
- Services are advertised in an HTML form at a URI. This HTML form allows the execution of the service by a human user.
- Automated validation. Preconditions and postconditions are checked when executing a service, which allows checking the service's correct execution.
- Automated discovery. Microservice descriptions are advertised as Linked Data in the HTML form, which allows processing by an automatic agent.

[1] `http://lab.gsi.dit.upm.es/microservices`

[2] `http://lab.gsi.dit.upm.es/microservices/http://www.google.com/search?keywords=test`

[3] `http://lab.gsi.dit.upm.es/microservices/http://www.flickr.com/search?keywords=test`

## 5 RELATED WORK

Some ongoing research works focus on HTML form description. RDForms (Baker, 2005) attempt to "add to the Semantic Web capabilities similar to HTML forms". Schemas for indexable, container and settable operations are defined, which represent HTTP GET, POST, and PUT methods, respectively.

OpenSearch (A9.com, inc., 2005) is an approach to modelling search services. It is aimed at a very specific kind of service, which allows building fine-grained specifications.

However, these and other already mentioned alternatives such as WSMO, SA-REST or WADL are

Table 1: Term definitions for search services

| | |
|---|---|
| Terms | picture, get |
| Description | Output representation is a picture |
| Postconditions | output.resources.checkAll('i.image != null') |
| Terms | keyword-filtered |
| Description | Keywords are used to filter the output |
| Preconditions | params.keywords |
| | params.keywords.length > 0 |
| Postconditions | output.resources.checkAll('i.keywords != null') |
| | output.resources.checkAll('i.keywords.contains(params.keywords.split(" "))') |
| Terms | get |
| Description | Executing this service will retrieve a representation given by the Request-URI |
| Postconditions | response.status == 200 |
| Terms | multiple, get |
| Description | Multiple resources may be returned in the form of a list |
| Postconditions | output.resources.length() >= 1 |

heavy-weight approaches that provide means to describe every possible service from scratch. On the contrary, microservices framework attempts to allow reusing other service's descriptions.

## 6 CONCLUSIONS

In this paper, current service automation approaches have been reviewed. In general, these approaches are focused on providing flexible ways to describe every possible service. However, this flexibility occasionally makes service description a time-consuming task.

In this paper, we have introduced microservices as an alternative to simplify service description. Microservice descriptions are simple and allow automating various tasks in order to push service automation in the semantic web. A use case that adapts a set of heterogeneous search services has been described, offering a uniform interface, discoverability, and automating documentation and validation.

Future work involves analyzing other case studies and considering automatic wrapper induction to simplify the task of building microservice proxies.

## ACKNOWLEDGEMENTS

This research project is funded by the European Commission under the R&D project ROMULUS (FP7-ICT-2007-1) and by the Spanish Government under the R&D project Java sobre Ruedas (FIT-350401-2007-8).

## REFERENCES

A9.com, inc. (2005). OpenSearch specification. http://www.opensearch.org/Specifications/OpenSearch/1.1.

Baker, M. (2005). RDF Forms. http://www.markbaker.ca/2003/05/RDF-Forms/.

Hadley, M. J. (2006). Web application description language. https://wadl.dev.java.net/wadl20061109.pdf.

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

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology, Applied Ontology. IOS Press.

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

World Wide Web Consortium (2004). OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/.

World Wide Web Consortium (2005). Web Service Semantics - WSDL-S. http://www.w3.org/Submission/WSDL-S/.

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

Yu, J., Benatallah, B., Casati, F., and Daniel, F. (2008). Understanding mashup development. *IEEE Internet Computing*, 12(5):44–52.

Zhou, J., Koivisto, J.-P., and Niemela, E. (2006). A survey on semantic web services and a case study. In *Computer Supported Cooperative Work in Design, 2006. CSCWD '06. 10th International Conference on*, pages 1–7.