Towards a Method for Engineering Social Web Services

Zakaria Maamar[1], Noura Faci[2], Leandro Krug Wives[3], Hamdi Yahyaoui[4], and Hakim Hacid[5]

[1]Zayed University, Dubai, U.A.E.
zakaria.maamar@zu.ac.ae
[2]Claude Bernard Lyon 1 University, Lyon, France
noura.faci@liris.cnrs.fr
[3]Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
wives@inf.ufrgs.br
[4]Kuwait University, Safat, State of Kuwait
hamdi@sci.kuniv.edu.kw
[5]Alcatel-Lucent Bell Labs, Paris, France
hakim.hacid@alcatel-lucent.com

Abstract. This paper motivates the blend of social computing with service-oriented computing, giving "birth" to social Web services. On the one hand, social computing builds user applications upon the principles of collective action and content sharing. On the other hand, service-oriented computing builds enterprise applications upon the principles of service offer and demand and loose coupling. Thanks to this blend social Web services can operate taking into account with whom they worked in the past and with whom they would like to work in the future. To engineer social Web services, this paper presents a four-step method that addresses several questions related to the engineering exercise. These questions are what relationships exist between Web services, what social networks correspond to these relationships, how to build social networks of Web services, and what social behaviors can Web services exhibit. Experiences dealing with implementing social Web services are, also, reported in the paper.

Keywords: Engineering, Service-oriented computing, Social computing, Web service.

1 Introduction

It is largely known that those responsible for designing and developing enterprise applications appreciate greatly the use of engineering methods while completing their duties. Indeed these methods are road maps that indicate among other things the steps to carry out, the notations to use, the meetings to schedule, and the deliverables to turn in. With the increasing complexity and diversity of today's enterprise applications, different engineering methods (e.g., situational and domain-specific) and scientific fora (e.g., ME'11) have been set up.

J. Ralyté, I. Mirbel, and R. Deneckère (Eds.): ME 2011, IFIP AICT 351, pp. 153–167, 2011.

The purpose of these methods and fora is to keep up the pace with the challenges that these applications pose on enterprise applications designers and developers.

Service-Oriented Architecture (SOA) paradigm and its flagship implementation technology namely Web services are among the latest trends in enterprise applications design and development. According to Engels et al., SOA promotes the separation of concerns, information hiding, strong cohesion, and loose coupling [7]. The compliance with SOA principles results in business processes that are flexible and capable of crossing organization boundaries. A Web service is *"a software application identified by a URI, whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts and supports direct interactions with other software applications using XML-based messages via Internet-based applications"* (3WC). When necessary Web services are put together to offer new added-value composite services to users. Different methods and approaches to engineer Web services based-enterprise applications are reported in the literature [10,11,16]. For example, Foster et al. use a model-based approach to verify Web services composition interactions for a coordinated SOA [10]. The adoption of Web services means the ability to support early verification of service implementations against design specifications and that compositions are built with compatible interfaces. Maamar et al. adopt goals to engineer a specific type of Web services that they referred to as capacity-driven Web services [16]. The goals are established to define the roles that capacity-driven Web services play when implementing business applications, frame the requirements that will be put on capacity-driven Web services, and identify the processes in term of business logic that capacity-driven Web services will carry out.

Recently, we started looking into Web services from a social perspective [14]. The purpose is to address some obstacles such as discovery and high-availability that still hinder the widespread acceptance of Web services by IT practioners. By imposing a social perspective on how Web services need to be handled at design- and run-times, we could make Web services (using appropriate tools) establish networks of contacts (i.e., peers) with whom these Web services "feel comfortable" for example to work on common compositions and to recommend for compositions. In this paper, we present our method to engineer $\mathcal{S}$ocial $\mathcal{W}$eb $\mathcal{S}$ervices ($\mathcal{SWS}s$). Briefly this method proceeds as follows: it identifies possible relationships between Web services, builds social networks out of these relationships, and finally, labels Web services as per their roles in these social networks. In this paper the social perspective refers to the social relationships that people come across daily and can be mapped onto relationships linking Web services.

This paper is organized as follows. Section 1 motivates the importance of engineering methods with focus on $\mathcal{SWS}s$ as a case study. Section 2 introduces a running scenario, discusses the overlap between social and service-oriented computing, and provides a brief literature review of $\mathcal{SWS}s$. Section 3 introduces our engineering method. Prior to concluding in Section 5, some technical details on $\mathcal{SWS}s$ are presented.

