Social Discovery and Composition of Web Services

Abderrahmane Maaradji[1,2], Hakim Hacid[1], Ryan Skraba[1], Adnan Lateef[1], Johann Daigremont[1], Noël Crespi[2]

[1] Alcatel-Lucent Bell Labs France, Route de Villejust, 91620 Nozay, France
{firstname.lastname}@alcatel-lucent.com
[2] Telecom SudParis, 9 Rue Charles Fourier, 91000 Evry, France
{firstname.lastname}@it-sudparis.eu

Abstract. In this paper, we propose a new approach for services recommendation to assist services composition in a Mashup environment capturing and analyzing social interactions. This approach uses an implicit social graph inferred from the common composition interests of users. We describe in detail the transformation of users-services interactions into a social graph and a possible means to leverage that graph to derive service recommendation. This proposal was implemented within a platform called *SoCo* and the experiments show interesting results.

1 Introduction

By providing building blocks of data provisioning and transformation, Web services composition makes it possible to create new and unexpected services based on existing services one of the reasons why these compositions have proven so successful. This has been an area of great interest and research from both academic and industrial perspectives [1][2]. Current efforts aim to improve these techniques by addressing issues such as selecting the appropriate relevant services among those available to best reach a user's goal in what we call semi-automatic approach. This approach guide users in the composition process, enabling their explicit participation in constructing powerful new services without overwhelming them with details, such as Mashups. Besides, this approach aims at exploiting the information they have generated implicitly, such as their personal user data or the data generated by their community. This is especially true in an business context where community knowledge may be used.

With the emergence of collaborative environments, service composition has become relevant to environments outside the enterprise. The Web has become a place where users can create and share data and user-generated content (UGC), as well as the means to access and use this information by means of building blocks that can be combined. We believe that service composition is poised to become increasingly important and ubiquitous. Moreover, with the popularity of social networks, this ongoing collaboration has taken on a daily importance in users' social lives and tasks. In this context, how can Web 2.0 and social networks be leveraged to provide a socially-aware Mashup creation environment for end users?

One of the possibilities that has emerged as a technique for semi-automatic service composition is a new way to dynamically recommend services, thereby supporting composite service combination in a Mashup environment. We consider the problem of Mashup creation and propose a contribution from the perspective of social network analysis. The approach is based on the implicit construction of a social network from data generated in a Mashup environment according to common interests shared by users and the services they interact with. Service recommendations are then calculated on top of this social network to help end users to build and complete their Mashup service(s).

The rest of this paper is organized as follows: Section 2 gives an overview of existing and related work to our research work. Section 3 presents our contribution to dynamic services recommendation in Mashup environments. In Section 4, we show experimental results regarding the performances of the proposed approach. Finally, Section 5 concludes the paper and discusses the algorithm and its possible improvements.

2 Related work

A Mashup is certainly the most interesting instantiation of end users' services composition. A Mashup is a Web application that combines existing services (API, data sources, etc.) into a single integrated service. An example could be the use of cartographic data and interfaces from Google Maps[3] to add location information about real estate data [3]. This creates a new and distinct Web service. Just as Web2.0 technologies enable the well-known concept of *User Generated Content* (UGC), Mashup creation environments enable *User Generated Services* (UGS). Semi-automatic services composition systems, and particularly Mashups, which involve users, contain an unexploited repository of information. Indeed, the different types of interactions between entities involved in semi-automatic composition, i.e., users and services, could feed many systems.

Furthermore, the explosion of the number of Web services and APIs exposed through the Web has only accentuated the need for such service composition platforms. Dynamic service recommendation could be a solution to enable service discovery and composition (our proposed method). Through a use-case approach, Floyd et al. [4] highlight the proliferation of APIs on the Web in parallel with the number of creative Web users. Their study shows the multiple benefits of collaboration between end users and developers that combines the innovation and creativity of end users with the expertise of developers. Automating this process is the important challenge we address here. In the same vein, an interesting study [5] describes the interactions of Yahoo!-Pipes users and how they can be used to extract social structures based on an analysis of user interactions. In addition, those users interact with services via the Mashups they themselves create.

