# A Lightweight Model for Publishing and Sharing Linked Web APIs*

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Dept. of Information Engineering – University of Brescia
Via Branze, 38 – 25123 Brescia, Italy
{bianchin|deantone|melchior}@ing.unibs.it

**Abstract.** The web of Linked Data has been proposed in the last years in order to create a global data graph, that spans data sources, connected by RDF links, and enables the discovery of new resources. Recently, Web APIs have been more and more used to access documents and metadata from the web of Linked Data and to easily compose new applications called web mashups. In this paper, we describe a lightweight model for publishing and sharing Web APIs and mashups on the web of Linked Data. The model has been designed (a) to support providers who publish new Web APIs used for accessing the web of Linked Data, (b) to support the web designer who aims at exploring and selecting available Web APIs for building or maintaining a web mashup, and (c) to make the mashup itself available on top of the web of Linked Data. The functional architecture of a platform based on the model is also introduced.

## 1 Introduction

The web of Linked Data can be seen as a global database, where resources are identified through URIs, are semantically described and globally connected through RDF links [3]. Recently, Web APIs have been more and more used to access documents and metadata from the web of Linked Data and to easily compose new applications called web mashups. In this paper, we describe a lightweight model for publishing and sharing Web APIs and mashups on the web of Linked Data. A lightweight model for Linked Web APIs built on top of the web of Linked Data has been proposed in [7], where the aim is to support users in the semantic annotation of Web APIs by leveraging on the huge amount of linked web resources. In particular, the SWEET tool [5] has been designed to assist users in annotating HTML descriptions of Web APIs, relying on solutions like Watson [4] for browsing the web of Linked Data, so that it is possible to identify suitable vocabularies (e.g., eCl@ass, Good Relations and FOAF) and use them for the annotation. Our goal in this paper is to extend the model described in [7] to improve the selection from-the-shelf of available Web APIs and to better support their aggregationfor building new web mashups.

* Extended Abstract

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

The motivations of our effort rely on a lack of support for browsing and selecting the right Web APIs from a highly populated repository of third party components. As a motivating example, consider a web designer who wants to make available for his friends a new web application for finding informations, trailers and ratings on Michelangelo Antonioni's movies (see Figure 1, where a screenshot of `MovieGram`[1], a similar application, is shown). To this aim, the web designer has to find and wire Web APIs that provide information about movies (e.g., `Rottentomatoes.com`) and Web APIs to show videos and trailers (e.g., `YouTube`), instead of designing the application from scratch (that could be a non-trivial and time consuming task). On the other hand, browsing and selection of the right functionalities from a huge list of heterogeneous Web APIs is not an easy task if manually performed. To mention the well-known ProgrammableWeb API repository, which registers over 5,600 Web APIs (a number that is continuously growing), the web designer may perform a keyword-based search and find up to 32 Web APIs related to the keyword `movie` and up to 141 APIs for displaying videos, each of them presents several invocable operations with I/Os that must be properly wired (for instance, in the application in Figure 1, videos related to the Antonioni's Zabriskie Point movie are shown on the `YouTube` Web API). Therefore, the web designer must be properly supported in the selection and aggregation of available Web APIs.

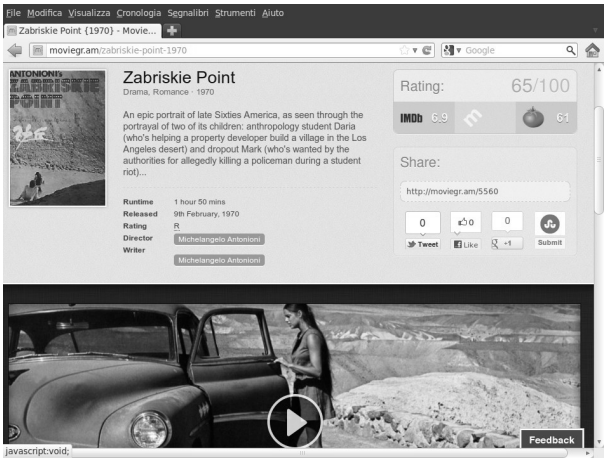


**Fig. 1.** An example of mashup that mixes up few APIs out of an highly populated repository.

To this aim, we propose the model described in this paper, where: (1) HTTP URIs are used to identify Web APIs, (2) useful (RDF) information are provided on the Web API description when someone looks up a Web API through its URI,

[1] `http://moviegr.am/`.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

and (3) proper links are set to relate Web APIs to each other, thus enabling easy development and sharing of new web applications.

