Analyzing Social Web Services' Capabilities

Zakaria Maamar[1], Hamdi Yahyaoui[2], Azzam Mourad[3], and Mohamed Sellami[4]

[1]Zayed University, Dubai, U.A.E, *zakaria.maamar@zu.ac.ae*

[2]Kuwait University, Kuwait City, Kuwait, *hamdi@cs.ku.edu.kw*

[3]Lebanese American University, Beirut, Lebanon, *azzam.mourad@lau.edu.lb*

[4]ISEP Paris, Paris, France, *mohamed.sellami@isep.fr*

Abstract

This paper looks into ways of supporting social Web services react to the behaviors that their peers expose at run time. Examples of behaviors include selfishness and unfairness. These reactions are associated with actions packaged into capabilities. A capability allows a social Web service to stop exchanging private details with a peer and/or to suspend collaborating with another peer, for example. The analysis of capability results into three types referred to as functional (what a social Web service does), non-functional (how a social Web service runs), and social (how a social Web service reacts to peers). To avoid cross-cutting concerns among these capabilities aspect-oriented programming is used for implementing the proof-of-concept system.
Keywords: *Behavior, Capability, Social Web service, AOP.*

1. Introduction

Web Services (WSs) are regularly hailed for their role in supporting the development of business processes that can across organization boundaries transparently and smoothly [21]. Today's organizations deem necessary to work with different partners so they remain competitive, respond to globalization challenges, secure more market-share, etc. Simply put a WS is an accessible online application that other applications and humans as well can discover and trigger. Benatallah et al. associate the following properties with a WS [4]: independent as much as possible from specific platforms and computing paradigms; developed primarily for inter-organizational and not intra-organizational situations; and easily composable so that developing complex adapters for the needs of composition is not required.

On top of having an online presence on the Internet, organizations are now embracing new means to reach out to their stakeholders such as consumers and suppliers. Web 2.0 applications such as Facebook and Twitter illustrate some of these means [2]. Many organizations are getting the message about Web 2.0: "*enterprise spending on Web 2.0 technologies will grow strongly over the next five years, reaching $4.6 billion globally by 2013, with social networking, Mashups, and RSS capturing the greatest share*" [7]. Contrary to traditional organizations that adopt top-down command flow and bottom-up feedback flow, these flows in a Web 2.0 organization (*aka* social enterprise [8]) cross all levels and in all directions giving room for creativity and innovation to employees.

In line with the social "fever" that has caught every single activity of people's daily lives ranging from sharing live experiences online to seeking feedback on any matter like what to cook on a special occasion, we demonstrated on different occasions the outcome of blending social computing (exemplified with Social Networks (SNs)) with service-oriented computing (exemplified with WSs technology). We refer to this outcome as Social Web Services (SWSs) [10, 11]. Compared to (regular) WSs, SWSs establish and maintain networks of contacts; count on their ("privileged") contacts when needed; form with their contacts strong and long lasting collaborative groups; and know with whom to partner so that reconciliation efforts are minimized due to ontology and policy disparities [14].

The aforementioned four operations illustrate the capabilities (c) that support SWSs function compared to regular WSs. Without these capabilities a SWS cannot identify the peers that it would like to work with, for example. In this paper we examine the specification of such capabilities by decomposing them into three types: functional (fc: what a SWS does), non-functional (nfc: how a SWS runs), and social (sc: how a SWS reacts to what other SWSs do). WSs capabilities are reported through the literature. For instance, Derguech and Bhiri use capabilities to discover WSs [6] and Maamar et al. use capabilities (referred to them as capacities) to address the particular concern that WSs do not cater to different levels of service offers and hence, treat all user invocation requests uniformly [16]. To the best of our knowledge this is the first time that capabil-

ity analysis targets SWSs. Our contributions are manifold namely (*i*) capability categorization into functional, non-functional, and social, (*ii*) concise definition of capability in the particular context of SWSs, (*iii*) analysis of how SWSs react to peers' actions and behaviors using social capabilities, and (*iv*) a system implementing social capability using Aspect-Oriented Programming (AOP). The rest of this paper is organized as follows. Section 2 is overview of SWSs and capabilities in the WSs literature. Section 3 specifies SWSs' capabilities. Prior to concluding in Section 5, implementation is presented in Section 4.