[3] http://maps.google.com/

Soriano et al. [6] emphasize the growing importance of the user-service relationship in a Service Oriented Architecture (SOA) for composing services. These authors introduce EZWeb, an environment for sharing Mashups between colleagues, as a basis for co-production in an enterprise context. In addition, [7] emphasizes the phenomenon of what they call "social interaction" between services. In fact, the aspects of trust and reliability between services may indeed impact the service selection for composition. Yu and Woodard [8] propose a very interesting view of Mashup ecosystems. Their study of an API repository[4] shows that utilization of services follows a long-tail effect (power-law distribution), one of the principle and most interesting properties of social networks [9].

From the end users' perspective, the semi-automatic composition of Web services generally takes the form of Mashups. The major Web players (such as Microsoft, Yahoo!, IBM, etc.) each offer environments to allow Web users to create new services. In these environments, users can discover exposed services, and create their own services as needed by combining and composing existing ones. To assist users in the editing phase of a new service (the semi-automatic composition process), current environments offer a multitude of features. These features facilitate the services discovery process by operating at the service description level (abstraction, categorization, etc.) or by providing search tools.

From the services recommendation perspective, some systems are based on user preferences (user profile) to suggest services [10]. Others rely instead on the concept of domain-specific knowledge. Knowledge (also called language) expressed in a specific area (science, business, etc.) is processed to extract rules that are used to help build services' recommendations [11]. In addition, another team [12] has proposed a community-based approach by recommending popular composition schema.

Our proposal is focused on extracting useful information from social networks to feed a service recommendation construction and assistance feature. Indeed, we believe that we must distinguish between two approaches: one based on communities and the other based on social networks. By definition, a community develops social links between people on the basis of a single common interest (e.g., free software community, medical community, etc.) while a social network develops interesting interactions based on a variety of particular interests. In our approach, we introduce recommendations based on the knowledge generated in social networks through implicit social interaction analysis.

3 A SNA based Mashups auto-completion

In Web 2.0, people can create, use, and share services in communities and social networks. The question we are addressing is: *can we define social structures in the services area? If yes, how could we leverage such structures to facilitate the resolution of the problems involved in SOA?* To resolve this problem statement and approach the related work, we propose a general framework, called *Social*

[4] http://programmableweb.com

Composer (SoCo), for services discovery and composition. *SoCo* is based on the transformation of *user* $\rightarrow$ *services* interactions to *user* $\rightarrow$ *users* social network, on top of which statistical processes are applied to determine recommendations for assisting a user in constructing a Mashup (a composition of services). Before detailing our proposal, we clearly define the notion of social network in our context, which is an abstraction of interactions that occur between people and services in Web service environments in the form of a social graph translating the behavior of social entities, i.e., users. This structure may be inferred or extracted directly from common interests between the users of the composition platform.

The explicit social relation has been addressed in our previous work [13] and we focus here on the implicit case, which is richer and enables more sophisticated tasks. An implicit social relation is one that is inferred according to the activities of different users, e.g. when one person uses several services created by another person. As in the explicit case, a graph can be developed, linking the users according to their interests and defined by their composition activities. Social networks, as they are currently constructed, include a profile for each user containing information that describes his/her special interests and preferences, and the history of his/her interactions with the system (i.e., a dynamic profile). Typically, these include statistics on services utilization (consumption and composition). This information allows the system to learn about the expertise of a given person in a particular area, and thus the relevance of services used by that person. A social network also includes a description of the links that define the social network itself. These links are used to calculate the social proximity between two users according to a particular context. This information then allows us to calculate the service recommendation confidence between two individuals based on specific joint interests. To leverage this information, it is therefore very important to extract, analyze and model the information contained in a social network.

Information regarding users' interactions is used by the recommendation step that helps *SoCo*'s users during the creation of new Mashups, which dynamically suggests services according to the current status of the services composition process, i.e. which service is likely to come after the currently selected service? Thus, it intervenes in the services selection process. This recommendation logic considers different parameters to determine the user's position within their social network, and evaluates the information on the use of services by the social neighbors. To interpret and leverage social interactions in a Mashup environment, one needs to process *users* $\rightarrow$ *services* interactions into a social graph between users. In fact, we have two sets of distinct inputs: users and services. There may very well be several techniques for processing these interactions. We describe our approach in the following section.