The paper is organized as follows. In Section 2 the lightweight model is described. Section 3 introduces the I-MASH platform, an on-going project, based on the model, that is being designed (a) to support providers who publish new Web APIs used for accessing the web of Linked Data, (b) to support the web designer who aims at exploring and selecting available Web APIs for building or maintaining a web mashup, and (c) to make the mashup itself available on top of the web of Linked Data. Finally, Section 4 closes the paper.

## 2 Linked Web API model

Following the lesson learnt by the iServe platform [7], we propose the conceptual model shown in Figure 2. The model aims at connecting the contents of the ProgrammableWeb repository of Web APIs with the web of Linked Data to fasten Web API selection and aggregation, going beyond keyword-based searching facilities offered by the repository, given the huge number of Web APIs among which the web designer must choose with few information to support him. We considered the ProgrammableWeb repository since, to the best of our knowledge, it is the most complete collection of Web APIs and user-defined web mashups.

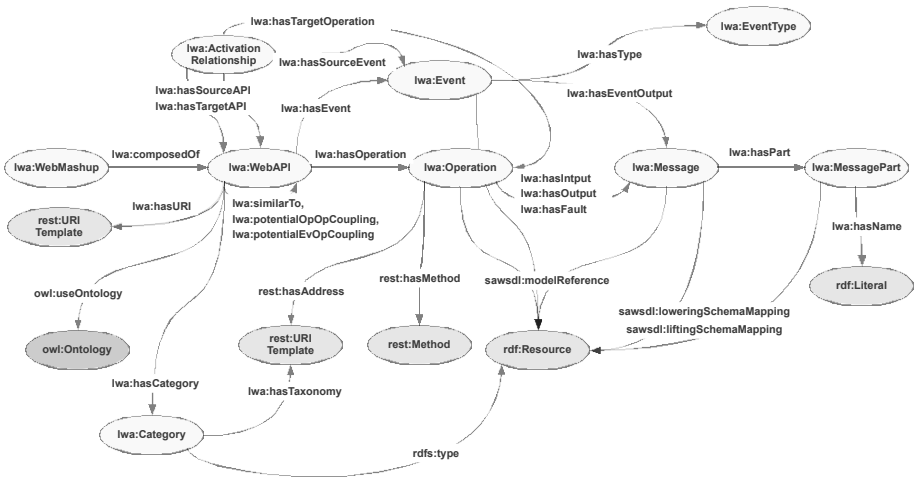


**Fig. 2.** The semantics-enabled Linked Web API conceptual model.

Basic elements of the model have been defined in the `lwa` namespace. We rely on existing vocabularies, namely SAWSDL, RDF(S) and hRESTS, identified with the `sawsdl`, `rdf`/`rdfs`, `rest` namespace prefixes, respectively. We define a

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

`lwa:WebMashup` as composed of Web APIs. Each Web API is uniquely identified by an URI. Moreover, a set of common elements can be identified in Web API descriptions:

- *inputs* and *outputs*, represented through the `Message` construct, which in turn can be composed of several `MessageParts`;
- *operations*, associated with an `address` (which is built by concatenating the API URI with the operation name), used to invoke the operation itself, and a `method` (e.g., POST and GET HTTP methods) for its invocation;
- *events*, to model user's interactions with the Web API interface (they are commonly used in complex Web APIs such as `GoogleMaps`); events are described by an event type (e.g., `onmouseover`, `onmouseclick`) and the outputs or arguments which are raised on event occurrence (for instance, a click on a map raises an event which contains the coordinates of the points which the mouse is positioned on);
- *categories*, that are taken from public classifications available on the Web (see for example the ProgrammableWeb site, where 67 categories such as `mapping`, `payment`, `search` are considered).