2 Background

We start with a running scenario that reveals the potential relationships between Web services. Then we introduce the disciplines of social computing and service-oriented computing and how both overlap giving "birth" to $\mathcal{SWS}s$. Finally we review some research works on $\mathcal{SWS}s$.

2.1 Running Scenario

We illustrate the use of social networks of Web services with a scenario related to purchase orders. A customer places an order for a variety of products via *CustomerWS*. Based on this order, *CustomerWS* obtains details on the customer's purchase history from *CRMWS* (Customer Relationship Management). Then, *CustomerWS* forwards these details to *BillingWS* to calculate the customer's bill and subsequently send the bill to *CRMWS*. This latter prepares the detailed purchase order and sends it to *InventoryWS* for order completion. For the in-stock products, *InventoryWS* sends a shipment request to *ShipperWS* to deliver the products to the customer. For the out-of-stock products, *InventoryWS* sends a supply request to the selected *SupplierWS*, which provides *ShipperWS* with the products for subsequent shipments to the customer.

Traditionally the aforementioned Web services (e.g., *CustomerWS*, *ShipperWS*) are discovered and selected without considering or even acknowledging the interactions they had with other peers in previous compositions. Such interactions would have been useful if captured properly using for instance social networks. During the preparation of an order, different relationships are established. The first relationship is the selection that led into identifying *CustomerWS* instead of another Web service for example *OrderSubmissionWS*. Both Web services compete against each other since they do the same job, which is handling customers' online orders. The second relationship is the dependencies between Web services that can be recurrent. *InventoryWS* and *ShipperWS* have constantly participated in several joint compositions. Finally, the third relationship is the high availability of Web services. When *ShipperWS* fails, *DeliveryWS* takes over automatically. If all these relationships had to be captured properly, a Web service would:

- recommend the peers that it likes to **collaborate** with in case of compositions, e.g., *InventoryWS* and *ShipperWS*;
- recommend the peers that can **substitute** for it in case of failure, e.g., *ShipperWS* and *DeliveryWS*;
- and, be aware of the peers that **compete** against it in case of selection, e.g., *CustomerWS* and *OrderSubmissionWS*.

Collaboration, substitution, and competition are relationships commonly found in people's daily life.

2.2 Social Computing Meets Service-Oriented Computing

With the emergence of Web 2.0, social computing is nowadays a hot discussion topic. Some well-known social applications like FaceBook and Twitter exemplify

the successful embracement of Web 2.0. Plus different case studies look into this topic ranging from social computing importance and challenges it raises like privacy to the benefits of adopting social applications by organizations [2].

In [8] social computing is related to applications that support collaborative work (GroupWare) and techniques for modeling, simulating, studying, and analyzing the society (i.e., study the social behavior). Examples of applications include on-line communities and tools, and interactive entertainment and training. Another definition sees social computing as an emerging paradigm that involves a multi-disciplinary approach for analyzing and modeling social behaviors on different media and platforms to produce intelligent applications [12]. Main characteristics of social computing are connectivity, collaboration, and community.

Service-oriented computing is, also, another discipline that attracts the attention of academics and industry people. It aims at bridging the gap between business services and IT services. As stated earlier, SOA and Web services have enabled a new wave of business processes that are loosely-coupled and can cross organization boundaries. The trend of offering business services through the Internet in the form of software services [19] allows the expansion of business services into global markets, ease of access for customers, and increased productivity for companies.

Would there be any overlap between social computing and service-oriented computing? Our answer is affirmative. On the one hand, social computing builds applications upon the principles of collective action and content sharing. On the other hand, service-oriented computing builds applications upon the principle of "I offer services that somebody else may need" and "I require services that somebody else may offer". Service offer and demand illustrate perfectly how people behave in today's society imposing a social dimension on the analysis of Web services. $\mathcal{SWS}s$ can capitalize on their experiences to "identify" with whom they worked in the past and with whom they would like to work in the future.

2.3 Social Web Services in the Literature

$\mathcal{SWS}s$ are at the cross road of service-oriented computing and social computing. Our literature review concluded to a lack of studies that address specific questions related to $\mathcal{SWS}s$ engineering such as how to identify the interactions (or relationships) between Web services and between users and Web services, how to build social networks that capture these interactions, how to navigate through these networks during Web services functioning, and how to maintain social networks in response to changes in Web services. Some other studies adopt $\mathcal{SWS}s$ to illustrate how Web services can help humans interact. For instance, a social service network is proposed in [6]. It integrates Web 2.0 aspects to enrich Web services with semantics. The social aspects, here, are not based on Web services interactions, but how users develop tags out of domain ontologies (i.e., folksonomies) so they assign them to Web services.

