Recommending Experts for Collaboration in Mashup Development

Valeria De Antonellis and Michele Melchiori

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

Abstract. Mashup development is currently powered by Web sites, as ProgrammableWeb and Mashape, that offer large, continuously updated and growing, catalogues of software components accessible through Web APIs. Developing Web mashup is becoming attracting for developing Web and enterprise applications, but it requires specialized knowledge about Web APIs, their technologies and the way to combine them. In this paper, firstly we discuss how social-based semantic approaches can support the phases of mashup development. Then, we describe our framework LINKSMAN (LINKed data Supported MAshup collaboratioN) for expert search in enterprise mashup development based on integrating knowledge both internal and external to enterprises.

1 Introduction

Web mashup is nowadays a popular approach for implementing Web applications leveraging on third party functionalities and data. The approach is powered by Web sites offering large, ever growing, catalogues of software components accessible through Web APIs (e.g., ProgrammableWeb, Mashape). On the one hand, this availability of Web APIs makes mashup interesting also for enterprises [5]. On the other hand, developing mashups requires specialized knowledge about Web APIs, their technologies and the way to combine them in a meaningful way. This kind of knowledge can be available in an organization, but distributed among various experts. An opportunity in this sense is to take advantage of Enterprise 2.0 that is a recent specialization of the Web 2.0 social-based technologies to the enterprise needs and requirements.

In this paper, firstly we discuss how social-based semantic approaches can support the different mashup development phases. Then, we discuss the specific role of our framework LINKSMAN (LINKed data Supported MAshup collaboratioN) for expert search in enterprise mashup development.

In general, expert search systems aim at identifying candidate experts with respect to a topic of interest and ranking them with respect to the expertise level based on available sources of evidence, typically document contents. On the contrary, the work [8] discusses the potential benefits and drawbacks of using Linked Open Data (LOD) as sources for expert search. The authors analyze what is currently present and missing in various LOD datasets w.r.t. different

Q.Z. Sheng and J. Kjeldskov (Eds.): ICWE 2013 Workshops, LNCS 8295, pp. 215–222, 2013.

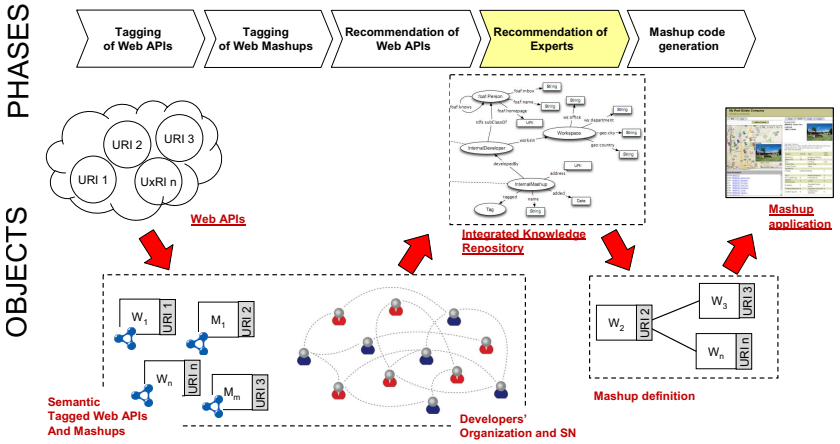

Fig. 1. Integrated Development of Mashups

kinds of expert search. The authors in [6] remark that in the existing expert search approaches there are several flaws related to the lack of focus on realistic applications and limitations to consider a single data source. Supporting the collaboration inside organizations modeled as P2P networks and the contribution that semantics can give to the problem have been tackled in the framework proposed in [7]. With respect to expert search approaches for general purposes, LINKSMAN specializes in enterprise mashup development and exploits in an integrated way knowledge that is both private internal to the enterprise and external. In LINKSMAN, the internal knowledge mainly concerns developers, their organization inside the enterprise, their social connections and software artifacts they have developed. To this purpose, we adopt ProgrammableWeb[1] as external source because this site is actually one of the most updated and popular for mashup development. External knowledge concerns mashups and Web APIs from public Web APIs repositories.

