Leveraging Social Patterns in Web Application Design

Devis Bianchini(✉), Valeria De Antonellis, and Michele Melchiori

Department of Information Engineering, University of Brescia,
Via Branze, 38, 25123 Brescia, Italy
{devis.bianchini,valeria.deantonellis,michele.melchiori}@unibs.it

Abstract. In this paper we propose a multi-layered model meant for the selection of data services for web application design. Our aim is at complementing existing data service selection criteria, e.g., matching based on (semantic) data coming from the services, by also considering the experience of other developers, who used the services in the past for designing their own web applications. In this sense, it becomes crucial the importance that a developer gives to past experiences of other developers in selecting a data service, that might depend on the social relationships that relate the developers each other as well. The model proposed in this paper takes into account these challenging issues by considering available data services, web applications where services have been aggregated, and social relationships between web application developers, which identify different kinds of *social patterns*.

1 Introduction

Modern web applications should be created also exploiting information made available in the form of data services, that enable to access data from different web sources. Data services need to be selected, before being properly integrated to become meaningful and valuable. Let's consider a web application developer, working for the marketing department of an enterprise, who has to build an application that integrates information about potential markets, sales and demographic data. This application requires to merge data coming from sources internal to the enterprise (e.g., information about the target clients) and external data sources (e.g., providing information about demographic data), made available as data services. On the one hand, data services may be selected according to their pertinence with respect to the application that is being designed, e.g., entailed by the (semantic) type of data coming from the sources [1–3]. On the other hand, there are many functional and non functional aspects, such as the ones related to quality of service [4,5] or service trustworthiness and reputation [6], that are not always available within data service descriptions, although they could improve service selection strategies [7]. Nevertheless, in an enterprise context, such as the one depicted in our running example, it is frequent that a developer, who learns by examples, searches for advices based on design experiences of other developers (of the same enterprise or different ones), rather than relying on votes/ratings

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 13–23, 2015.
DOI: 10.1007/978-3-319-19890-3_2

assigned to data services by developers over the Web. Moreover, even if developers' votes would be available to rate data services, approaches for data service selection that rely on such votes [7,8] do not consider another important aspect: the relevance of a data service may depend on the web application design project that is being carried on; such relevance might be high since the service has been already used in similar applications. Finally, in the enterprise context we are considering credibility of developers, who know each others, can be assumed as high, compared to a development scenario where services are ranked and used by developers over the Web. Therefore, for the moment, we do not use advanced developers' credibility assessment techniques such as the ones described in [7–9]. It is reasonable to assume that developer's preference is for experts who can be contacted/engaged, because some mutual social or organizational relationships exist among them. Such relationships constitute particular structures that in [10] are called *social patterns*, that can be fruitfully exploited for data service selection.

The main contribution in this paper is a *multi-layered model*, meant for supporting the selection of data services for web application design purposes, and of metrics specifically based on the model elements. The model is organized on three levels. At the lowest level (*data service layer*), the available data services and corresponding metadata are taken into account to enable service search. At a second level (*web application layer*), the model includes past experiences of collection and aggregation of data services to design web applications; these experiences act as a bridge between the data service layer and the *social layer*, that includes the developers, who experienced the collection of data services, organized according to different social patterns, that reflect different ways developers learn from each other experience in web application design.

The paper is organized as follows. In Section 2 we describe the multi-layered model. In Section 3 we propose the criteria for data service selection based on past experiences and social patterns, with some hints on preliminary experiments that are being carried on. Finally, future work are discussed in Section 4.

2 The Three-Layer Model

Data Service Layer. The multi-layered model we propose in this paper is shown in Figure 1. We define a (web) data service s as a single operation, method or query to access data from a web data source Σ. We model a data service s as a set of service inputs s^{IN} and a set of service outputs s^{OUT}. Service outputs represent data that are accessed through the service s, service inputs are parameters that are needed to invoke the service and access data. We denote with $\mathcal{S}$ the overall set of available data services and with $\Sigma[\mathcal{S}] \subseteq \mathcal{S}$ the set of data services made available by the source Σ. Examples of data services are the methods of a Web API (for instance, the method of GeoData Demographics API[1] that provides demographic data for a given zone), queries formulated by using search-specific languages (such as Yahoo! Query Language,

[1] http://geodataservice.net.