To add semantics to the Linked Web API model, we used the *model reference*, *lifting schema mapping* and *lowering schema mapping* constructs proposed in the SAWSDL specification. Schema mappings are used to provide grouding from the conceptual model to the concrete message formats (lowering schema mapping) and viceversa (lifting schema mapping). Model reference construct is used to associate operation names, inputs and outputs and event arguments to concepts taken from the web of Linked Data.

### 2.1 Linking Web APIs

Publication of new Web APIs and new web mashups on the web of Linked Data must be properly supported to help providers in identifying links between them. We consider three kinds of links between Web APIs: in the following, for each kind of link, we will provide a short description, an example, semi-automatic techniques relying on the web of Linked Data to support the link identification and the way the link is included in the model in Figure 2. Additional details can be found in [2].

---

**Similar Web APIs**

---

*A) Synopsis.* Web APIs are linked towards other APIs that provide similar functionalities. This means that linked Web APIs present semantically close I/O messages and operations and compatible categories. Web API similarity is evaluated with respect to a set of operations in the API that is source of the relationship. In Figure 2, Web API similarity is represented through the `lwa:similarTo` construct.
*B) Example.* Both `Rottentomatoes.com` API and the `Internet Movie Database` (`IMDb`) API present an operation for searching movies; the operation requires as

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

input one or more keywords and returns movie information. Therefore, with respect to this operation, the `Rottentomatoes.com` and `IMDb` APIs are similar.

*C) Semi-automatic link identification.* We provide a set of matching techniques to identify similar Web APIs, extensively defined in [2]. These techniques are based on the computation of a concept affinity $CAff() \in [0..1]$ to quantify the degree of similarity between pairs of concepts, used in the semantic annotation of, respectively, (i) operations and (ii) I/Os parameters. Here we simply state that $CAff$ is based on both a terminological (domain-independent) matching based on the use of WordNet and a semantic (domain dependent) matching based on the ontologies on the web of Linked Data, used as source of knowledge. The concept affinity evaluation algorithm looks for a path of relationships between two concepts $c_1$ and $c_2$; we consider four cases: (1) $c_1$ and $c_2$ are defined as equivalent in one of the ontologies; (2) there is a path of *isa* relationships from $c_1$ to $c_2$ in one of the ontologies; (3) $c_1$ and $c_2$ are not semantically related in any ontology, but their names belong to the same synset in WordNet; (4) $c_1$ and $c_2$ are not semantically related in any ontology and their names belong to different synsets in WordNet, but there is a path of hyponymy/hypernymy relations which connects the two synsets. The affinity between two concepts $c_1$ and $c_2$ is maximum (that is, equal to 1.0) in the first and third case, otherwise, the highest the length of the path in the second and fourth case, the lowest is concept affinity. Two Web APIs are similar if they are classified in the same category and the average $CAff$ between their operations and I/O messages, namely the *functional similarity degree* between Web APIs, exceedes a threshold experimentally set. The average $CAff$ is computed, through the application of the Dice formula [8], as the average affinity between pairs of concepts, one from the first API and one from the second one. Pairs of concept to be considered in the average similarity computation are selected according to a maximization function that relies on the assignment in bipartite graphs and has been introduced in [1]. This function ensures that each concept from the first API participates in at most one pair with one of the concepts from the second API. The formula the functional similarity degree is summarized in Appendix A.

**Op2Op wiring of Web APIs**

*A) Synopsis.* Web APIs can be aggregated by wiring their operations. In this case, the outputs of an operation from the first Web API can be used as inputs of an operation from the second Web API. Potential coupling between Web APIs is checked with respect to a set of operations in the first Web AP. In Figure 2, this kind of link between Web APIs is represented through the `lwa:potentialOpOpCoupling` construct.

*B) Example.* The `Rottentomatoes.com` API presents an operation to find movies given one or more keywords (for instance, to find Antonioni's movies). Movie titles can be used to retrieve related video by invoking the proper operation from the `YouTube` API.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

*C) Semi-automatic link identification.* This kind of link is identified by evaluating the average $CAff$ between the outputs of the operations in the first API and the inputs of the operations in the second one. The link is set if the average $CAff$, denoted with *functional Op2Op coupling degree* (see Appendix A), exceeds a threshold experimentally set.

