Linked Data Based Expert Search and Collaboration for Mashup

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. Web mashup is becoming more and more popular for both general and enterprise purposes in order to implement applications that leverage on third party components. However, developing a mashup requires specialized knowledge about Web APIs, their technologies and the way to combine them in a meaningful way. For this problem, we describe in this paper an approach for searching experts in the context of enterprise mashup development and in particular, we describe how to implement typical expert search patterns. The approach is based on the integration of knowledge both internal and external to the enterprise and represented as a linked data.

1 Approach Overview

In this paper we discuss the LINKSMAN (LINKed data Supported MAshup collaboratioN) framework for expert search finalized to collaboration in enterprise mashup development. Generally speaking, expert search systems aim at identifying candidate experts on a topic of interest and ranking them with respect to their expertise using available sources of evidence, typically content of documents. With respect to the expert search approaches for general purpose, our proposal: (i) is specialized in enterprise mashup development; (ii) exploits in an integrated way both the internal enterprise knowledge and the public external one; (iii) keeps into account social relationships between expert candidates to rank them. Specifically, internal knowledge mainly concerns developers, their organization inside the enterprise, their social connections and software artifacts they have developed, and cane be extracted by Enterprise 2.0 platforms. External knowledge concerns mashups and Web APIs from public repositories and to this purpose we adopt ProgrammableWeb[1]. Modeling social aspects for building web mashups has been discussed in [1]. The approach has been also extended in [2] where we provided a framework based on different Web API features to support Web API search and reuse. Related efforts in literature about Linked Web services or Linked Web APIs for discovery purposes are the ones described in [4]. Traditional expert search approaches and systems use as sources of evidence closed sets of document and databases. On the contrary, the work [3]

[1] See http://www.programmableweb.com

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 726–729, 2013.

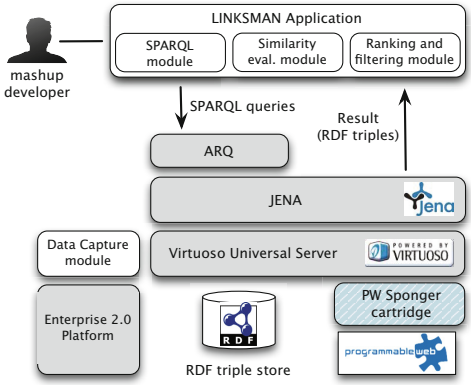

Fig. 1. The LINKSMAN functional architecture

discusses the potential benefits and drawbacks of using Linked Open Data (LOD) for expert search.

In LINKSMAN the external and the internal knowledge are modeled as two Linked Data (LD) vocabularies and data sets. Then, a set of perspectives are identified and associated with the main classes of the LD vocabularies. The perspectives allow for introducing descriptors on the data objects and similarity measures on them. Four different expert search patterns are then identified (see Section 2). The functional architecture of the prototype is illustrated in Figure 1 and uses the Virtuoso RDF data storage system. In the figure, the white blocks are component specifically developed for the LINKSMAN prototype and the grey ones are third party software components. The internal knowledge data set is stored in the is data storage component. A developer in order to submit a request in LINKSMAN specifies: (i) the goal (e.g., searching for competencies on a given Web API); (ii) either one Web API, for the first three patterns $R1$, $R2$ and $R3$, or one/more Web APIs for pattern $R4$. The submitted request is then translated by the *SPARQL module* into a set of SPARQL queries processed by *ARQ*, a SPARQL query engine to translate the query into Jena API calls. The RDF triples returned to the LINKSMAN application are used to build descriptors evaluated by the *Similarity evaluation module*, according to the metrics discussed in the next section. The search results are then filtered and ranked to be presented on the application GUI.