Xie et al. suggest a framework for semantic service composition based on social networks [20]. Trust between service providers, service consumers, and services themselves is the social element that is taken into account in this composition.

The framework consists of several modules including semantic extraction and social network construction, social network storage, and trust computing.

Maaradji et al. propose *SoCo* for *So*cial *Co*mposer. *SoCo* advises on the next course of actions to take in response to events such as Web services selection [17]. The advices are built upon the interactions that occur between users and Web services as well as the previously built compositions. *SoCo* consists of different components including social knowledge extraction and modeling, recommendation manager, connection manager, and service repository.

Maamar et al. develop *LinkedWS* as a social networks model for Web services discovery [14]. Different social networks permit to describe the situations in which Web services are engaged for instance collaboration and recommendation. *LinkedWS* stresses out that Web services should not be treated as isolated components that respond to users' queries, only. Contrarily, Web services compete against other, similar Web services during selection, collaborate with other, different Web services during composition, and may replace other, similar Web services during failure despite the competition. Competition and substitution relationships raise an interesting point, which is Web services competing to take part in compositions and at the same time collaborating to support each other during failure. This kind of "behavior" is referred to as coopetition standing for cooperation and competition [3].

Although short, the list of aforementioned research works shows the growing interest in $\mathcal{SWS}s$. To sustain this growth methods for engineering $\mathcal{SWS}s$ are highly deemed appropriate to help highlight the relationships between Web services, the social networks to build with respect to these relationships, and the mechanisms that let Web services use these networks during functioning.

3 Our Engineering Method

Our engineering method consists of four steps that each addresses one of the following questions: what kind of relationships can put Web services in contact, what social networks correspond to these relationships, what components constitute social networks of Web services, and what social behaviors can Web services exhibit. $\mathcal{SWS}s$ that result out of our engineering method are regular Web services that are connected to each other through social networks and exhibit social behaviors based on their role in these networks. We recall that functionality represents the "service" that a Web service offers to users and peers as well.

3.1 Overview

Like any engineering method, our method consists of steps and models that fall into either analysis **or** design phase (Fig. 1). During the analysis phase a service engineer performs two steps. First she establishes relationships between Web services as per the nature of the case study that is under discussion (Section 2.1). Afterwards the service engineer maps the relationships established previously onto social networks, though the mapping is not always one-to-one. During the design phase the service engineer performs two steps as well. First she defines

the characteristics of each social network in terms of number of nodes, types of edges connecting these nodes, and weight formulas for these edges. Finally the service engineer analyzes the social behaviors that Web services exhibit by being part of these networks.

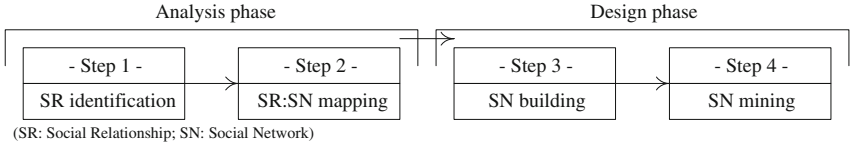

Fig. 1. General representation of $\mathcal{SWSs}$ engineering method

3.2 Step 1: What Relationships Can Put Web Services in Contact?

As stated earlier the objective of this step is to identify the relationships that can exist between Web services. The running scenario has shown three relationships in response to the following cases:

1. Web services that offer semantically similar functionalities like `ShipperWS` and `DeliveryWS`
   - **compete** against each other during selection as only one Web service is considered at a time [4].
   - **substitute** for each other in case of failure so that application operation continuity is maintained [15].
2. Web services that offer separate functionalities like *CustomerWS* and ***InventoryWS*** **collaborate** in the development of new added-value composite services [9].

3.3 Step 2: What Social Networks Correspond to Web Services' Relationships?

As stated earlier the objective of this step is to identify potential social networks that can put Web services in contact. Step 1 resulted in the identification of competition, substitution, and collaboration relationships. Each relationship constitutes a basis upon which a specific social network is developed. As a result, Web services can sign up with three types of social networks: competition, substitution, and collaboration.

- The objective of a competition social network is to make Web services aware of their competitors. This awareness triggers possible enhancements of Web services in case they regularly turn out less competitive at selection time [1].
- The objective of a substitution social network is to make Web services highly available in case failures arise [15]. The potential substitutes are reported in this network, which eases their identification in the future.

– The objective of a collaboration social network is to keep track of all the peers that worked with a Web service on the completion of compositions. The potential collaborators are reported in this network, which eases their identification in the future, as well.