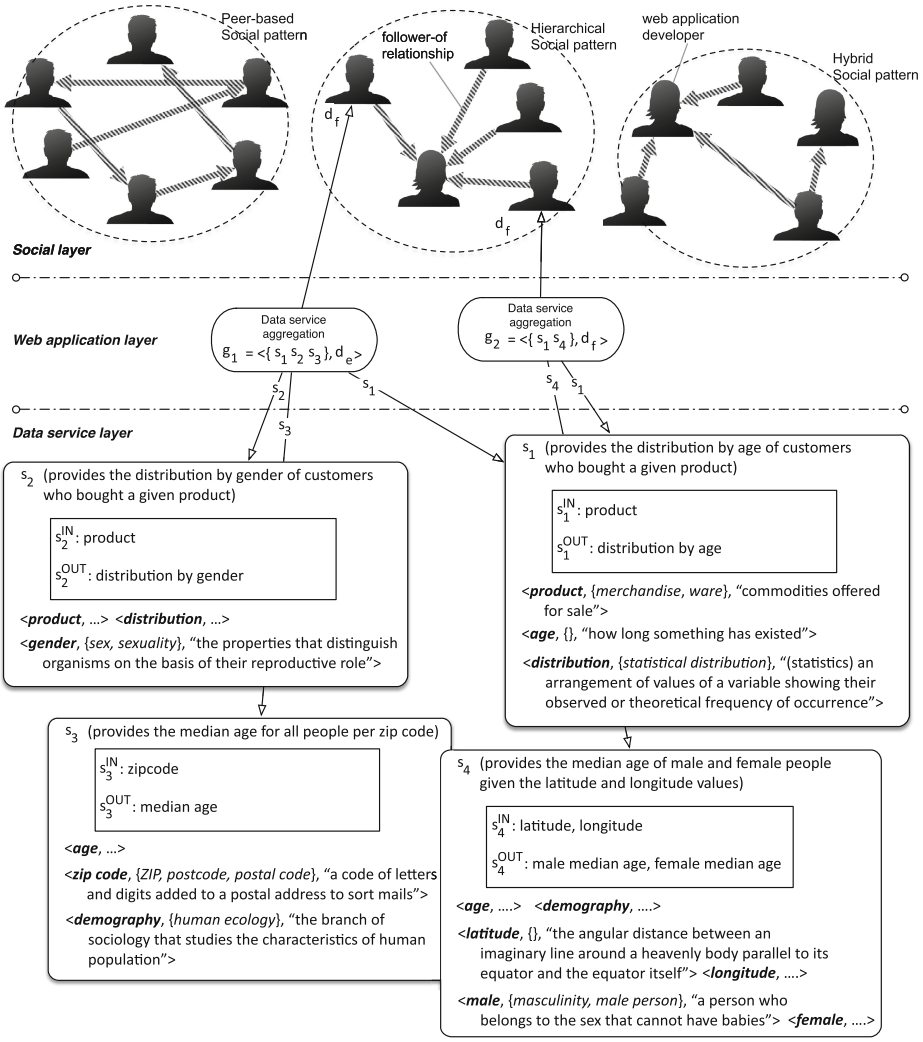

Fig. 1. The multi-layered model for data service selection

`https://developer.yahoo.com/yql/`), services delivering data in tabular format (e.g., Google Fusion Tables, `tables.googlelabs.com/`) or in row format (such as Factual, `http://www.factual.com`). Data services are usually wrapped as web services, that can be implemented according to different styles (e.g., using REST or WSDL).

A data service s is also associated with some *metadata*, that in the current version of the model contains (semantic) tags, used to enable coarse-grained search of the data services. In our model, we admit different ways of assigning (semantic) tags to data services: (i) keywords extracted from the data service name, I/O names and textual description through the application of text mining

techniques (such as stop words removal, stemming, camel case word decomposition, etc.) [11]; (ii) tags, assigned by developers who used the data services to design their own applications. Developers, while assigning tags, might apply sense disambiguation techniques such as the ones we described in [12], based on WordNet lexical system. In Figure 1 four sample data services are shown for the running example, where tags have been disambiguated using WordNet. For each tag, the tag name, the set of synonyms and a human readable definition are given.

Web Application Layer. A data service aggregation represents a web application, that integrates a set of services used to access data coming from different data sources. An aggregation g is modeled as a pair, composed of the set of data services, and the reference to the developer who has been in charge of designing the web application. We denote with $\mathcal{G}$ the overall set of data service aggregations, that is, $g \in \mathcal{G}$, and with $\mathcal{S}[g] = \{s_1, \ldots, s_n\}$ the data services used in g, that is, $g = \langle \mathcal{S}[g], d \rangle$, where d is the developer who designed the web application. Conversely, $\mathcal{G}[s]$ denotes the set of aggregations where s has been used.