First, we consider *users* $\rightarrow$ *services* interactions as a bipartite graph [14] which represents the usage rate that users have on the services they use. It should be explicitly stated that the bipartite graph is based on the usage frequency of services by users in composition schemes, with no account being taken of services succession in those schemes. Figure 1 illustrates a bipartite graph of services and users. The links represent the usage frequency, denoted by $f(u_i, s_j)$, that a user u_i makes of a service s_j in all the compositions she created. To transform the bipartite graph into a social graph for successor recommendation we rely on three main steps: (i) local information extraction, (ii) semi-global information extraction, and (iii) global information extraction.

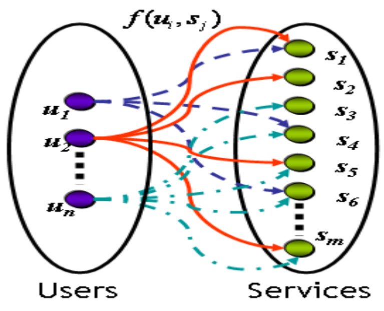


Fig. 1. Illustration of a bipartite graph between users and services

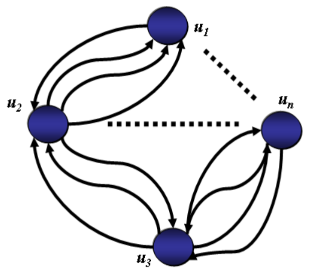


Fig. 2. Illustration of the output after applying a semi-global information level

3.1 Local information extraction

The *local information* category only considers the interaction between a specific user and a specific service. This means that a user and a service are considered independently of the other users and services of a system. This information tells us whether a specific user is confident (i.e., an expertise indicator) using this service among others. The more a given user is confident about his/her usage of a given service, the more the recommendation of this service matters. To materialize this idea, we define this information in a quantity called *Activity* defined in equation 1 where M is the total of service a user exploited in her different compositions.

3.2 Semi-global information extraction

This step is important in the transformation of a bipartite graph to a social graph (directed graph). Once this step is applied on the bipartite graph, we obtain several multi-layered directed graphs corresponding to the number of services in the system. This is illustrated in Figure 2. In other terms, in the level of semi-global information, we consider the interest a user may have in other users in regards to a given service. Thus, for a given user u_i we calculate how much the service s_j, recommended by the user u_l matters to her. This is called *Special Interest* (SI) and is calculated using equation 2.

$$Act(u_i, s_j) = \frac{f(u_i, s_j)}{\sum_{k=1}^{M} f(u_i, s_k)} \quad (1) \qquad SI(u_i, u_l, s_j) = \frac{f(u_l, s_j)}{f(u_i, s_j)} \quad (2)$$

3.3 Global Information extraction

In order to have as precise a transformation as possible with the least data loss, we add another level of information in the transformation process. Global information extraction determines whether a pair of users share a common general interest or not. At this stage of our study, and for simplification reasons, we consider that the general interest of a pair of users is equal to the sum of their specific interests, thus building the implicit graph as illustrated in equation 3. The output of this step is a graph that aggregates all the specific interest graphs obtained previously.

$$IG(u_i, u_l) = \sum_{k=1}^{M} SI(u_i, u_l, s_k) \tag{3}$$

3.4 Services recommendation strategy

Once the bipartite graph is transformed into a social graph thanks to the three steps described above, we proceed to the recommendation calculation wherein a newer service is suggested, based the predefined metrics. Thus, considering the intrinsic user's usage frequency (local information), the specific interest between two users (semi-global information), and the implicit graph which expresses the global interest between the two users, we can define the recommendation confidence of a given service s_k with respect to a current service s_j for user u_i as follows:

$$RC_{imp}(u_i, s_j, s_k) = \sum_{l=1}^{N} SI(u_i, u_l, s_k) \times Act(u_l, s_k) \times IG(u_i, u_l) \tag{4}$$

Algorithm 1 summarizes the recommendation process. Given a configuration (u_i, s_j) where u_i represents the current user and s_j the selected service, the algorithm returns Rec_{List}, a sorted list of recommended services that are socially relevant to be successor of s_j for user u_i. It should be noted that since the output of the algorithm is a list and the selection follows a $Top - K$ principle (K services), the values are not necessarily inside the interval $[0, 1]$ even though this could be easily integrated, e.g. by maintaining the maximum value of the recommendation to normalize the output.