In [14], social networks of Web services are classified into either positive or negative. This is based on the impact that Web services have on each other when they are in the same social network. A positive social network includes Web services that work together since their functionalities are different and hence, can be combined. Contrarily, a negative social network includes Web services that cannot work together since their functionalities are similar and hence, cannot be combined[1]. As per this classification, a competition social network is negative while the other two are positive.

3.4 Step 3: How to Build Social Networks of Web Services?

As stated earlier the objective of this step is to identify the components upon which the social networks of Step 2 are built. We refer to these components as node and edge. In our engineering method nodes and edges correspond to Web services and relationships, respectively.

Competition social network. Fig. 2 illustrates a simple competition social network. Since this network involves similarly functional Web services only, they are all in competition against each other and hence, all connected to each other through bidirectional edges.

To evaluate the weight of a competition edge, which we refer to as *Comp*etition *L*evel ($CompL$, Equation 1) between two Web services ws_i and ws_j, we use the Functionality Similarity Level (FSL) to compare their respective functionalities and the No-Functionality Similarity Level ($NFSL$) to compare their respective non-functional properties (QoS, e.g., reliability level, response time). We assume that the non-functional properties of Web services are defined with the same taxonomy. The use of FSL is shown in Section 4.2.

$$CompL_{ws_i,ws_j} = FSL_{ws_i,ws_j} \times (1 - NFSL_{ws_i,ws_j}) \quad (1)$$

where:

– FSL_{ws_i,ws_j} corresponds to the similarity level between the functionality of ws_i and the functionality of ws_j. This level is determined using existing approaches such as [5] and should be either close to or equal to 1.

[1] Not all Web services can be combined as this depends on factors such as (i) Web services belonging to the same domain for example travel (e.g., *AirBookingWS* and *TaxiBookingWS*) and (ii) data dependencies between Web services (e.g., *OutdoorActivityWS* depending on the feedback of *WeatherForecastWS*).

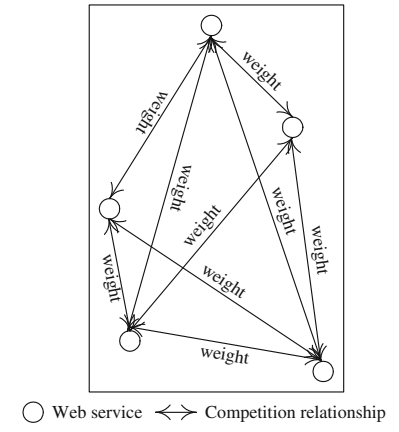

Fig. 2. Illustration of a competition social network

– $NFSL_{ws_i,ws_j} = \omega_1 \times (|P_{ws_{i,1}} - P_{ws_{j,1}}|) + \cdots + \omega_n \times (|P_{ws_{i,n}} - P_{ws_{j,n}}|)$ with $P_{ws_{i,k}}$ is the value of the k^{th} non-functional property of the i^{th} Web service (assumed to be between 0 and 1), ω_k is a weighting factor representing the importance of a non-functional property, and $\sum_{k=1}^{n} \omega_k = 1$.

As per Equation 1 the more the competition level is close to one, the closest ws_i is to ws_j. As a result ws_i threatens the competitiveness capacity of ws_j. We recall that only one Web service can be selected at a time to handle a user's request. A competition social network is useful when a Web service decides to reject processing a user's request for different reasons such as guaranteeing its non-functional properties [13]. This Web service's competition social network permits to identify its competitive Web services so that this request is assigned to one of them upon its approval.

Substitution social network. Fig. 2, also, illustrates a substitution social network after updating the edges' name from competition to substitution. It should be noted that not all edges are bidirectional. Since all Web services in a substitution social network offer the same functionality, any peer can be selected as a potential candidate that replaces a failing Web service.

To evaluate the weight of a competition edge, which we refer to as *Sub*stitution *L*evel (*SubL*) between ws_i and ws_j, we use the Functionality Similarity Level (FSL) and the No-Functionality Similarity Level ($NFSL$) like previously on top of the Reliability Level (RL) that shows how successful ws_i is when it replaces ws_j (Equation 2).

$$SubL_{ws_i,ws_j} = FSL_{ws_i,ws_j} \times RL_{ws_i,ws_j} \times (1 - NFSL_{ws_i,ws_j}) \qquad (2)$$

where:

– FSL_{ws_i,ws_j} and $NFSL_{ws_i,ws_j}$ are defined in Equation 1.