Social Layer. The set $\mathcal{D}$ of developers, who used data services to design their applications, is included in the *social layer*. Each developer $d_i \in \mathcal{D}$ is modeled as $\langle \mathcal{G}[d_i], \mathcal{D}^* \rangle$, where $\mathcal{G}[d_i] \subseteq \mathcal{G}$ is the set of data service aggregations designed by d_i in the past, $\mathcal{D}^* \subseteq \mathcal{D}$ is the set of other developers, whom d_i declares to be inclined to learn from in order to design new web applications, formally defined as $\mathcal{D}^* = \{d_j \in \mathcal{D} | d_i \xrightarrow{f} d_j\}$. In the current version of our approach, the *follower-of relationships* are only set after an explicit endorsement of developers. An overview over the network of social relationships between developers might reveal different kinds of *social patterns*, that is, particular configurations of social relationships that can be recognised and represent different design scenarios. In our model, we consider three distinct patterns, exemplified in the social layer of Figure 1: (a) *hierarchical pattern*; (b) *peer-based pattern*; (c) *hybrid pattern*. The first pattern is typical of organisations where the work is performed in a hierarchical way: each developer learns from past choices made by the developer who leads the hierarchy. Usually, in this case we are inside the same enterprise or across different departments of the same enterprise. The second pattern is typical of a totally collaborative environment, both within the same enterprise and across different ones, where a leadership in the selection of data services can not be identified. The third scenario represents an hybrid situation, where a developer is or has been involved in different web application design projects and, maybe depending on the particular application domain, has different other developers whose past choices could be considered to learn from. Automatic detection of social patterns, based on social network properties such as degree centrality, betweness centrality and closeness centrality [13], will be addressed in future work.

3 Data Service Selection Driven by Social Patterns

3.1 Problem Statement

Given a developer $d^r \in \mathcal{D}$, hereafter denoted as the *requester*, who is designing a new web application starting from a set of available data services $\mathcal{S}$, given the set $\mathcal{G}[d_k]$ of data service aggregations designed in the past by developer $d_k \in \mathcal{D}$, our aim is at supporting d^r in performing data service selection, by proposing an ordered set of candidate data services $\mathcal{S}^* \subseteq \mathcal{S}$ taking into account the past experiences in $\mathcal{G}[d_k]$ for each developer d_k such that $d^r \xrightarrow{f} d_k$, that is, for each developer from whom d^r explicitly declared to learn from. The request s^r is formulated as a set of desired (semantic) tags and a set of data services, that have been already included in the application, namely $s^r = \langle \{t^r\}, g^r \rangle$, where $\{t^r\}$ is a list of tags and $g^r = \{s_1, s_2, \ldots s_n\}$ is the list of already selected data service descriptions. Tags in the set $\{t^r\}$ can be disambiguated using WordNet as well. For example, the following request s^r is formulated to find a `demography` data service, annotated with a `postal code`, to produce a distribution of people by `sex`. The service must be used in a web application, that is being designed and already contains data services s_1 and s_2 (see Figure 1):

$s^r = \langle$ t_1^r = {⟨`postal code`, {`zip code`, `ZIP`, `postcode`}, "a code of letters and digits added to a postal address to sort mails"⟩};

t_2^r = {⟨`sex`, {`gender`, `sexuality`}, "the properties that distinguish organisms on the basis of their reproductive roles"⟩};

t_3^r = {⟨`demography`, {`human ecology`}, "the branch of sociology that studies the characteristics of human populations"⟩};

$g^r = \{s_1, s_2\}\rangle$

3.2 Data Service Selection

The overall similarity between s^r and each available data service $s \in \mathcal{S}$, denoted with $Sim(s^r, s) \in [0, 1]$, is used to filter out not relevant data services and is computed as linear combination of two matching techniques:

$$Sim(s^r, s) = \omega_s \cdot Sim_{tag}(s^r, s) + (1 - \omega_s) \cdot Sim_{agg}(s^r, s) \in [0, 1] \quad (1)$$