2. Background

2.1. Social Web services in a nutshell

Three communities analyze the blend of social computing with service-oriented computing. A first community deploys SNs of persons using WSs as a development technology of these SNs. A second community deploys SNs of SWSs to address issues like WSs discovery. Finally a third community mixes SNs of users and SNs of SWSs to develop composite WSs.

In the first community, we cite initiatives such as [1, 3, 17, 18, 23, 24]. Al-Sharawneh and Williams [1], mix semantic Web, SNs, and recommender systems to help users select WSs with respect to their functional and non-functional requirements. Bansal et al. [3] examine trust for WSs discovery. Users' trust in the providers of WSs is the social element in this discovery. Maaradji et al. [17] propose a social composer known as *SoCo* to advise users on the next actions to take in response to specific events like selecting specific WSs. Last but not least, Wu et al. [23] rank WSs based on their popularity among users.

In the second community, we cite other initiatives such as [5, 12, 15, 25]. Chen and Paik [5], build a global social service network to improve service discovery. They link services together using specific data correlations. Maamar et al. [12] develop a method to engineer SWSs. Questions that the method addresses include what relations exist between WSs, what SNs correspond to these relationships, how to build SNs of SWSs, and what social behaviors can SWSs exhibit. Last but not least, Maamar et al. [15] use SNs of SWSs to tackle the "thorny" problem of WSs discovery. WSs run into various situations at run time like competing against similar peers during selection, collaborating with peers during composition, and replacing similar peers during failure despite the competition. These situations help build the privileged contacts of a SWS.

In the last community that mixes SNs of users and SNs of SWSs, Maamar et al. [13] intertwine these networks to compose, execute, and monitor composite WSs. To achieve this intertwine three components namely composer, executor, and monitor are developed. The social composer develops composite WSs considering social relations between users and between WSs. The social executor assesses the impact of these relations on these composite WSs execution progress. Finally, the social monitor replaces failing WSs to guarantee the execution continuity of these composite WSs.

2.2. Capabilities of Web services

As stated in Section 1 capability adoption is extensively reported in the WSs literature. However, and to the best of our knowledge, nothing is reported in the particular context of SWSs. In the following we discuss some work on capability/WS combination.

Derguech and Bhiri [6] discuss WSs' capabilities from modeling, interlinking, and discovery perspectives, and stress out that "*a good capability description is a must either for allowing machine processing or human centricity*". The authors identify the shortcomings of description of WSs' capabilities namely capabilities are treated as annotations and not functionalities, capabilities are described at different levels of abstraction, and there is no explicit link between these levels which requires a manual intervention to identify the necessary capabilities that satisfy users' needs. To address these shortcomings Derguech and Bhiri develop a meta-model for capability description along with the following three principles: capability is described via domain specific features, capability offers can be generated automatically from the abstract descriptions, and capabilities are described at several levels of abstraction and explicit links are established between these levels.

Kagal et al. [9] describe WSs' capabilities and constraints using declarative policies that are based on deontic concepts namely permissions, obligations, claims, prohibitions, and privileges. These policies describe what the ideal behavior for an entity should be in a certain context. The policies are specified in the Rei language, which is based on OWL-Lite.

Maamar et al. [16] propose a goal-based approach for engineering capacity-driven WSs. Capacity, which is basically a set of operations to execute, is used instead of capability. In this approach goals define the roles that capacity-driven WSs will play in implementing business applications, frame the requirements that will be put on these WSs, and identify the business processes that these WSs will carry out. Because of the nature of capacity-driven WSs compared to regular (i.e., mono-capacity) WSs, their engineering is quite different. Indeed a capacity-driven WS has to choose out of many the capacity to trigger at run-time taking into account some requirements such as data availability and privacy level.

Paolucci et al. [19] discuss the semantic matching of

WSs' capabilities. This match requires a declarative language of capabilities so that what is looked for *versus* what is offered occurs with success. The authors propose an approach based on DAML-S with focus on profiles. Because WSs may offer several functionalities, not all of them need to be advertised to the community. For Paolucci et al. a WS's capability consists of inputs, outputs, preconditions, and effects.