This paper is organized as it follows. In Section 2 we present a the integrated development scenario and identify its use cases. In Section 3 the models for the internal and external knowledge are given. In Section 4 we identify the collaboration patterns and provide suitable metrics for implementing them. Finally, Section 5 provides final remarks and future work.

2 Integrated Mashup Development Scenario

In order to clarify the problem, let us consider a developer working for the marketing department of an enterprise, who has to build an application to visualize,

[1] See `http://www.programmableweb.com`: the last access on May 20th, 2013 counts more than 9,100 Web APIs and 7000 mashups.

on an interactive map, information about potential markets, sales and demographic data. This application can be obtained by composing functionalities and data that are made available by means of Web APIs. To this aim suitable Web APIs have to be selected (for example, the GeoData Demographics API, that provides demographic data for a given zone) from public repositories and possibly integrated with private components. This selection activity may exploit advanced search techniques [1,3]. However, using the selected Web APIs in order to build an application may be difficult because of the need to understand their syntax, semantics and data formats. In this context, it is frequent that a developer searches for advices and collaboration with other developers of the same enterprise. Moreover, it is reasonable to assume that her preference is for experts that she can contact/involve because some social or organizational contacts exist with them.

Integrated Development Approach. In our approach we identify different phases of development of mashups and involved objects as represented in Fig. 1. Our purpose is to support the developer along the mashup lifecycle represented by these phases. In a repository like ProgrammableWeb, Web APIs and mashups are described in terms of URIs, descriptions, categories, tags and technical features (protocols, data formats and security features). This characterization is further enriched in our approach by associating Web APIs and mashups with semantic tags (*Tagging of Web APIs* and *Tagging of Mashups* phases, in the figure) according to the framework described in [2]. The approach has been extended in [4] where we provided a framework based on different Web API features to support Web API search and reuse in enterprise mashup design. Based on semantically tagged Web APIs and Mashups we define different kinds of Web API request, as detailed in the use cases described below, to support the *Recommendation of Web APIs* phase. Concerning involved developers, we model their organization and social network. This knowledge is integrated with knowledge about Web APIs and mashups and made available as *Integrated Knowledge Repository*. On this repository, we define a framework for the *Recommendation of Experts*, that is the focus in this paper, with the purpose to advice the developer with experts inside the organization that can be involved or give advices on the design and code writing of the mashup.

Use cases. Concerning supporting a developer in the phases above described, we identified three main use cases for mashup development (see Fig. 2): (i) developing a new mashup, (ii) completing an existing mashup with functionalities provided by additional Web APIs, and (iii) substitution of a Web API in an existing mashup with a functionally equivalent one (e.g., to improve the quality or replacing a no more available API). These cases are supported by specific kinds of Web API request as detailed in [3], that here we summarize:

– *single Web API selection*, when the designer is developing a new mashup and aims at finding a Web API by specifying the Web API category and semantic tags; optionally, the designer may also specify the desired technical features for the Web API to search for;

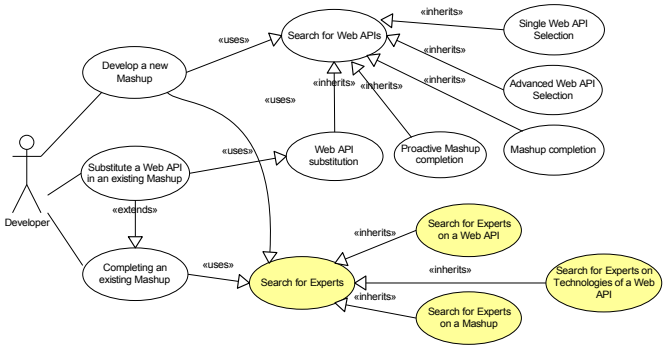

Fig. 2. Use cases for mashup development

- *advanced single Web API selection*, it is a variant of the previous one; in this case, the designer has also in mind the kind of mashup where the Web API has to be included, that is specified through a set of semantic tags;
- *mashup completion*, the mashup is already partially specified and the designer desires adding a new Web API; in this case, the request includes the set of Web APIs already in the mashup; the category for the requested Web API, semantic tags featuring it and the mashup can be optionally specified;
- *proactive mashup completion*, it is a variant of the previous use case; the designer does not specify the category and the semantic tags of the Web API to search for, since she does not know exactly what APIs are available, what to look for and using what tags for the request; therefore, she relies on suggestions of the system, which proposes candidate Web APIs based on existing mashups that includes Web APIs similar to the one already in the mashup;
- *Web API substitution*, when the designer desires to substitute a Web API in an existing mashup; in this case, the category and the tags are automatically extracted from the Web API to substitute;.