$Sim_{tag}(s^r, s)$ is based on WordNet and has been widely described in [12],:

$$Sim_{tag}(s^r, s) = \frac{2 \cdot \sum_{t^r,t} TagAff(t^r, t)}{|\{t^r\}| + |\{t\}|} \in [0, 1] \quad (2)$$

where $TagAff(t^r, t) = 0.8^L$, if there is a path of L hyponymy/hypernymy relations between t^r and t. In the running example, $Sim_{tag}(s^r, s_3) = 0.667$ and $Sim_{tag}(s^r, s_4) = 0.862$, since $TagAff(\texttt{zipcode}, \texttt{postalcode}) = 1.0$, $TagAff(\texttt{sex}, \texttt{male}) = 0.8$, $TagAff(\texttt{sex}, \texttt{female}) = 0.8$, $TagAff(\texttt{postalcode}, \texttt{latitude}) = 0.64$, $TagAff(\texttt{postalcode}, \texttt{longitude}) = 0.64$. $Sim_{agg}(s^r, s)$ quantifies the average similarity between g^r and the aggregations where s has

been used in the past. Such similarity relies on the Dice formula and is computed as:

$$AggSim(g^r, \mathcal{S}[g]) = \frac{2 \cdot \sum_{i,j} Sim_{tag}(s_i, s_j)}{|g^r| + |\mathcal{S}[g]|} \in [0,1] \quad (3)$$

The rationale here is that the more similar are data services used in the two compared aggregations according to their tag similarity, the more similar are the two aggregations as well. For example, since g^r already contains s_1 and s_2, we have

$$Sim_{agg}(s^r, s_3) = \frac{2 * (1.0 + 1.0 + 0.667)}{3 + 3} = 0.889 \quad Sim_{agg}(s^r, s_4) = \frac{2 * (1.0 + 0.862)}{3 + 2} = 0.745 \quad (4)$$

The weight $\omega_s \in [0,1]$ is used to balance the impact of similarity based on (semantic) tags and similarity based on data service aggregations. If $g^r = \emptyset$, that is, the developer is looking for the first data service to be included in the new web application that is being designed, then $\omega_s = 1$. In the running example, $Sim(s^r, s_3) = 0.8224$ and $Sim(s^r, s_4) = 0.7801$, where $\omega_s = 0.3$. Data services included in the search results (that we denote with $\mathcal{S}' \subseteq \mathcal{S}$) are those whose overall similarity $Sim(s^r, s) \geq \tau$, where τ is set by the requester.

3.3 Data Service Ranking

Each data service $s \in \mathcal{S}'$ is ranked taking into account both its overall similarity compared to the request, $Sim(s^r, s)$, and the value assigned to s through a ranking function $\rho : \mathcal{S}' \mapsto [0,1]$, that is based on the ranking of developers in terms of number of their followers at the Social layer. In particular, the better the ranking of developers who used the data service s, the closer the value $\rho(s)$ to 1.0 (maximum value). The $Sim(s^r, s)$ and $\rho(s)$ elements are combined in the following harmonic mean:

$$rank(s) = \frac{2 \cdot \rho(s) \cdot Sim(s^r, s)}{\rho(s) + Sim(s^r, s)} \in [0,1] \quad (5)$$

In the following, we further discuss how we estimate the ranking of developers for evaluating $\rho(s)$. For simplicity, let's denote the ranking of a developer $d \in \mathcal{D}$ as $DR[d] \in [0,1]$. The value $\rho(s)$ is computed as follows:

$$\rho(s) = \frac{\sum_{k=1}^{n} DR[d_k]}{N} \in [0,1] \quad (6)$$

where $\{d_k\} \subseteq \mathcal{D}$ are the developers who used the data service s in their own web application design projects, $DR[d_k]$ is the ranking of developer d_k, N is the number of times s has been selected (under the hypothesis that a developer might use a data service s in $m > 1$ projects, then $DR[d_k]$ is considered m times). The computation of $DR[d]$, based on the number of followers of d, depends on the

type of social pattern, among the ones shown in Figure 1, as discussed in the following.

Hierarchical Social Pattern. This pattern can be represented through a tree structure, where children nodes *follow* their own parent node and only one parent is allowed for each child. Now, let's consider the structure shown in Figure 2(a), where we partially expanded the social layer of Figure 1. The developer d_c is the requester and has to choose among data services that have been used in the past by developers d_a, d_b, d_d, d_e and d_f, whose social relationships are depicted in figure. In particular, note that d_e and d_f designed the two aggregations g_1 and g_2, that contain data services s_3 and s_4, respectively.

