Social Web Services Development Based on MDA: Extending WSDL to Inject Social-QoS

Hichem BOUCHAKOUR ERRAHMANI
Computer science department, EEDIS laboratory, Djilali Liabès University of sidi-bel-abbès, Algeria
Email:b_hichem@hotmail.fr

Sidi Mohammed BENSLIMANE
Computer science department, EEDIS laboratory, Djilali Liabès University of sidi-bel-abbès, Algeria
Email: benslimane@univ-sba.dz

Abstract: Injecting social networks principles into service oriented architecture falls into our research work on social Web services. This paper will focus on social Web services design and development by examining the description, discovery, and binding specifications of Web services from a social perspective. We look at approaches that adapt models to represent systems. Models may also be transformed into other models and the transformation itself is a model. Model-driven architecture is free of social elements that could be taken into account in the process of design systems. This raises questions about the appropriateness of model-driven architecture for social Web services design and development, unless proper actions are carried out to enhance model-driven architecture with appropriate means. In this paper we address the following list of issues: how are social aspects of services identified, represented, and structured in a model, how to specify a service joining a social network, how to transform social models into implementation models (e.g., SWSDL for Social WSDL), how to provide an automatic tool for this transformation?

Keywords: Web Service, Social Network, Model Driven Architecture.

1. Introduction

Social networks are one of the latest trends in Web applications development. Facebook and MySpace, for example, have reinforced the role of the Web as an inevitable platform for cross-collaboration and cross-communication between independent partners. Indeed social computing introduces a new dimension to the Web by capitalizing on users’ ability and willingness to interact, share, collaborate, and make recommendations.

There is certainly an overlap between social computing and service-oriented computing. On the one hand, social computing represents relationships that people experience daily, such as trustworthiness and fairness, via automated structures known as social networks. On the other hand service-oriented computing abstracts application design using the principles of "I offer services that somebody else might require" and "I require services that somebody else might offer". Injecting social networks principles into service-oriented architecture falls into our research work on social Web services [6].

This paper will focus on social Web services design and development by examining the description, discovery, and binding specifications of Web services from a social perspective. We look at approaches that adopt models to represent systems. Models may also be transformed into other models and the transformation itself is a model.

Model-driven architecture is free of social elements that could be taken into account in the process of design systems. This raises questions about the appropriateness of model-driven architecture for social Web services design and development, unless proper actions are carried out to enhance model-driven architecture with appropriate means. In this paper we address the following list of issues: how are social aspects of services identified, represented, and structured in a model, how to specify a service joining a social network, how to transform social models into implementation models (e.g., S-WSDL for Social WSDL), how to provide an automatic tool for this transformation?

The organization of this paper is as follow, section 2 shows how social computing can help to improve Service-Oriented Architecture (SOA), section 3 is devoted to describe some works related to the extension of WSDL to add additional information such as quality of service, section 4 presents our approach of social Web services development based on MDA, followed by section 5 which implements our approach, where we will explain our prototype before concluding in Section 6.

2. Using Social Computing to Improve Service Computing

The current situation of SOA has limited large scale use of Web services, because several important questions remain unanswered, such as where to advertise for improved services? How to discover the services compared to the needs of users? How to trust services when they are found? And how to replace services smoothly when they fail? [6]. these issues made the services below their potential because they:

- know only about themselves and not on their users or peers;
- limit user intervention and operate as black boxes;
- consider only their own internal details functional and non-functional during the execution and ignoring other exterior details such as the user's past interactions;
- cannot delegate their invocations;
- Do not cooperate instantly and spontaneously with each other or the self-organization.

Fortunately, the semantic behavior is available on how services are used and combined, and how they behave and interact. This is the social side of services [9]. The capture of the service interactions using, for example, social
networks can be useful for engineers who can take advantage of the known interactions successfully as needed [6]. If social networks could capture all these interactions, a service (SOA compliant) could:

- be aware of peers who are competing with him in case of selection;
- recommend peers with whom he would collaborate in the case of composition;
- Recommend peers that can substitute him in case of failure.