Example 1. A request to find a Web API in the category `Mapping` to be used for plotting data on maps together with the `Data-Planet` and `GeoData Demographics` APIs (*mashup completion* request) can be expressed by specifying the desired category, a set of semantic tags {`plot`, `geographic map`} and the set of Web APIs already in the mashup.

The use cases related to the *Search for Experts* (see Fig. 2) to support recommendation of experts are discusses in the following Sect 4.

3 Data Model for Integrating Public and Enterprise Knowledge

We model as linked data (LD) both the public knowledge about Web APIs, based on ProgrammableWeb and the knowledge about the organization and

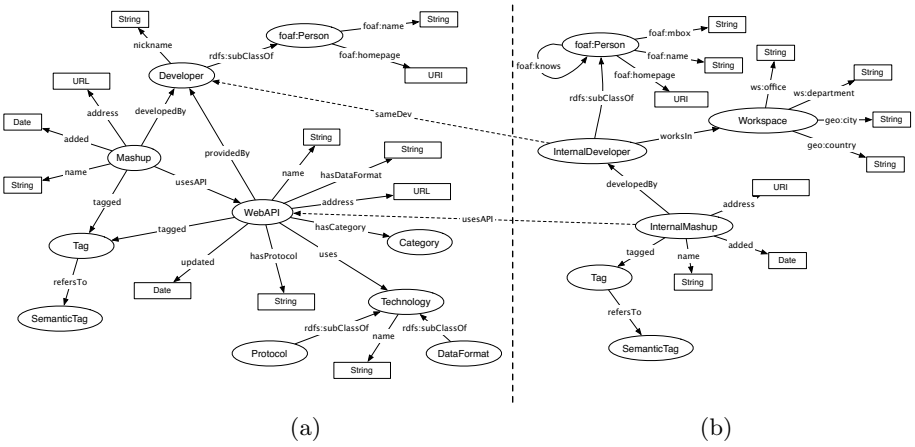

Fig. 3. Data models: (a) `PW repository`; (b) `developer organization`

social network of developers. The models are represented ad RDF vocabularies and are linked to provide an integrated data set.

Design of the Vocabularies. The representation of the linked RDF vocabularies concerning ProgrammableWeb (PW) and Developers Organization (DO) is shown in Figure 3. The former vocabulary has been obtained by analyzing the content of PW. The DO vocabulary keeps into account and estends the content of a well know Enterprise 2.0 platform, that is IBM Connections. Developers in the DO vocabulary have social and collaboration relationships inside the organization through the `foaf:knows` property. The PW and DO data sources are linked (see dashed links in Figure 3) according to the following criteria: (i) a `sameDev` link is established between a DO resource `InternalDeveloper` and a PW resource `Developer` when an internal developer is also registered as developer in PW; (ii) a DO mashup (instance of the `InternalMashup` class) is linked through a `usesAPI` property to each PW `WebAPI`.

Descriptors for expert search. In the PW and DO data models we identify the classes that are most suitable sources of expertize evidence (see Table 1). Moreover, we establish perspectives that can be applied on these classes: *Organization perspective*, *Social perspective*, *Web APIs perspective* and *Technologies perspective*. A perspective is a set of elements (classes or properties) that are related to a specific aspect of the domain represented in the data models. Table 1 shows applicability of each perspective to classes. For example, the `InternalDeveloper` class has an *Organization perspective* in order to indicate that there is a set of elements in the vocabularies that provides organization related information about an internal developer, such as office and department.

Descriptors are associated with the objects of these classes based on the applicable perspectives. A descriptor for an object o_i according to a perspective p is defined as a set of terms, $des_p(o_i) = \{t_{i1}, t_{i2}, ..., t_{in}\}$.