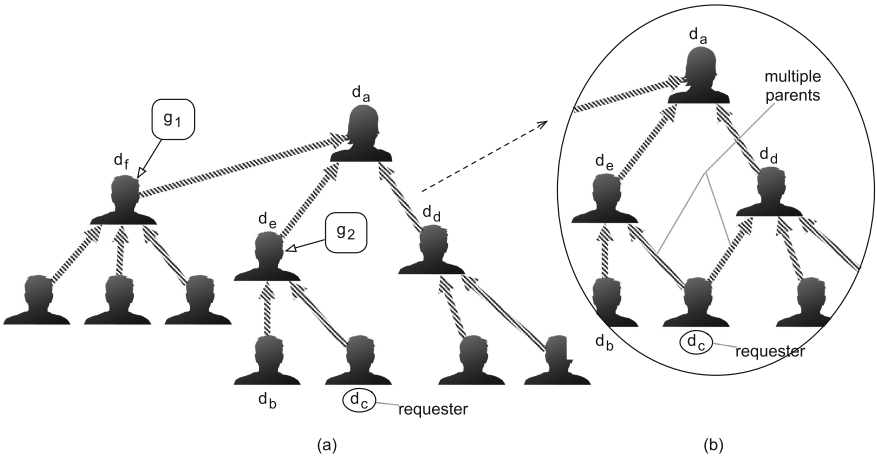

Fig. 2. An example of hierarchical social pattern (a) and a variant of the example, that represents a hybrid social pattern (b)

In this scenario, a developer d_i is ranked better than another developer d_j, under the viewpoint of a requester d, denoted with $d_i \prec_d d_j$, if one of the following conditions holds:

1. **condition (C1) -** d_i is one of the ancestors of d in the tree, while d_j is not; the rationale is that d always prefers to learn from developers, for whom d explicitly declared a *follower-of relationship*; for instance, considering the example in Figure 2(a), $d_e \prec_{d_c} d_d$;
2. **condition (C2) -** both d_i and d_j are ancestors of d in the tree, and $\ell(d, d_i) < \ell(d, d_j)$, where $\ell(d, d_i)$ denotes the distance (in terms of number of *follower-of relationships*) between d and d_i; for instance, in Figure 2(a), $d_e \prec_{d_c} d_a$, since $\ell(d_c, d_e) = 1 < \ell(d_c, d_a) = 2$, ; the rationale here is that developer d_c prefers to follow the examples of closer developers, for whom d_c explicitly declared a *follower-of relationship*.

In all the other cases, there is not a direct *follower-of relationship* or a chain of such relationships from the requester d to d_i or d_j, therefore the precedence

$d_i \prec_d d_j$ or $d_j \prec_d d_i$ is chosen, according to the number of d_i and d_j followers and in turn of the followers themselves, in a recursive way. The following formula is used, inspired by the PageRank metric:

$$w(d) = \frac{1-\alpha}{|\mathcal{D}|} + \alpha \cdot \sum_{k=1}^{n} w(d_k) \tag{7}$$

where: $w(d)$ is the weight of developer d; $|\mathcal{D}|$ is the number of developers; n is the number of developers d_k who follow d; $w(d_k)$ is the weight of developer d_k; α is the *dumping factor*, that, following the studies of the PageRank metric, has been set to 0.85, to weight the most those developers who have more followers. This metric has been adapted to the proper tree structure of the hierarchical social pattern, where only one parent is allowed for each node.

Specifically, $d_i \prec_d d_j$ if $w(d_i) > w(d_j)$, according to Equation (7). Note that the computation of $w(d_i)$ and $w(d_j)$ is independent from their relative position in the tree with respect to d, that is, in this case $d_i \prec_d d_j$ coincides with $d_i \prec d_j$. However, we maintain the former notation to indicate that we are weighting developers after a request issued by d. For instance, $w(d_f) = 0.0483$ (given that its children, who have no followers, have a weight equal to 0.0136) and $w(d_d) = 0.0367$, therefore $d_f \prec_{d_c} d_d$. If $w(d_i) = w(d_j)$, then the following further conditions are checked: (a) number of *data service aggregations* designed by d_i and d_j, namely $\mathcal{G}[d_i]$ and $\mathcal{G}[d_j]$, respectively, that is, $d_i \prec_d d_j$ if $\mathcal{G}[d_i] > \mathcal{G}[d_j]$; (b) average complexity of *data service aggregations* designed by d_i and d_j, in terms of average number of data services included within the aggregations, denoted with $\hat{\mathcal{G}}[d_i]$ and $\hat{\mathcal{G}}[d_j]$, respectively. We compute $\hat{\mathcal{G}}[d_i]$ as:

$$\hat{\mathcal{G}}[d_i] = \frac{\sum_{g \in \mathcal{G}[d_i]} |\mathcal{S}[g]|}{|\mathcal{G}[d_i]|} \tag{8}$$

where $|\mathcal{S}[g]|$ denotes the number of data services in g and $|\mathcal{G}[d_i]|$ denotes the number of aggregations in $\mathcal{G}[d_i]$. Therefore $d_i \prec_d d_j$ if $\hat{\mathcal{G}}[d_i] > \hat{\mathcal{G}}[d_j]$. For the example shown in Figure 2(a), the final ranking is: $d_e \prec_{d_c} d_a \prec_{d_c} d_f \prec_{d_c} d_d \prec_{d_c} d_b$. Note that, $d_e \prec_{d_c} d_f$ even if $w(d_e) < w(d_f)$, since in our approach our aim is at giving more importance to the developers among which *follower-of relationships* have been explicitly declared. The final developer's ranking $DR[d]$ used in Equation (6), is computed as:

$$DR[d] = \frac{N - (pos(d) - 1)}{N} \in [0, 1] \tag{9}$$

where $pos(d)$ represents the position of d in the ranking and N is the total number of ranked developers. For example, $DR[d_e] = (5-0)/5 = 1$, $DR[d_f] = (5-2)/5 = 0.6$. Therefore, according to Equation (6), $\rho(s_3) = 1$ and $\rho(s_4) = 0.6$, that is, $rank(s_3) = 1.556/1.778 = 0.875$ and $rank(s_4) = 0.9688/1.404 = 0.687$, according to Equation (5).

Peer-Based Social Pattern. This pattern can be represented through a graph, where there is not a hierarchical structure to be recognized. Therefore, conditions **C1** and **C2** meant for the hierarchical social pattern are not applicable, due to

the possible presence of cycles. In this case, we use the following equation to weight developers:

$$w(d) = \frac{1-\alpha}{|\mathcal{D}|} + \alpha \cdot \sum_{k=1}^{n} \frac{w(d_k)}{C(d_k)} \tag{10}$$

The equation is a slight variation compared to Equation (7): $w(d)$ is the weight of developers d; $|\mathcal{D}|$ is the number of developers; n is the number of developers d_k who follow d; $w(d_k)$ is the weight of developer d_k; α is the *dumping factor* (set to 0.85); $C(d_k)$ is the number of developers followed by d_k, where, in this case, $C(d_k)$ can be greater that 1, while in the case of hierarchical pattern $C(d_k)$ can be at most equal to 1, due to the constraint on the single parent node. If $w(d_i) = w(d_j)$, then the following further conditions are checked: (a) number of *data service aggregations* designed by d_i and d_j; (b) average complexity of *data service aggregations* designed by d_i and d_j, in terms of average number of data services included within the aggregations. The final developer's ranking $DR[d]$ is computed according to Equation (9).

Hybrid Social Pattern. This pattern can be represented through a tree structure, where children nodes *follow* their own parent node and also multiple parents are allowed. Therefore, conditions **(C1)** and **(C2)** still hold. Now, let's consider the structure shown in Figure 2(b), where a new *follower-of relationship* from d_c to d_d has been added. In the scenario depicted in Figure 2(b), conditions **(C1)** and **(C2)** are not satisfied, because, for the developer d_c, both d_e and d_d are ancestors, but $\ell(d_c, d_e) = \ell(d_c, d_d)$. Therefore, we add the following additional condition:

3 **condition (C3) -** both d_i and d_j are ancestors of d in the tree, $\ell(d, d_i) = \ell(d, d_j)$, where $\ell(d, d_i)$ denotes the distance (in terms of number of *follower-of relationships*) between d and d_i, and $w(d_i) > w(d_j)$, where $w(d)$ is computed as shown in Equation (10); for instance, in Figure 2(b), $d_d \prec_{d_c} d_e$, since $\ell(d_c, d_e) = 1 = \ell(d_c, d_a)$, but $w(d_e) = 0.0309$ and $w(d_d) = 0.0425$.