We consider the social Web Services (SWS) as the result of blending of Social Computing with Service-Oriented Computing (SOC).

On the one hand, the Social Computing consists in facilitating the calculation of social studies and human social dynamics as well as the design and the use of information technology and communication that take into account the social context [11]. Social Computing also concerns collective actions, content sharing and diffusion of information in general.

On the other hand, SOC built applications on the principles of offer and demand for services, as well as loose coupling [10]. Combine Social Computing with SOC leads to Social Computing with Service-Oriented Computing (SOC).

To evaluate the weight of a collaboration edge, which we call Collaboration Level (ColL) between \(w_s\) and \(w_j\), we track the number of times that the two Web services have participated in joint compositions, focusing on the total number of compositions \(w_s\) are involved.

$$\text{ColL}_{w_s, w_j} = \frac{\sum J_{w_s, w_j}}{\sum TP_{w_s}}$$

Where:

- \(\sum J_{w_s, w_j}\) is the total number of participations of \(w_s\) and \(w_j\) in joint compositions.
- \(\sum J_{w_s, w_j}\) and \(\sum J_{w_s, w_j}\) are equal;
- \(\sum TP_{w_s}\) is the total number of participations of \(w_s\) in joint compositions.

- **Collaboration:** SWS compete against each other when they offer similar functionalities. Their non-functional properties differentiate them when users’ non-functional requirements must be satisfied. Consequently, a SWS learns about its own network of competitors, so that it can attempt to improve its non-functional properties with respect to other peers [6].

![Figure 1](image1.png)

**Figure 1.** Illustration of a social network of collaboration

![Figure 2](image2.png)

**Figure 2.** Illustration of a social network of competition

To evaluate the weight of a competition edge, which we call Competition Level (CompL) between two services \(w_s\) and \(w_j\), we use the Functionality Similarity Level (FSL) to compare their respective functionalities (This level is determined by using existent approaches like [3]) and the Non-Functionality Similarity Level (NFSL) to compare their non-functional properties (QoS, for example: reliability, response time).

$$\text{CompL}_{w_s, w_j} = \text{FSL}_{w_s, w_j} \times (1 - \text{NFSL}_{w_s, w_j})$$

Where:

$$\text{NFSL}_{w_s, w_j} = w_{ik} \times \left(\sum_{j=1}^{n} \left(\frac{P_{w_s,j_1} - P_{w_j,j_1}}{\sum_{j=1}^{n} P_{w_s,j_1}}\right)\right) + \cdots + w_{in} \times \left(\frac{P_{w_s,j_n} - P_{w_j,j_n}}{\sum_{j=1}^{n} P_{w_s,j_n}}\right) + \sum_{k=1}^{m} w_k = 1$$

- **Substitution:** Although SWS compete against each other, they can still help each other when they fail if they offer similar functionalities. Consequently, a SWS manages its own networks of substitutes, so that it can
identify its own best substitutes in response to users’ non-functional requirements [6].

To evaluate the weight of a substitution edge, which we call Substitution Level (SubL) between ws_i and ws_j, we use FSL and NFSL as previous, and also the Reliability Level (RL).

\[ \text{SubL}_{ws_i,ws_j} = \text{FSL}_{ws_i,ws_j} \times RL_{ws_i,ws_j} \times (1 - \text{NFSL}_{ws_i,ws_j}) \]

Where:

\[ \text{RL}_{ws_i,ws_j} = \frac{\sum \text{SR}_{ws_i,ws_j}}{\sum \text{TR}_{ws_i,ws_j}} \]

Here are some social qualities that can be deduced from the behavior of a web service. The calculation of their values is implemented by Z. Maamar in [5].

- **Selfish:** Substitution reveals the selfishness of a Web service when this latter refuses continuously to replace failing peers. A Web service exhibits a selfish behavior if the majority of its substitution relationships with peers are in its favor.