1.1 Perspectives and Descriptors for Expert Search

In the external and the internal knowledge LD vocabularies, we identify a set of classes that are suitable sources of evidence for establishing the expertise of a developer. These classes are: `InternalDeveloper`, `InternalMashup`, `WebAPI` and `Mashup` and a set of perspectives applicable to each of them is identified. A perspective is a set of class attributes describing a given aspect of the knowledge in

the vocabularies. The attributes belong to one or more classes of the vocabularies. We consider four perspectives, describing: (i) enterprise-related information about developers (*Organization perspective*) and their social relationships (*Social perspective*); (ii) Web APIs involved in the mashup construction (*Web APIs perspective*); (iii) technologies, such as data formats and protocols, of the involved Web APIs (*Technologies perspective*); (iv) functionalities offered by Web APIs (*Functionality perspective*). For example, the `InternalDeveloper` class has an *Organization perspective* that collects the attributes in the vocabularies that provide organization related information about an internal developer, such as the office and department.

Descriptors are associated with the objects of these four classes based on the perspectives applicable to each class. A descriptor for an object o_i of the class C according to a perspective p is defined as a set of terms that are values of the attributes defined by p on C:

$$des_p(o_i) = \{t_{i1}, t_{i2}, ..., t_{in}\} \tag{1}$$

For example, for an `InternalDeveloper` object, a descriptor according to the *Organization* perspective is build as the union of values of attributes describing the knowledge about the organization of a developer working for the organization (e.g., `city`, `department`, `office`).

Descriptors for other perspectives and classes are defined in a similar way. A measure of similarity between two descriptors on the same perspective is defined according to the classical Dice's formula for similarity over sets:

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

The similarity ranges in [0..1].

2 Expert Search Patterns

An expert search pattern specifies a type of expertise request. We define in a general way a request R submitted by a developer D_R as follows: *R1*) search for experts on a specific Web API W_R; *R2*) search for experts on the technologies of a Web API W_R; *R3*) search for experts on the functionalities of a Web API W_R; *R4*) search for experts on a mashup whose component APIs are $\{W_{Ri}\}$. In the case *R4*, the developer D_R is looking for developers that have experience in mashups that are built from a given set of Web APIs (or a not empty subset of it). A metrics for each search pattern is defined in order to measure the matching between a candidate developer and R, using the similarity between descriptors as in Equation (2). Application of an expert search pattern produces a list of developers, possibly empty, matching the request R. The list is ordered according to a ranking function and a threshold is used to filter out the less relevant results. Let us introduce informally the rationale for defining the metrics m for the pattern $R2$. The metrics for $R1$, $R3$ and $R4$ are defined in a similar way. In the case $R2$, if the developer D_i has used APIs that share technologies with the Web API W_R (that is, $Sim(des_{pT}(D_i), des_{pT}(W_R)) \neq 0$, where pT denotes

the *Technologies* perspective), then $m(R, D_i)$ is not equal to zero and is based on a weighted sum of three terms: i) similarity m 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 the technologies that D_i has used in developer's mashups and technologies featuring W_R.

Considering the aging of experience In order to take into account the obsolescence of the experience of a developer D_i in using a given Web API W_j, its technologies or its functionalities in the context of developing a mashup we extend the definition of descriptor assigning an age to its components. Aging is meaningful when a descriptor of a developer is considered according to the $WebAPIs$, $Functionalities$ and $Tecnologies$ perspectives. The age associated with an element t_{ij} in the experience of D_i is therefore weighted by introducing an obsolescence factor that reduces accordingly the m value:

$$obs(t_{ij}, D_i) = \frac{1}{K^{age(t_{ij}, D_i)}} \tag{3}$$

where $K > 1$.

3 Conclusions and Future Work

In this paper, we described a Linked Data based framework to support expert search for collaboration in enterprise mashup development. The more relevant features of the framework are: (i) the internal and external knowledge sources are modeled and made suitable as RDF datasources; (ii) four patterns to support different kinds of expert search have been defined; (iii) an architecture and a prototype application have been designed and implemented. Future work includes more extensive experimentation and use of additional data sources (e.g., Mashape, `http://www.mashape.com`).

References

1. 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)
2. 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)
3. Stankovic, M., Wagner, C., Jovanovic, J., Laublet, P.: Looking for experts? what can linked data do for you? In: Proc. of Linked Data on the Web (LDOW) at WWW 2010 (2010)
4. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Semi-Automatically Modeling Web APIs to Create Linked APIs. In: Proceedings of the ESWC 2012 Workshop on Linked APIs (2012)