– $RL_{ws_i,ws_j} = \frac{\sum SR_{ws_i,ws_j}}{\sum TR_{ws_i,ws_j}}$, with $\sum SR_{ws_i,ws_j}$ is the total number of Successful Replacements that ws_i made for ws_j (i.e., no failure) and

$\sum TR_{ws_i,ws_j}$ is the Total number of Requests that ws_i received to replace ws_j. If ws_i never replaced ws_j then the substitution level is 0.

Collaboration social network. Fig. 3 illustrates a simple collaboration social network. It is built when at least one composition of Web services is complete. For navigation purposes in a collaboration social network, an entry node is required and represented differently from the rest of nodes (Fig. 3). We refer to this entry node as "focus" Web service. All edges coming out of this "focus" Web service are unidirectional pointing towards other Web services.

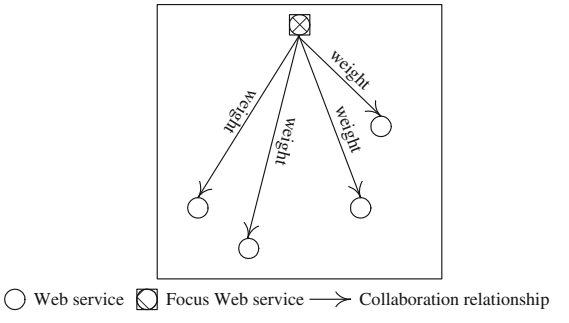

Fig. 3. Illustration of a collaboration social network

To evaluate the weight of a collaboration edge, which we refer to as *Col*laboration *L*evel (*ColL*) between ws_i ("focus") and ws_j, we track the number of times that both Web services participated in joint compositions with emphasis on the total number of compositions that ws_i took part in.

$$ColL_{ws_i,ws_j} = \frac{\sum JC_{ws_i,ws_j}}{\sum TP_{ws_i}} \tag{3}$$

where:

- $\sum JC_{ws_i,ws_j}$ is the total number of participations of ws_i and ws_j in Joint Compositions. $\sum JC_{ws_i,ws_j}$ and $\sum JC_{ws_j,ws_i}$ are equal.
- $\sum TP_{ws_i}$ is the Total number of Participations of ws_i in compositions.

3.5 Step 4: What Social Behaviors Can Web Services Exhibit?

As stated earlier the purpose of this step is to identify the potential social behaviors that Web services can exhibit based on the details that each type of social network (substitution, competition, and collaboration) carries on. Different types of social behaviors exist in real life such as selfish, trustworthy, opportunistic, malicious, vindictive, reliable, etc.

Selfish social behavior. Substitution reveals the selfishness of a Web service when this latter refuses continuously to replace failing peers. However these

peers accept continuously to replace this Web service when it failed. A Web service can use different reasons to back its refusal decisions including "fear" of not meeting its non-functional properties or inappropriateness for replacing a failing peer as per the competition social network ($CompL$ close to 0). To analyze selfishness the substitution relationship between ws_i and ws_j is used as follows, where ws_i substitutes ws_j:

- `If` $SubL_{ws_i,ws_j} = SubL_{ws_j,ws_i}$ `then` the substitution relationship is balanced between both.
- `If` $SubL_{ws_i,ws_j} > SubL_{ws_j,ws_i}$ `then` the substitution relationship is in favor of ws_j, i.e., ws_j did not replace ws_i as many as ws_i did for ws_j; Otherwise the substitution relationship is in favor of ws_i.

Definition 1. ***Selfishness.*** *A Web service ws_i exhibits a selfish behavior if the majority of its substitution relationships with peers are in its favor, i.e., the number of times that $SubL_{ws_i,ws_j} < SubL_{ws_j,ws_i}$ holds, is greater to a threshold T_{SubL}.* □

A Web service that is known as selfish can be ignored by similar peers since these ones cannot count on it when they fail. Corrective actions could be taken by reviewing this Web service's non-functional properties.

Malicious social behavior. Competition reveals the maliciousness of a Web service when it accepts to handle user requests that it receives from other peers, though this Web service is not sure to guarantee its QoS level. Initially these peers declined handling the user requests for reasons listed in the description of the selfish social behavior, and hope that this Web service will not disappoint them. This Web service is reported in these peers' competition social networks.

To analyze maliciousness we introduce a function, called disappointment Dis_{ws_i,ws_j} that tracks of the number of times that ws_i failed in maintaining its QoS level for the user requests it receives from ws_j over the total number of requests that ws_j passed on to ws_i ($Dis_{ws_i,ws_j} = \frac{\sum Fail_{ws_i}(req_{ws_j})}{\sum req_{ws_i,ws_j}}$).