- **Spiteful:** Competition reveals the spitefulness of a Web service when agreeing to process user requests it receives from other peers, although this Web service is not sure to guarantee the QoS. A Web service has a spiteful behavior, if it is involved in many disappointment relationships with peers.

- **Dominant:** Collaboration reveals the dominance of a Web service on a pair, if this Web service participates in the compositions of this pair more than what made this pair in the compositions of its collaborative relationships with peers.

2.2. Social Networks of Users

Users can also have relationships with services, which characterizes the social networks that highlight the interaction that lie between user-user and user-Web service, we cite some metrics to illustrate our network:

- **Like:** The number of positive votes that the Web service has had from all users;

- **Dislike:** The number of negative votes that the Web service has had from all users;

- **Share:** It represents the number of times that the customer shared (recommended) the service, however the idea of recommendation does not relate to a positive opinion, but on the experience that had the client with the service, in which this sharing could be associated with a comment.

3. Related Works

We propose to see some works on the extension of WSDL, to take advantage for elaborating our own architecture of SWS development based on MDA.

Fei Cao et al. (2004) have proposed an extension of WSDL to support specific characteristics of a given domain (the example used in this approach is that of the bank), for that they used MIC (Model Integrating Computing) which defines a language specific modeling. The principle is based on a general architecture of metamodeling Web services that will be used in the construction of an environment modeling of specific Web services to a domain. [1].

Andrea D’Ambrogio (2006) realized a metamodel transformation to generate Q-WSDL extension of WSDL to contain the characteristics of QoS such as performance, reliability, availability, security, etc. The proposed extension (Q-WSDL) is defined by first introducing the WSDL metamodel, derived from the XML Schema of the WSDL document, and then transform it into a Q-WSDL metamodel, from which the XML schema document Q-WSDL is derived to produce a valid document Q-WSDL [2].

The team of N. Parimala et al. (2011) introduced in Web service a new information called criteria that depends on the functional property that will be combined in the search for such a service. To do this, they extended the WSDL file to X-WSDL by adding the non-functional property called criteria [8].

After extended into a previous work, UML profiles in AWS-UML (Adaptive Web Service Unified Modeling Language), which describes the appropriate user profiles. In this work, Chiraz El Hog et al. (2011) have extended the service description (WSDL) to support the additional adaptation information. For this, they have proposed an extension of WSDL and named AWS-WSDL, and this extension can be used by the provider to describe the adaptation criteria supported by the Web service or the consumer to express his needs specific adaptation in formalizing his request [4].

We have seen different approaches known for extending Web service description, therefore, we propose in the next section a lucid approach to the development of social Web services based on MDA, the novelty in this approach is how to model the social aspect of a Web service, which actually reacts to the commitment of Web services in social networks, and to integrate this model into a WSDL metamodel to obtain a coherent S-WSDL metamodel.

4. Our Approach

![Figure 3. Global architecture of our approach](image-url)
Our approach (Figure 3) involves two steps, the first is to get social information of the Web service as a model, and then, in the second step, we make a metamodels transformation to obtain the final document S-WSDL from an S-WSDL metamodel.

In the following, we will explain the three steps shown by the blue boxes.

4.1. SWS Design

To build the network of social Web services, we have relied on several services through a directory and a log file that describes the history of interaction between Web services (Figure 4), without forgetting our Web service, represented by its WSDL description file.

After processing, we will have built the three social networks, social network of collaboration, of substitution, and of competition.

At this stage, we will calculate the different qualities of services seen in [5]. In addition to these qualities, others may be added as the reputation and trust.

4.1.1. Reputation

Reputation should help users to find the right Web service according to their needs. The metrics that we consider to get the reputation are share (shown in section 2.2) and satisfaction:

- Satisfaction: It is the subjective opinion of customers who have recently dealt with the Web service.