**Ev2Op wiring of Web APIs**

*A) Synopsis.* Web APIs can be wired also at the UI level, by coupling an event from the first Web API with an operation from the second one. Event-operation coupling is made according to a publish/subscribe-like mechanism. In this way, interactions with the UI of the first Web API raise events that trigger operations from the second API, ensuring the synchronization of all Web APIs that compose the web mashup. An event-operation pair is represented in the model through an activation relationship. An activation relationship is modeled with reference to the source and target Web APIs (`hasSourceAPI` and `hasTargetAPI` constructs), to the source event (`hasSourceEvent` construct) and to the target operation (`hasTargetOperation` construct). Potential coupling between Web APIs is checked with respect to a set of events in the first Web API. In Figure 2, this kind of link between Web APIs is represented through the `lwa:potentialEvOpCoupling` construct.
*B) Example.* The `Rottentomatoes.com` API presents an operation to find locations of movie theatres. These locations can be displayed an a map using the `GoogleMaps` APIs. If the zoom level on the map is changed, the list of movie theatres and related information must be properly updated.
*C) Semi-automatic link identification.* This kind of link can be identified in a similar way with respect to the previous one. The average $CAff$ between the arguments of the events from the first Web API and the inputs of the operations from second one is evaluated and the link is set if the average $CAff$, denoted with *functional Ev2Op coupling degree* (see Appendix A), exceeds a threshold experimentally set.

## 3 The I-MASH platform

The lightweight Linked Web API model is the basis for the I-MASH platform, an on-going project whose architecture is shown in Figure 3. The core module of the platform is the *Mashup Engine*, which implements the Web API selection and aggregation. The engine also includes the modules for $CAff$ evaluation, relying on the terminological knowledge provided by WordNet and on the domain-specific knowledge from the web of Linked Data, that is accessed through the *Linked Web API Repository*. The *Linked Web API Repository* maintains the representation of Linked Web APIs published according to the model presented in Figure 2. The Linked Web API descriptions refers to things (i.e., URIs of ontological concepts) taken from the web of Linked Data, which the *Linked Web API Repository* is built upon. Other I-MASH modules, such as the *Mashup En-*

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

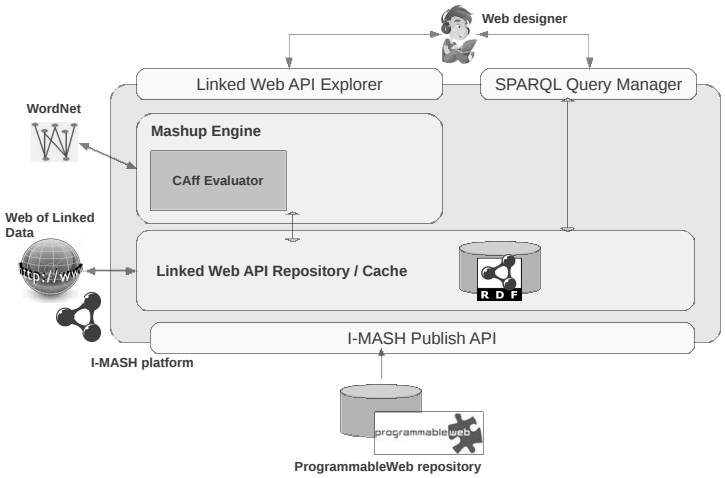


**Fig. 3.** The I-MASH platform architecture.

*gine* and the I-MASH *Publish API*, exploit the *Linked Web API Repository* to access the web of Linked Data. The I-MASH *Publish API* module implements the semi-automatic link identification metrics introduced in the previous section and is used to publish new Web APIs extracted from the ProgrammableWeb repository by invoking its methods (`api.programmableweb.com/`). Finally, the *Linked Web API Explorer* supports the web designer to find Web API descriptions and compose them in a web mashup, by leveraging the net of links in the *Linked Web API Repository*. For skilled web designers, we equipped the I-MASH platform with an interface to directly query the *Linked Web API Repository* through the formulation and execution of SPARQL queries. For more details on the I-MASH platform, we refer to [2].