- `If` $Dis_{ws_i,ws_j} = Dis_{ws_j,ws_i}$ `then` the disappointment relationship is balanced between both.
- `If` $Dis_{ws_i,ws_j} > Dis_{ws_j,ws_i}$ `then` the disappointment relationship affects ws_j more than ws_i;

Definition 2. ***Maliciousness.*** *A Web service ws_i exhibits a maliciousness behavior if it is involved in a large number of disappointment relationships with peers, i.e., the number of times that $Dis_{ws_i,ws_j} < Dis_{ws_j,ws_i}$ holds, is greater to a threshold T_{Dis}.* □

Dominant social behavior. Collaboration reveals the dominance of a Web service over a peer when this Web service participates in the compositions of this peer more than what this peer did in the compositions of this Web service.

To analyze dominance the collaboration relationship between ws_i and ws_j is used as follows:

- `If` $ColL_{ws_i,ws_j} = ColL_{ws_j,ws_i}$ `then` the collaboration relationship is balanced between both.
- `If` $ColL_{ws_i,ws_j} < ColL_{ws_j,ws_i}$ `then` the collaboration relationship is in favor of ws_i, i.e., ws_i did participate in the compositions of ws_j more than what ws_j did by participating in compositions of ws_j; Otherwise the collaboration relationship is in favor of ws_j.

Definition 3. ***Dominance.*** *A Web service ws_i exhibits a dominant behavior if the majority of its collaboration relationships with peers are not in its favor, i.e., the number of times that $ColL_{ws_i,ws_j} < ColL_{ws_j,ws_i}$ holds, is greater to a threshold T_{ColL}.* □

4 Implementation

We discuss first, the tools that support service engineers in developing $\mathcal{SWS}s$ and managing their respective social networks and then, report on our experience of dealing with $\mathcal{SWS}s$ in the context of *LinkedWS* project.

4.1 Support Tools

Service engineers who plan to convert Web services into $\mathcal{SWS}s$ use tools to perform this conversion along with other duties such as building $\mathcal{SWS}s$' networks, exhibiting $\mathcal{SWS}s$' behaviors, finding substitutes for failing $\mathcal{SWS}s$, etc.

The first tool called $\mathcal{F}$unctionality $\mathcal{A}$ssessment $\mathcal{T}$ool ($\mathcal{FAT}$) is used by service engineers to establish relationships between $\mathcal{SWS}s$ based on their respective functionalities (Steps 1 and 2). Functionality categorizes $\mathcal{SWS}s$ into either similar or different (Fig. 4). Multiple assessment techniques like those reported in [5] and [18] can be integrated into the $\mathcal{FAT}$. Out of the $\mathcal{FAT}$, two values are obtained: degree of similarity (ds) upon which competition and substitution social networks are built and degree of complementarity (dc) upon which collaboration social network is built.

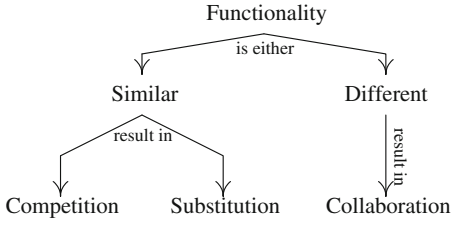

Fig. 4. $\mathcal{SWS}s$ categorization based on functionality

The second tool called $\mathcal{N}$etwork $\mathcal{M}$anagement $\mathcal{T}$ool ($\mathcal{NMT}$) is used by service engineers to build networks of $\mathcal{SWS}s$ and oversee changes in these networks (Step 3). A social network is either built from scratch or extended after adding new nodes/edges to it. For each type of social network discussed in Step 3, the collaboration, substitution, and competition levels ($ColL$, $SubL$, $CompL$) are calculated using the $\mathcal{NMT}$.

The third tool called $\mathcal{N}$etwork M$\mathcal{I}$ning $\mathcal{T}$ool ($\mathcal{NIT}$) is used by service engineers to analyze the details in the three social networks so that the social behaviors of each $\mathcal{SWS}$ are exhibited (Step 4).

4.2 Experience with *LinkedWS*

The development of *LinkedWS* is thoroughly detailed in [14]. We briefly illustrate the use of the $\mathcal{FAT}$ and $\mathcal{NMT}$.