Table 1. Applicability of perspectives to DO and PW classes

Class	Perspectives			
	Organization	*WebAPIs*	*Technologies*	*Social*
`InternalDeveloper`	**ok**	**ok**	**ok**	**ok**
`InternalMashup` , `Mashup`		**ok**	**ok**	
`WebAPI`			**ok**	

Example 2. For an `InternalDeveloper` object, a descriptor according to the *Organization* perspective is built as the union of values of `country`, `city`, `department`, `office` properties of the `workplace` objects associated with the `InternalDeveloper` object.

Similarity of Descriptors. Because the descriptors are set of terms, a measure of similarity between pairs of descriptors on the same perspective is defined according to the classical Dice's formula for similarity over sets:

$$Sim(des_p(o_i), des_p(o_j)) = \frac{2 \cdot |des_p(o_i) \cap des_p(o_j)|}{|des_p(o_i)| + |des_p(o_j)|} \quad (1)$$

The similarity ranges in [0..1].

4 Expert Search Use Cases

An expert search use case specifies a type of request in order identify experts about a specific kind of problem. We define in a general way a request R submitted by a developer D_R as follows.

$$R =< T_R, D_R, WA_R > \quad (2)$$

where T_R is the type of request, $WA_R = \{W_{Ri}\}$ is a set of Web APIs W_{Ri} specified by the developer D_R for implementing a mashup. We distinguish three different types of request R corresponding to the use cases in Fig. 2 that specialize the *Search for Experts* one:

R1) search for experts on a specific Web API W_R, where $WA_R = \{W_R\}$;
R2) search for experts on the technologies of a Web API W_R, where $WA_R = \{W_R\}$;
R3) search for experts on a mashup whose component APIs are $WA_R = \{W_{Ri}\}$.

In the third case D_R is looking for developers that have experience in mashups built from the set of Web APIs WA_R or at least with a not empty subset if it.

In the literature, different assumptions are made on what makes an expert and how to assess her expertise [8]. In order to answer to R we focus on content/activities and formulate the expertize assumptions as in the following: (i) if a developer has developed a mashup containing a Web API W_{Ri}, then she might have competencies about the W_{Ri} Web API; (ii) if a developer has developed a mashup containing a Web API W_{Ri}, then she might have competencies about the technologies of W_{Ri}; (iii) if a developer has developed a mashup including a subset of the Web APIs in $\{W_{Ri}\}$, then she might have competencies on how to compose these APIs.

Table 2. Definition of metrics for the collaboration patterns

Pattern	Definition of $m(R, D_i)$
P1	$\alpha \cdot Sim(des_{pO}(D_R), des_{pO}(D_i))+$ $\beta \cdot Sim(des_{pS}(D_R), des_{pS}(D_i))\in[0,1]$, *with* $\alpha, \beta \in [0..1]$ *and* $\alpha + \beta = 1.0$
P2	$\alpha \cdot Sim(des_{pO}(D_R), des_{pO}(D_i))+$ $\gamma \cdot Sim(des_{pT}(D_R), des_{pT}(W_R))\in[0,1]$, *with* $\alpha, \beta, \gamma \in [0..1]$ *and* $\alpha + \beta + \gamma = 1.0$
P3	$\alpha \cdot Sim(des_{pO}(D_R), des_{pO}(D_i))+$ $\beta \cdot Sim(des_{pS}(D_R), des_{pS}(D_i))+$ $\gamma \cdot Sim(des_{pW}(D_R), des_{pW}(\{W_{Ri}\}))\in[0,1]$, *with* $\alpha, \beta, \gamma \in [0..1]$ *and* $\alpha + \beta + \gamma = 1.0$

4.1 Definition of Collaboration Patterns

We define a collaboration pattern in a general way as:

$$CP =< R, m, \delta, \prec > \tag{3}$$

where R is the request, m is the metric used to measure the matching between a candidate developer and R, using the similarity between descriptors as in Equation (1). Application of a collaboration pattern CP to a dataset described according to the data models of Section 3 produces a set of developers matching the request R. The developers are then ordered according to the ranking function $\prec$, that is $D_i \prec D_j$ if $m(R, D_i) \leq m(R, D_j)$. A threshold δ is used to filter out less relevant search results.