Senivongse et al. [20] examine invocations of service clients to a particular WS for the sake of finding fine-grained capabilities that can be invoked so that a task is fulfilled. The capability granularity analysis framework monitors invocations to WSs and analyzes them using association rules and an algorithm to discover relations between the invoked operations. The analysis result can suggest to the service provider some possibilities of combining certain operations in order to reduce invocation costs.

The aforementioned paragraphs provide an overview of some existing works on WSs' capabilities. However social aspects are still overlooked. There is enough competition between WSs that these aspects will surely affect their selection when answering users' requests. For instance selfishness would require special arrangements to ensure fair collaboration among all WSs.

3. Specification of SWSs' capabilities

This section discusses the foundations of SWSs' capabilities and then presents how capabilities are put into action. For the needs of activating capabilities we assume that each SN is led by an authority component referred to as SN_{auth} (more details are given in [11]).

3.1. Foundations

We decompose a SWS's capability (c) into into three categories: fc, nfc, and sc. fc refers to the functionality (e.g., *reserve meeting-room*) of the SWS that both users and peers invoke at run-time. The specification of this functionality is ontology-dependent and outside this work's scope. nfc refers to the non-functional properties (*aka* as QoS) that establish the performance of the SWS at run-time. Like with fc, the establishment of nfc does not fall into this work's scope. Last but not least sc refers to the additional actions (neither related to fc nor to nfc) that empower a SWS since it now signs up in many SNs[1].

By being a member of a SN a SWS interacts with other SWSs in this SN and reacts to their behaviors. For this purpose we decompose sc into operation (*op*) and behavior (*be*). On the one hand sc_{op} refers to actions dedicated to SNs management like sign-in, sign-off, and contact members. On the other hand sc_{be} refers to actions that permit to respond to some members' behaviors in a SN. For instance if the SN_{auth} labels a member as selfish in its network then the SWSs in the network may (depending on their "utility functions") execute actions included in sc_{be} to tackle this selfishness. More details on these actions are given in Section 3.2.

Definition 1 (*Capability*). A SWS capability is a triple SWS-c = $< \mathsf{fc}, \mathsf{nfc}, \mathsf{sc} >$ where:

1. fc is a 4-tuple `<Input, Output, Condition, Effect>`.
2. nfc is a set of non-functional properties $\{\mathsf{nf}_{p1}, \mathsf{nf}_{p2}, ... \mathsf{nf}_{pi}\}$.
3. sc is a pair $< \mathsf{sc}_{op}, \mathsf{sc}_{be} >$ where:
   (a) sc_{op} is a set of actions $\{\mathsf{a}_{op_1}, \mathsf{a}_{op_2}, ... \mathsf{a}_{op_i}\}$ (that are different from the actions in the business logic of fc).
   (b) sc_{be} is a set of pairs $\{(\mathsf{be}_1, \{\mathsf{a}_{be_{1,i}}\}), (\mathsf{be}_2, \{\mathsf{a}_{be_{2,j}}\}), ... (\mathsf{be}_j, \{\mathsf{a}_{be_{j,k}}\})\}$ with be_i being a certain behavior (e.g., selfish and unfair) and $\{\mathsf{a}_{be_{i,j}}\}$ being a set of actions to carry out in order to respond to be_i. It might happen that (be_i, ϕ) exists, which means that no actions are available to respond to a certain behavior.

Note: all be_i are different and that some actions could be common to behaviors, i.e., $\cap\{\mathsf{a}_{be_{i,j}}\} \neq \emptyset$.

3.2. Reacting to peers' behaviors

A social capability (sc) allows a SWS to react to peers' behaviors so that this SWS continues functioning as per the objectives (e.g., process more users' requests) that it (i.e., its owner) has set. These reactions take different forms and will be discussed hereafter. Prior to this we define the sc's life cycle using 3 phases (Fig. 1)[2]: sensing, assessment, and action (focus of this paper).