In all the other cases, there is not a direct *follower-of relationship* or a chain of such relationships from the requester d to d_i or d_j, therefore the final developer's ranking $DR[d]$ is computed as for the peer-based social pattern. Preliminary laboratory experiments are being performed to test the effectiveness of our approach, on a proper dataset that is compliant with the model depicted in Figure 1. First experiments show how our system presents better results in the first positions of the search outcome. As expected, considering only $Sim_{tag}()$ values for selecting relevant data services is not effective, since the use of a specific data service for a particular web application cannot be inferred by only inspecting tags used to classify the service; instead, also Sim_{agg} values and ranking based on social patterns should be exploited to this purpose. This has been investigated by varying the ω_s parameter. In the first experiments, we are obtaining satisfying results for $\omega_s = 0.3$ (see Equation (1)): this means that in the dataset considered in the experiments, the importance of past developers' experiences in

using candidate data services is even greater than the information coming from data service (semantic) tagging.

4 Conclusions

In this paper, we proposed a multi-layered model and proper metrics relying on it, meant for supporting the selection of data services for web application design purposes, taking into account also past experiences of other developers in collecting and aggregating data services in similar contexts, i.e., to build similar applications. Moreover, we modeled the network of social relationships between developers as third layer of the model, in order to exploit it for estimating the importance that a developer assigns to past experience of other developers. Further information might be integrated within the model, such as quality features among metadata and automatic annotation of service interfaces at the data service and web application layers, to refine the selection step. Extension of the approach to different contexts, where high credibility of developers cannot be assumed, will require the introduction of additional social patterns and the integration of credibility estimation techniques, such as the ones described in [7–9]. Finally, extensive experimentation (also considering real cases of web application design) has to be completed.

References

1. Li, Y., Wang, Y., Du, J.: E-FFC: an enhanced form-focused crawler for domain-specific Deep Web databases. J. of Intelligent Information Systems **40**(1), 159–184 (2013)
2. Quarteroni, S., Brambilla, M., Ceri, S.: A Bottom-up, Knowledge-Aware Approach to Integrating and Querying Web Data Services. ACM Trans. on the Web **7**(4), 44–76 (2013)
3. Bozzon, A., Brambilla, M., Ceri, S., Mazza, D.: Exploratory Search Framework for Web Data Services. VLDB Journal **22**, 641–663 (2013)
4. Bianchini, D., De Antonellis, V., Melchiori, M.: QoS in ontology-based service classification and discovery. In: 15th International Workshop on Database and Expert Systems Applications, pp. 145–150. IEEE Computer Society, Los Alamitos (2004)
5. Dillon, S., Stahl, F., Vossen, G.: Towards the web in your pocket: Curated data as a service. In: Nguyen, N.T., Trawinski, B., Katarzyniak, R., Jo, G.S. (eds.) Advanced Methods for Computing Collective Intelligence, pp. 25–34. Springer, Berlin Heidelberg (2013)
6. Balakrishnan, R., Kambhampati, S., Manishkumar, J.: Assessing Relevance and Trust of the Deep Web Sources and Results Based on Inter-Source Agreement. ACM Trans. on the Web **7**(2), 32 (2013)
7. Al-Sharawneh, J., Williams, M., Wang, X., Goldbaum, D.: Mitigating risk in web-based social network service selection: follow the leader. In: 6th Int. Conference on Internet and Web Applications and Services, pp. 156–164. IARIA XPS Press (2011)

8. Malik, Z., Bouguettaya, A.: RATEWeb: Reputation Assessment for Trust Establishment among Web Services. VLBD Journal **18**, 885–911 (2009)
9. Bianchini, D., De Antonellis, V., Melchiori, M.: Capitalizing the designers' experience for improving web API selection. In: Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T. (eds.) OTM 2014. LNCS, vol. 8841, pp. 364–381. Springer, Heidelberg (2014)
10. Fuxman, A., Giorgini, P., Kolp, M., Mylopoulos, J.: Information systems as social structures. In: 2nd Int. Conf. on Formal Ontologies for Information Systems, pp. 12–21. ACM, New York (2001)
11. Gupta, V., Lehal, G.: A Survey of Text Mining Techniques and Applications. J. of Emerging Technologies in Web Intelligence **1**(1), 60–76 (2009)
12. 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)
13. dos Santos, T., de Araujo, R., Magdaleno, A.: Identifying Collaboration Patterns in Software Development Social Networks. J. of Computer Science, 51–60 (2010)