The generic collaboration pattern is specialized for each type of request. In particular, based on a suitable expertise assumption, each type of request determines the structure of R, m and a set of considered perspectives. The definitions of the metrics $m()$ for each specialization of the collaboration pattern are given in Table 2.

*Pattern p*1 *(searching for expertize on a given Web API)* if the developer D_i has used the Web API W_R in some mashups (that is, $W_R \in des_{pW}(D_i)$), then $m(R, D_i) \neq 0$ and it is based on a weighted sum of two terms: i) similarity of D_i and D_R w.r.t. the *Organization* perspective, ii) similarity of D_i and D_R w.r.t. the *Social* perspective. So, developers that are closer to D_R w.r.t. the *Social* and *Organization* perspectives will be ranked better, because it is supposed to be easier for the developer D_R to contact them and to get their collaboration. In the current version of LINKSMAN, we set $\alpha = \beta = 0.5$.

*Pattern p*2 *(searching for expertize on the technologies of a given Web API)* if developer D_i has used APIs that share technologies with the Web API W_R (that is, $Sim(des_{pT}(D_i), des_{pT}(W_R)) \neq 0$), then $m(R, D_i) \neq 0$ and it is based on a weighted sum of three terms: i) similarity of D_i and D_R with respect the *Organization* perspective, ii) similarity of D_i and D_R with respect to the *Social* perspective, iii) similarity between technologies that D_i has used in developer's mashups and technologies featuring W_R. Also in this case and in the following *p*3, we set equally the values of α, β and γ.

*Pattern p*3 *(searching for expertize on a given mashup)* if developer D_i has used at least one of the APIs $\{W_{Ri}\}$ specified in the request (that is, $Sim(des_{pT}(D_i), des_{pT}(W_R)) \neq 0$), her evaluation is not zero and it is based on a weighted sum of three terms: i) similarity of D_i and D_R with respect the *Organization*

perspective, ii) similarity between D_i and D_R with respect to the *Social* perspective iii) similarity between the set of APIs used by D_i in her mashups and $\{W_{Ri}\}$ (*WebAPIs* perspective). In the Table 2, pW denotes the WebAPIs perspective.

5 Conclusions and Future Work

In this paper, we described a social and semantic based approach to support enterprise mashup development. In particular, we discussed how different use cases can be supported by Web API search techniques that are based on social semantic tagging. Then, we presented the LINKSMAN framework for expert search of mashup developers based on collaboration patterns. Future work includes identification of additional collaboration patterns, using additional external sources of knowledge (e.g., Mashape, `http://www.mashape.com`) and performing evaluation based on the prototype we are developing.

References

1. Bianchini, D., De Antonellis, V., Melchiori, M.: Service-based Semantic Search in P2P systems. In: ECOWS 2009 - 7th IEEE European Conference on Web Services, pp. 7–16 (2009)
2. Bianchini, D., De Antonellis, V., Melchiori, M.: Semantic Collaborative Tagging for Web APIs Sharing and Reuse. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 76–90. Springer, Heidelberg (2012)
3. Bianchini, D., De Antonellis, V., Melchiori, M.: A Linked Data Perspective for Effective Exploration of Web APIs Repositories. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 506–509. Springer, Heidelberg (2013)
4. Bianchini, D., De Antonellis, V., Melchiori, M.: A Multi-perspective Framework for Web API Search in Enterprise Mashup Design. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 353–368. Springer, Heidelberg (2013)
5. Hoyer, V., Fischer, M.: Market overview of enterprise mashup tools. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 708–721. Springer, Heidelberg (2008)
6. Hristoskova, A., Tsiporkova, E., Tourw, T., Buelens, S., Putman, M., De Turck, F.: Identifying experts through a framework for knowledge extraction from public online sources. In: 12th Dutch-Belgian Information Retrieval Workshop (DIR 2012), Ghent, Belgium, pp. 19–22 (2012)
7. Montanelli, S., et al.: The ESTEEM platform: Enabling P2P Semantic Collaboration through Emerging Collective Knowledge. Journal of Intelligent Information Systems 36(2), 167–195 (2011)
8. Stankovic, M., Wagner, C., Jovanovic, J., Laublet, P.: Looking for Experts? What can Linked Data do for You? In. In: Proc. of Linked Data on the Web (LDOW) at WWW 2010 (2010)