Building upon the work of Min et al. [18], the $\mathcal{FAT}$ associates a Web service (s) with a profile that consists of precondition, input, output, effect, and QoS. The $\mathcal{FAT}$ establishes the degree of similarity $ds(s_i, ds_j)$ (i.e., FSL) (Equation 4) between s_i and s_j using a *matching score* (ms) function defined in Equation 5.

$$ds(s_i, s_j) = \frac{\sum_k w_k \times ms(c_{s_{i_k}}, c_{s_{j_k}})}{\sum_k w_k} \tag{4}$$

where k is the total number of concepts being similar and w_k is the weight associated with the matching score between a pair of concepts. The resulting degree of similarity is between 0 (completely dissimilar) and 1 (maximum similarity).

$$ms(c_{s_i}, c_{s_j}) = f_1 \times f_2 \times f_3 \tag{5}$$

where $f_1 = e^{\alpha l}$ with α as a constant, $f_2 = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}$ with β as a smoothing factor, and $f_3 = \frac{e^{\lambda l} - e^{-\lambda l}}{e^{\lambda l} + e^{-\lambda l}}$ with λ as another smoothing factor.

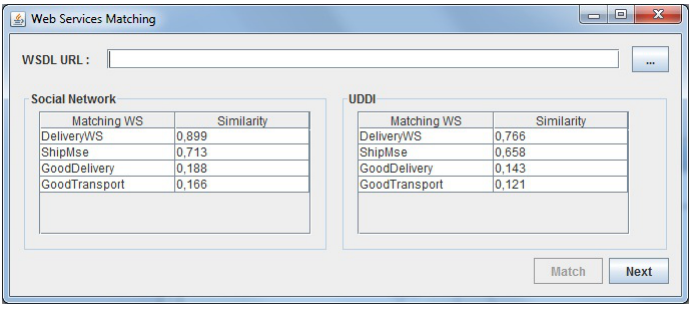

Fig. 5. $\mathcal{WSMC}$ interface for *ShipperWS*

To replace a failing Web service (e.g., *ShipperWS*) using a substitution social network, we implemented the $\mathcal{W}$eb $\mathcal{S}$ervices $\mathcal{M}$atching $\mathcal{C}$omponent ($\mathcal{WSMC}$ that is part of the $\mathcal{NMT}$). The $\mathcal{WSMC}$ is triggered when a Web service fails taking as input the WSDL document of this Web service (Fig. 5). The $\mathcal{WSMC}$ permits to a service engineer to navigate through the substitution social network of a failed Web service.

To build the substitution social network of *ShipperWS* (Fig. 6), we identified manually (using `jUDDI`) some similar peers such as *DeliveryWS*, *ShipMse*, *GoodDelivery*, and *GoodTransport*. Equation 4 assesses the initial weights of the edges that connect *ShipperWS* to these Web services. Once the substitution social network becomes effective these weights are reevaluated as per Equation 2. For instance the total number of Successful Replacements ($\sum SR_{ws_i,ShipperWS}$) that ws_i made for *ShipperWS* is updated where ws_i could for example correspond to *DeliveryWS*.

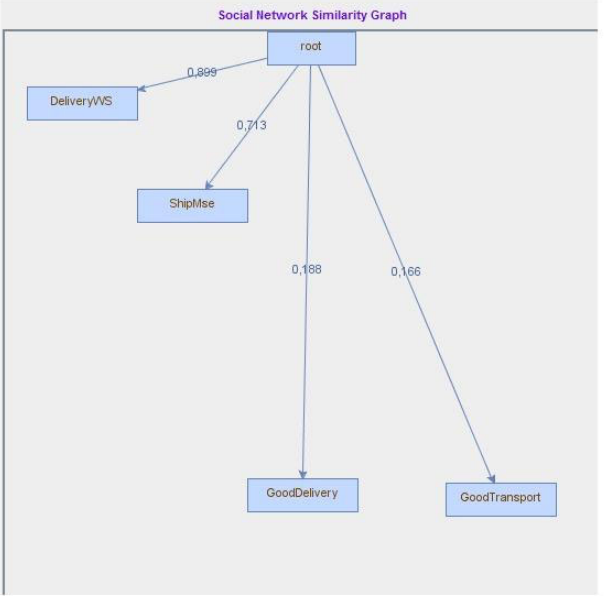

Fig. 6. *ShipperWS*'s substitution social network

5 Conclusion

In this paper we discussed a method to engineer social Web services. This engineering took into account the fact that Web services compete against similar peers during selection, collaborate with different peers during composition, and replace similar peers during failure. These three cases constitute the basis of developing networks of social Web services. The engineering method consists of four steps. In the first step, the objective is to shed the light on the potential

relationships between Web services. In the second step, the objective is to associate these relationships, once identified, with social networks. In the third and four steps, the objectives are to build and analyze these networks so that the social behaviors of Web services are established. Finally, a set of tools like functionality assessment supporting the engineering method were listed. Future work would consist of putting social Web services into action like we did in the *LinkedWS* project, which should permit a further refinement of the method.