1. Sensing phase consists of collecting details on peers in a network in terms of what actions they execute, what responses they provide, what credentials (e.g., reputation and credibility) they announce, etc. These details allow to establish the peers' behaviors as per the next phase.
2. Assessment phase consists of evaluating the details collected during the sensing phase in preparation for taking

[1] Collaboration, competition, and substitution are examples of social networks of SWSs [14].

[2] According to West et al., achieving dynamic capabilities with cloud computing includes four dimension [22]: sensing the environment, learning, integrating knowledge, and coordinating activities.

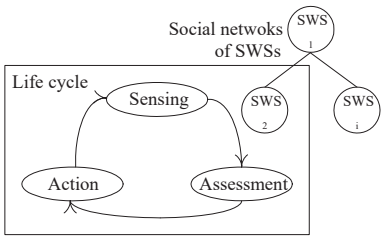

Figure 1. Representation of sc's lifecycle

actions as per the next phase. This evaluation uses rules that consist of Conditions (C) over Values (V) related to specific non-functional properties (nf_p). We recall that these properties constitute the nfc of the peer SWS_k. A rule is defined as follows:

$$\bigwedge C_i \Rightarrow \mathsf{be}_j \quad (1)$$

where C_i compares $V_{\mathsf{nf}_{pi}}$ to a Threshold (T_i) that the provider of $\mathsf{SWS}_{\_\neq k}$ sets ($\mathsf{compare}(V_{\mathsf{nf}_{pi}}, T_i)$). be_j is an element of a set of Behaviors (Be) (e.g., selfish and unfair) and denotes the behavior that SWS_k exposes.

3. Action phase consists of taking actions (reported in sc_{be} of $\mathsf{SWS}_{\_\neq k}$), if necessary, to respond to the behavior (be_j) defined in the assessment phase. Examples of actions include stop sharing private data with a peer, review the cooperation level with a peer, and adjust some non-functional properties to serve well a peer.

In [25] we made SWSs expose certain social behaviors like selfishness, untrustworthiness, and fairness based on factors that characterize collaboration, substitution, and selection scenarios. For instance, "*a slave Web service behaves in a selfish way if it does not show a positive attitude towards first, its direct master Web service and second, other peers located in its community or other communities*" [25]. In the following we develop strategies to respond to selfishness, untrustworthiness, and unfairness behaviors of SWSs (e.g., SWS_k) in networks.

A) How to deal with selfishness behavior (be:selfish)? Selfishness arises when a peer continuously refuses to act either as a substitute for a failing SWS or as a collaborator in an ongoing composition upon the request of a SWS. In either way the SWS could take actions included in $\mathsf{sc}_{selfish}$:

1. Substitution scenario: depending on the criticality level (e.g., high) of the composition that the SWS now takes part in, the SWS could take the following actions:
   (a) $\mathsf{a}_{selfish,1}$: the SWS stops relying on the peer for future substitutions.
   (b) $\mathsf{a}_{selfish,2}$: the SWS adjusts (e.g., decrements) the willingness level of the peer to act as a substitute. As a result the peer could be among the last to contact in the future.
2. Collaboration scenario: depending on the criticality level of the composition that the SWS now takes part in, the SWS could take the following actions:
   (a) $\mathsf{a}_{selfish,3}$: the SWS stops recommending the peer for future collaborations.

B) How to deal with untrustworthiness (be:untrustworthy)? Untrustworthiness arises when a peer does not comply with for instance, the non-functional properties that it announces to a SWS. This SWS needs to invoke the functionality of this peer as part of an ongoing composition execution. This lack of compliance will affect the SWS's performance. Examples of untrustworthiness signs and detection mechanisms are discussed in [25]. When a SWS interacts with an untrustworthy peer it could take actions reported in $\mathsf{sc}_{untrustworthy}$:

1. $\mathsf{a}_{untrustworthy,1}$: the SWS restricts data exchange to a minimum level with the peer, e.g., only public data.
2. $\mathsf{a}_{untrustworthy,2}$: the SWS agrees on sanctions on the peer prior to any interaction.
3. $\mathsf{a}_{untrustworthy,3}$: the SWS monitors interactions between the SWS and peer through an independent third-party.
4. $\mathsf{a}_{untrustworthy,4}$: the SWS adjusts (e.g., decrements) the trustworthiness level of the peer. As a result the peer could be among the last to contact in the future.