## 4 Concluding remarks

In this paper we proposed a lightweight model to support both providers who publish new Web APIs used for accessing the web of Linked Data and web designers who aim at exploring and selecting available Web APIs for building or maintaining a web mashup. Additional kinds of relationships to link Web APIs will be studied and analyzed. Among them, we will consider complemetary Web APIs, that is, Web APIs which start from similar inputs and provide alternative functionalities, according to their use in previously defined web mashups. For instance, according to ProgrammableWeb, the `Rottentomatoes.com` and the `YouTube` APIs have been used together in four web mashups out of seven mashups which include the `Rottentomatoes.com` API. The use of traditional support and confidence DM metrics to infer Web API complementarity will be studied, following preliminary results discussed in [6].

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

## References


1. D. Bianchini, V. De Antonellis, and M. Melchiori. Flexible Semantic-based Service Matchmaking and Discovery. *World Wide Web Journal*, 11(2):227–251, 2008.
2. D. Bianchini and V. De Antonellis. *Linked Data Services and Semantics-enabled Mashup*, chapter on Semantic Search on the Web, pages 281–305. Springer, 2012.
3. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story so Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
4. M. d'Aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, and D. Guidi. Toward a New Generation of Semantic Web Applications. *IEEE Inte*, 23(3):20–28, 2008.
5. M. Maleshkova, C. Pedrinaci, and J. Domingue. Semantic annotation of Web APIs with SWEET. In *Proc. of the 6th Workshop on Scripting and Development for the Semantic Web*, 2010.
6. M. Melchiori. Hybrid techniques for Web APIs recommendation. In *Proceedings of the 1st International Workshop on Linked Web Data Management*, pages 17–23, 2011.
7. C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecky, and J. Domingue. iServe: a Linked Services Publishing Platform. In *Proceedings of ESWC Ontology Repositories and Editors for the Semantic Web*, 2010.
8. C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.


## A Formulas for Web API linking identification

**Functional Web API similarity**

$$Sim_{IO}(\mathcal{W}_i, \mathcal{W}_j) = \frac{1}{3}\left[\frac{\sum_{s,t} CAff(in_s, in_t)}{|IN(\mathcal{W}_j)|} + \frac{\sum_{h,k} CAff(out_h, out_k)}{|OUT(\mathcal{W}_i)|} + \frac{\sum_{l,m} CAff(op_l, op_m)}{|OP(\mathcal{W}_i)|}\right]$$

- $IN(\mathcal{W}_j)$ (resp., $OUT(\mathcal{W}_i)$) is the set of concepts used to annotate the operation inputs of the $\mathcal{W}_j$ Web API description (resp., the operation outputs of the $\mathcal{W}_i$ Web API description);
- $OP(\mathcal{W}_i)$ is the set of concepts used to annotate the operations of the $\mathcal{W}_i$ Web API description;
- $in_s \in IN(\mathcal{W}_i)$, $in_t \in IN(\mathcal{W}_j)$, $out_h \in OUT(\mathcal{W}_i)$, $out_k \in OUT(\mathcal{W}_j)$, $op_l \in OP(\mathcal{W}_i)$, $op_m \in OP(\mathcal{W}_j)$.

**Web API Ev2Op coupling**

$$Coupl_{EV2OP}(\mathcal{W}_i, \mathcal{W}_j) = \frac{\sum_{s,t} Coupl_{EvOp}(ev_s, op_t)}{|EV(\mathcal{W}_i)|} \quad ev_s \in EV(\mathcal{W}_i),\ op_t \in OP(\mathcal{W}_j)$$

$$Coupl_{EvOp}(ev_s, op_t) = \frac{\sum_{h,k} CAff(out_h, in_k)}{|OUT_{ev}(ev_s)|}$$

**Web API Op2Op coupling**

$$Coupl_{OP2OP}(\mathcal{W}_i, \mathcal{W}_j) = \frac{\sum_{s,t} Coupl_{OpOp}(op_s, op_t)}{|OP(\mathcal{W}_i)|} \quad op_s \in OP(\mathcal{W}_i),\ op_t \in OP(\mathcal{W}_j)$$

$$Coupl_{OpOp}(op_s, op_t) = \frac{\sum_{h,k} CAff(out_h, in_k)}{|OUT(op_s)|}$$

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 