Thus, the general formula to calculate the reputation is given as follows:

$$ reputation = \frac{like + share}{like + share + dislike} $$

4.1.2. Trust:

Trust is calculated based on the behaviour of service to users (i.e. of reputation), and service to his peers (social qualities seen in section 2.1).

Thus, the general formula of the trust is:

$$ Trust = \frac{\alpha \cdot reputation + \beta \cdot selfishness + \gamma \cdot spitefulness + \delta \cdot dominance}{\alpha + \beta + \gamma + \delta} $$

Where:

$\alpha$, $\beta$, $\gamma$, and $\delta$ are weighting coefficients.

4.2. Social Aspect Modeling

In this step, we have obtained a model containing social qualities of our service (Figure 5).

An instance of such model is:

```xml
<Social-QoS>
  <Quality name="selfish" value="0.55"/>
  <Quality name="dominant" value="0.33"/>
  <Quality name="spiteful" value="0.0"/>
  <Quality name="like" value="76"/>
  <Quality name="share" value="99"/>
  <Quality name="satisfaction" value="0.69"/>
  <Quality name="trust" value="0.75"/>
  <List type="Substitution">
    <node name="service 2" label="0.76"/>
  </List>
  <List type="Collaboration">
    <node name="service 0" label="0.25"/>
    <node name="service 1" label="0.25"/>
    <node name="service 10" label="0.25"/>
  </List>
  <List type="Competition">
    <node name="service 8" label="0.09"/>
  </List>
</Social-QoS>
```

Each quality has two attributes name and value, and then we have the list of substitutes, collaborators, and competitors of the service concerned.

4.3. Metamodel Transformation

This is the key step of our work; it consists to establish an automatic transformation of the WSDL metamodel to the S-WSDL metamodel (Figure 6).
The transformation is performed by adding the social qualities described in the social metamodel to the WSDL metamodel. It is a transformation of tree type, where the tree structure of WSDL metamodel is modified, so that a branch describing the social model is added. The proper tool for this kind of transformation will be an XML specific language like XSLT because it handles XML files and all the metamodels are serialized in XMI (XML Metadata Interchange) files.

The transformation (Figure 6) begins with the serialization of WSDL file; obviously it is consistent with the WSDL metamodel. Then we apply transformation rule written in XSLT to merge the two models, the WSDL one, and the social aspect one, as a result we will have an S-WSDL metamodel (Figure 7).

Figure 7. S-WSDL MetaModel

5. Experimentation

To illustrate our approach, we have developed an application that tests a development scenario of social web services.

The first interface (Figure 8) shows the creation of several services, among them one is chosen to be the subject of our evaluation. In this example we create 13 services, study the social qualities of service 3 by drawing a graph showing the relationship between service 3 and the others, edges in red indicate a competition relationship, in green, a substitution relationship, and those in blue indicate a collaboration relationship then we calculate their weight.
Service 3 has a collaborative relationship with services annotated by 0, 1, 4, 6, 7 etc. After calculating the best collaborators, the list is reduced to three services: 0, 1 and 10.

On the second interface shown in Figure 9, we will build our S-WSDL file from the WSDL file of the service 3 and its social description given in the previous interface. The area on the left represents a simple WSDL file, after MDA transformation, we get an S-WSDL file on the right.

6. Conclusion
Throughout this paper, we were concerned in one form or another by the implementation of the MDA approach in the development of Web services. The Web services technology was sufficient for the market development of software applications until the evolution of the Internet and the metamorphosis of traditional computing, to share ideas, opinions, under automated structures such as social networks. Then it has been necessary to use or inject the
principles of social networks in service-oriented architecture.

The work of Zakaria Maamar [5] [6], gave birth to a new technology named social Web services, which implements the SOA and conforms to Web 2.0. Except for the development of the latter, this requires a lot of complexity and resource management. We considered it useful to remedy this by exploiting the MDA approach and take advantage of its benefits to develop a tool for automatic development of SWS.

References