C) How to deal with unfairness (be:unfair)? Unfairness arises when a failing SWS contacts specific peers for substitution needs and excludes others purposely. Examples of unfairness signs and detection mechanisms are discussed in [25]. When a peer detects unfairness from a SWS it could take actions reported in sc_{unfair}:

1. $\mathsf{a}_{unfair,1}$: the SWS limits the interactions with the SWS to the minimum.
2. $\mathsf{a}_{unfair,2}$: the SWS stops recommending the SWS to others in the future.
3. $\mathsf{a}_{unfair,3}$: the SWS adjusts (e.g., decrements) the fairness level towards the SWS .

In the aforementioned cases the SWS also informs the SN_{auth} of the actions that it has taken so that other members in the network are made aware of these actions as well.

This offers different benefits to the whole network's members such as adjusting the admission policies to the network, comparing reasons of similar actions taken in the past, and ensuring fairness across all the members.

4. Aspects for behaviors' reactions

To write...

5. Conclusion

In this paper we discussed the notion of capacity in the particular context of social Web services. Capacity is decomposed into functional, non-functional, ad social (decomposed as well into operation and behavior). We mainly focussed on the social/behavior capacity to illustrate how a social Web service should react to the behaviors of peers in the same network. A selfish peer might undermine the operation of the social Web service so that recommendations of what to do have been developed. Examples of recommendations are exchanging public data only and suspending interactions temporarily. In term of future work we would like to associate actions in social/behavior capacity with three properties: time-dependent property: the effect of executing actions in social/behavior capacity holds for a limited time-period; Afterwards the effect is canceled automatically; final property: the effect of executing actions in social/behavior capacity is indefinite and hence, cannot be reversed; and scope (local *versus* global) property: the effect of executing actions in social/behavior capacity targets either specific social Web services or all social Web services in a network.

References

[1] J. Al-Sharawneh and M.-A. Williams. A Social Network Approach in Semantic Web Services Selection using Follow the Leader Behavior. In *Proceedings of the 13th Enterprise Distributed Object Computing Conference Workshops (EDOCW'2009)*, Auckland, New Zealand, 2009.

[2] Y. Badr and Z. Maamar. Can Enterprises Capitalize on Their Social Networks? *Cutter IT Journal*, 22(10), October 2009.

[3] S. Bansal, A. Bansal, and M. B. Blake. Trust-based Dynamic Web Service Composition Using Social Network Analysis. In *Proceedings of the International Workshop on Business Applications for Social Network Analysis (BASNA'2010) held in conjunction with the Fourth International Conference on Internet Multimedia Systems Architecture and Applications (IMSAA'2010)*, Bangalore, India, 2010.

[4] B. Benatallah, Q. Z. Sheng, and M. Dumas. The Self-Serv Environment for Web Services Composition. *IEEE Internet Computing*, 7(1), January/February 2003.

[5] W. Chen and I. Paik. Improving Efficiency of Service Discovery using Linked Data-based Service publication. *Information Systems Frontiers, Spinger*, 2012 (forthcoming).

[6] W. Derguech and S. Bhiri. Modelling, Interlining, and Discovering Capabilities. In *Proceedings of the 10th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'2013)*, Fez, Morocco, 2013.

[7] S. Drupad. Enterprise Web 2.0 Worth $4.6 Billion in 2013, Visited September 2008. Searchviews, 21 April 2008, www.searchviews.com/index.php/archives/2008/04/enterprise-web-20-worth-46-billion-in-2013.php.

[8] N. Faci, Z. Maamad, E. Kajan, and D. Benslimane. Research Roadmap for the Enterprise 2.0 Issues & Solutions. *Scientific Publications of the State University Of Novi Pazar Journal, Series A: Applied Mathematics, Informatics & Mechanics*, 2(2), 2014.