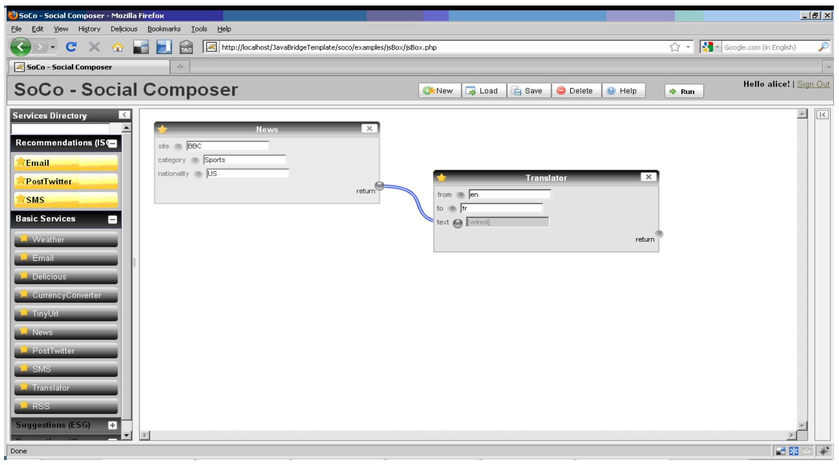
Given a set of services S with $|S| = M$, and a set a users U with $|U| = N$, the theoretical complexity of Algorithm 1 is $N \times M$. It makes one scan for potential successors of the current service (with M size at worst case). For each potential successor, the algorithm scans the current user's neighbour that is of potential interest (with N size at worst case). Notice that Act, SI, and IG are updated incrementally (one data access) when usage activity occurs.

Algorithm 1 The Recommendation algorithm for semi-automatic services composition

```
input $u_i \in U$, $s_j \in S$
for each $s_k$ where $serv_i \rightarrow serv_k$ exists do
    $RC_K = 0$
    for each $u_l$, where $u_l$ is neighbor of $u_i$ in a specific graph of $s_k$ do
        $RC_K = RC_K + Act(u_k, s_k) \times SI(u_i, u_k, s_k) \times IG(u_k, s_k)$
    end for
    Add ($s_k$,$RC_k$) in $Rec_{List}$
end for
Sort $Rec_{List}$ in descending order of $RC_k$
output $Rec_{List} = \{(s_k, RC_K)\}$
```

4 Experimental results

In this section, we evaluate the performance and the behaviour of the algorithm regarding different parameters. To implement the proposed algorithm, we have used the *SoCo* framework introduced in [15]. This framework provides a graphical environment that users utilize to create Mashups. It includes the recommendation algorithm to provide dynamic suggestions to users. Figure 3 shows the basic services list on the left side, and above it the suggested services list.



site: BBC
category: Sports
nationality: US
from: en
to: fr
text: [wired]

Fig. 3. Illustration of services suggestion (in yellow) after selecting translator service

Based on Web client/server architecture, SoCo captures user-service interactions on the client side; this information is stored and then modeled to supply the recommendation system (algorithm) to the server side. During the editing process, the user receives dynamic recommendations computed by the algorithm.

Our implementation used the PHP programming language, the MySQL server database, and an Apache server (Intel Core 2 Duo P8600 machine with 2.4GHz and 2GB RAM). As the first step of our evaluation, we considered the response time (i.e., the time required for a successor of a service to be recommended) as a major performance indicator. We calculated how long the algorithm takes to respond to a query for a given configuration. To ensure the system's interactivity with the user it is naturally imperative that the algorithm responds quickly to queries. For this evaluation, the following parameters were used: (i) The size of Mashups, represented by the number of services that make up those Mashups; (ii) The overall number of services $|S|$ or Mashups $|M|$ stored in the system. The number of Mashups is generally proportional to from one to three times the number of services according to observations of *ProgrammableWeb* ($|M| = 3 \times |S|$);(iii) The number of users in the system $|U|$.

Another performance parameter suitable to be considered but that we do not include here is the size of the recommendation list. In fact, the proposed algorithm does not consider this information because it calculates the recommendation confidence level for all potential successors to the current service.