References

1. Alrifai, M., Skoutas, D., Risse, T.: Selecting Skyline Services for QoS-based Web Service Composition. In: Proceedings of the the 19th International World Wide Web Conference (WWW 2010), Raleigh, North Carolina, USA (2010)
2. Badr, Y., Maamar, Z.: Can Enterprises Capitalize on Their Social Networks? Cutter IT Journal 22(10) (October 2009)
3. Bengtsson, M., Kock, S.: Coopetition in Business Networks to Cooperate and Compete Simultaneously. Industrial Marketing Management 29(5) (2000)
4. Bui, T., Gacher, A.: Web Services for Negotiation and Bargaining in Electronic Markets: Design Requirements and Implementation Framework. In: Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS 2005), Big Island, Hawaii, USA (2005)
5. Di Martino, B.: Semantic Web Services Discovery based on Structural Ontology Matching. International Journal of Web and Grid Services 5(1) (2009)
6. El-Goarany, K., Saleh, I., Kulczycki, G.: The Social Service Network - Web 2.0 Can Make Semantic Web Services Happen. In: Proceesings of the 10th IEEE Conference on E-Commerce Technology (CEC 2008) and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2008), Washington, DC, USA (2008)
7. Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J.P., Voß, M., Willkomm, J.: A Method for Engineering a True Service-Oriented Architecture. In: Proceedings of the Tenth International Conference on Enterprise Information Systems (ICEIS 2008), Barcelona, Spain (2008)
8. Fei-Yue, W., Kathleen, C., Daniel, M.Z., Wenji, M.: Social Computing: From Social Informatics to Social Intelligence. IEEE Intelligent Systems 22(2) (March 2007)
9. Fluegge, M., Santos, I.J.G., Paiva Tizzo, N., Madeira, E.R.M.: Challenges and Techniques on the Road to Dynamically Compose Web Services. In: Proceedings of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, California, USA (2006)
10. Foster, H., Uchitel, S., Magee, J., Kramer, J., Hu, M.: Using a Rigorous Approach for Engineering Web Service Compositions: A Case Study. In: Proceedings of the 2005 IEEE International Conference on Services Computing (SCC 2005), Orkando, Florida, USA (2005)
11. Karakostas, B., Zorgios, Y.: Engineering Service Oriented Systems: A Model Driven Approach. IGI Global Publishing (2008)
12. King, I., Li, J., Tong Chan, K.: A Brief Survey of Computational Approaches in Social Computing. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2009), Atlanta, Georgia, USA (2009)

13. Maamar, Z., Kouadri Mostéfaoui, S., Yahyaoui, H.: Towards an Agent-based and Context-oriented Approach for Web Services Composition. IEEE Transactions on Knowledge and Data Engineering 17(5) (May 2005)
14. Maamar, Z., Krug Wives, L., Badr, Y., Elnaffar, S., Boukadi, K., Faci, N.: LinkedWS: A Novel Web Services Discovery Model Based on the Metaphor of "Social Networks". In: Simulation Modelling Practice and Theory, vol. 19(10), Elsevier Science Publisher, Amsterdam (2011)
15. Maamar, Z., Sheng, Q.Z., Tata, S., Benslimane, D., Sellami, M.: Towards an Approach to Sustain Web Services High-Availability Using Communities of Web Services. International Journal of Web Information Systems 5(1) (2009)
16. Maamar, Z., Tata, S., Yétongnon, K., Benslimane, D., Thiran, P.: A Goal-based Approach to Engineering Capacity-driven Web Services. The Knowledge Engineering Review Journal, Special issue on Web and Mobile Information Services (2010) (forthcoming)
17. Maaradji, A., Hacid, H., Daigremont, J., Crespi, N.: Towards a Social Network Based Approach for Services Composition. In: Proceedings of the 2010 IEEE International Conference on Communications (ICC 2010), Cap Town, South Africa (2010)
18. Min, L., Weiming, S., Qi, H., Junwei, Y.: A Weighted Ontology-based Semantic Similarity Algorithm for Web Services. Expert Systems with Applications 36(10) (December 2009)
19. Papazoglou, M.: Web Services: Principles and Technology. Prentice Hall, Englewood Cliffs (2007)
20. Xie, X., Du, B., Zhang, Z.: Semantic Service Composition based on Social Network. In: Proceedings of the 17th International World Wide Web Conference (WWW 2008), Beijing, China (2008)