[9] L. Kagal, T. Finin, and A. Joshi. Declarative Policies for Describing Web Service Capabilities and Constraints. In *Proceedings of the W3C Workshop on Constraints and Capabilities for Web Services*, Oracle Conference Center, Redwood Shores, CA, USA, 2004.

[10] Z. Maamar, Y. Badr, N. Faci, and Q. Z. Sheng. Realizing a social ecosystem of web services. In A. Bouguettaya, Q. Z. Sheng, and F. Daniel, editors, *Handbook on Web Services - Advanced Web Services Part*. Springer, 2014.

[11] Z. Maamar, F. Faci, K. Boukadi, Q. Z. Sheng, and L. Yao. Commitments to Regulate Social Web Services Operation. *IEEE Transactions on Services Computing, IEEE Computer Society*, 7(2), April/June 2014.

[12] Z. Maamar, N. Faci, L. Krug Wives, H. Yahyaoui, and H. Hacid. Towards a Method for Engineering Social Web Services. In *Proceedings of the IFIP WG8.1 Working Conference on Method Engineering (ME'2011)*, Paris, France, 2011.

[13] Z. Maamar, N. Faci, Q. Z. Sheng, and L. Yao. Towards a User-Centric Social Approach to Web Services Composition, Execution, and Monitoring. In *Proceedings of the 13th International Conference on Web Information System Engineering (WISE'2012)*, Paphos, Cyprus, 2012.

[14] Z. Maamar, H. Hacid, and M. N. Huhns. Why Web Services Need Social Networks. *IEEE Internet Computing*, 15(2), March/April 2011.

[15] Z. Maamar, L. Krug Wives, Y. Badr, S. Elnaffar, K. Boukadi, and N. Faci. `LinkedWS`: A Novel Web Services Discovery Model Based on the Metaphor of "Social Networks". *Simulation Modelling Practice and Theory, Elsevier Science Publisher*, 19(10), 2011.

[16] Z. Maamar, S. Tata, K. Yétongnon, D. Benslimane, and P. Thiran. A Goal-based Approach to Engineering Capacity-driven Web Services. *Knowledge Engineering Review*, 29(2), 2014.

[17] A. Maaradji, H. Hacid, J. Daigremont, and N. Crespi. Towards a Social Network Based Approach for Services Composition. In *Proceedings of the 2010 IEEE International Conference on Communications (ICC'2010)*, 2010.

[18] M. Nam Ko, G. P. Cheek, M. Shehab, and R. Sandhu. Social-Networks Connect Services. *IEEE Computer*, 43(8), August 2010.

[19] M. Paloucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *Proceedings of the First International Semantic Web Conference (ISWC'2002)*, Sardinia, Italy, 2002.

[20] T. Senivongse, N. Phacharintanakul, C. Ngamnitiporn, and M. Tangtrongchit. A Capability Granularity Analysis on Web Service Invocations. In *Proceedings of the World Congress on Engineering and Computer Science (WCECS'2010)*, San Francisco, USA, 2010.

[21] Q. Z. Sheng, X. Qiaob, A. V. Vasilakosc, C. Szaboa, S. Bournea, and X. Xud. Web Services Composition: A Decade's Overview. *Information Sciences*, 2014 (to appear).

[22] B. C. West, D. A. Battleson, J. Kim, and B. Ramesh. Achieving Dynamic Capabilities with Cloud Computing. *IEEE IT Professional*, 16(6), November/December 2014.

[23] Q. Wu, A. Iyengar, R. Subramanian, I. Rouvellou, I. Silva-Lepe, and T. Mikalsen. Combining Quality of Service and Social Information for Ranking Services. In *Proceedings of ServiceWave 2009 Workshops held in conjunction with the 7th International Conference on Service Service-Oriented Computing (ICSOC'2009)*, Stockholm, Sweden, 2009.

[24] X. Xie, B. Du, and Z. Zhang. Semantic Service Composition based on Social Network. In *Proceedings of the 17th International World Wide Web Conference (WWW'2008)*, Beijing, China, 2008.

[25] H. Yahyaoui, Z. Maamar, E. Lim, and P. Thiran. Towards a Community-based, Social Network-driven Framework for Web Services Management. *Future Generation Computer Systems*, 29(6), 2013.