Given the three performance parameters defined above, we conducted three experiments. For each experiment, we fixed two parameters and varied the third. Due to the lack of benchmarks, the data sets were generated randomly. More concretely, services that compose a Mashup are independently and uniformly selected. A less-important parameter, representing the creation of a Mashup by a user, is also a randomly generated relation. We ran each experiment 25 times: five times for five randomly generated pairings of *(u_i: current user, s_j: current service)*, and each point in the curves thus generated shows the average.

To measure the impact of Mashup size on the algorithm performance, we began by fixing the number of users $|U| = 15 \times 10^2$ and the number of services $|S| = 1740$. We varied the size of the Mashups (in terms of services composing each Mashup) from two to seven services. Figure 4 shows the impact of Mashup size on the behaviour of the algorithm. Generally speaking, the size of a Mashup has a tangible impact on the algorithm which is observable in the associated response times for recommending a service, and that the algorithm response time increases linearly with respect to the increasing Mashup size (even though the curve takeoff is initially not quite linear).

As a second experiment, we varied the number of services $|S|$ in the system. We fixed the number of users $|U| = 10^4$ and the size of Mashups was uniformly distributed between two and five. Varying the number of services has an impact on the total number of Mashups. Figure 5 shows the results. It is clear that the algorithm's change in response time is not exponential, but it could be approximated by a linear regression.

For the third experiment, we evaluated the impact of the number of users on the algorithm's performance. We set (i) the number of services $|S| = 10^4$ (the number of Mashups $|M| = 30 \times 10^3$), (ii)the size of Mashups is uniformly distributed between two and five, and we vary the number of users $|U|$. Figure 6 shows the results and indicates where the algorithm response time is stable even when the number of users increases up to $|U| = 10^5$.

Other simulations have been performed with more realistic statistical distributions impacting the behaviour of the algorithm. Indeed, as pointed out above, several studies [8] have shown that the popularity of services used for Mashup creation follows a long-tail distribution, meaning that some services are much more frequently used than other services For example, mapping services are the they type most used for Mashups. We also note that the graphs representing links between users are social networks, and include special features such as small-world, power-law degree distribution, and additional information which may be leveraged to better evaluate a system. Concretely, we have generated datasets following a long-tail distribution (Zipf's law) that were used for the input to the recommendation algorithm. The results show that the recommendation algorithm response time decreases compared to the response time for uniformly distributed datasets(see Figure 7).

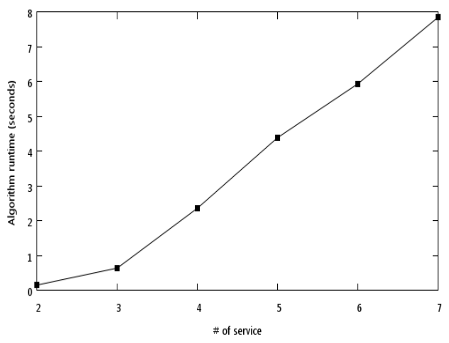


Fig. 4. Mashup size impact on algorithm execution time

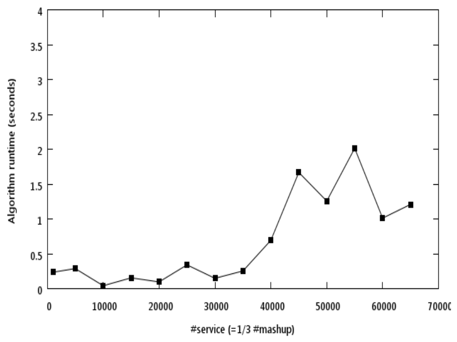


Fig. 5. Impact of the number of services on the performances of the algorithm

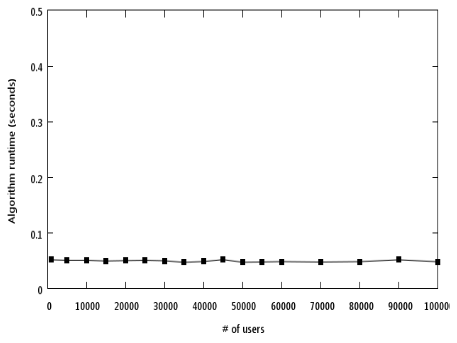


Fig. 6. Impact of the number of users on the the performances of the algorithm

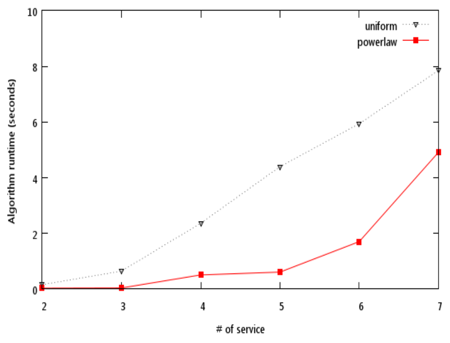


Fig. 7. The long-tail distribution impact on algorithm performance

Overall, these results are interesting and do show that this algorithm is useful in an interactive application. However, they also show that the algorithm's response time is particularly sensitive to Mashup size, which is, according to related studies and our observations on *ProgrammableWeb*, distributed between two and five services (mostly closer to two). From a more general perspective, we believe that there is still room for improvement by optimizing the recommendation strategy.

5 Conclusion

We have presented a new approach for services recommendation in a Mashup environment based on social network analysis. The particularity of this approach is the creation of an implicit social graph based on the interactions between users and services. We have described all the steps we utilized to generate this social graph and a possible way to leverage it for services recommendation. While the experimental results, implemented in a platform called *SoCo*, show the positive potential of the proposed approach, there are still some challenging issues to be addressed as future work: (i) the newcomers' problem, which deals with the algorithm behaviour regarding the lack of learning information, and (ii) the entire Mashup auto-completion [12] format, which has the goal of suggesting a complete composition schema.

References

1. M. H. ter Beek, A. Bucchiarone, and S. Gnesi, "Web service composition approaches: From industrial standards to formal methods," in *ICIW*, 2007, pp. 15–20.
2. Y. Yuan, J. Wen, W. Li, and B. Zhang, "A Comparison of Three Programming Models for Telecom Service Composition," in *AICT*. IEEE Computer Society Washington, DC, USA, 2007.
3. R. Ennals and M. N. Garofalakis, "Mashmaker: mashups for the masses," in *SIGMOD Conference*, 2007, pp. 1116–1118.
4. I. R. Floyd, M. C. Jones, D. Rathi, and M. B. Twidale, "Web mash-ups and patchwork prototyping: User-driven technological innovation with web 2.0 and open source software," in *HICSS '07*. Washington, DC, USA: IEEE Computer Society, 2007, p. 86.
5. M. C. Jones and E. F. Churchill, "Conversations in developer communities: a preliminary analysis of the yahoo! pipes community," in *CCT '09*. New York, NY, USA: ACM, 2009, pp. 195–204.
6. J. Soriano, D. Lizcano, J. J. Hierro, M. Reyes, C. Schroth, and T. Janner, "Enhancing user-service interaction through a global user-centric approach to soa," in *ICNS '08*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 194–203.
7. Z. Maamar, L. Wives, Y. Badr, and S. Elnaffar, "Even Web Services Can Socialize: A New Service-Oriented Social Networking Model," in *INCOS'09*, 2009, pp. 24–30.
8. S. Yu and C. J. Woodard, "Innovation in the programmable web: Characterizing the mashup ecosystem," pp. 136–147, 2009.
9. S. Wasserman and K. Faust, *Social network analysis: Methods and applications.* Cambridge Univ Pr, 1994.

10. T. Law, "Social Scripting for the Web," *Computer*, vol. 40, no. 6, pp. 96–98, 2007.
11. L. Chen, N. Shadbolt, C. Goble, F. Tao, S. Cox, C. Puleston, and P. Smart, "Towards a knowledge-based approach to semantic service composition," *Lecture Notes in Computer Science*, pp. 319–334, 2003.
12. O. Greenshpan, T. Milo, and N. Polyzotis, "Autocompletion for mashups," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 538–549, 2009.
13. A. Maaradji, H. Hacid, J. Daigremont, and N. Crespi, "Towards a social network based approach for services composition," in *IEEE ICC'10.*, may 2010.
14. J. Guillaume and M. Latapy, "Bipartite structure of all complex networks," *Information processing letters*, vol. 90, no. 5, pp. 215–221, 2004.
15. A. Maaradji, H. Hacid, J. Daigremont, and N. Crespi, "Social composer: A social-aware mashup creation environment," in *ACM CSCW '10 (Demos session)*, 2010.