Why Web Services Need Social Networks

Zakaria Maamar • *Zayed University, Dubai*
Hakim Hacid • *Alcatel-Lucent Bell Labs, Paris*
Michael N. Huhns • *University of South Carolina*

Service-oriented architecture (SOA) and its flagship implementation technology known as Web services have changed the way software engineers design and develop today's enterprise applications. Web services help organizations maintain an operative presence on the Internet. Acting as building blocks that can provide and transform data, Web services connect together to create new on-demand value-added composite services.[1] Although SOA practitioners advocate regularly for its benefits, SOA's current state doesn't really sustain these benefits: current SOA applications are designed primarily for closed environments, are static at runtime, and rely mainly on formal features and methods. The use of social networks of services can remove these limitations.

Web Services Fall Short of Their Potential

The current state of SOA has limited the widespread use of Web services (or services complying with SOA principles in general) because several important issues remain unresolved, including where to advertise services for better and immediate exposure, how to discover services with respect to user needs, how to trust services when they're found, and how to smoothly replace services when they fail. These issues have made services fall short of their potential because services

- know only about themselves, not about their users or peers;
- limit users' intervention considerably and operate as black boxes;
- consider only their own internal functional and nonfunctional details during execution and ignore other external details, such as past user interactions;
- can't delegate their invocations;
- don't instantaneously and voluntarily cooperate with each other or self-organize; and
- can't reconcile ontologies among each other or with their users.

As pointed out in other work,[2] current incarnations of Web services are impractical and almost unusable, except in carefully controlled corporate environments. The problem is in the semantics used to characterize services: the semantics in WSDL service descriptions, or in proposed extensions to WSDL, are inadequate for automated discovery. It was suggested[2] that a user community might be able to provide semantic descriptions via a Wikipedia-like effort. However, for appropriate and useful services to be discovered and engaged, sufficiently precise semantics for describing services must be combined with sufficiently intelligent software for understanding the semantic descriptions. Moreover, better descriptive semantics improves only the discovery part of the practical service problem, not the runtime execution part.

Fortunately, behavioral semantics are available in how services are used and combined, and how they behave and interact. This is the social aspect of services. Just as marshaling a social community of users could result in improved semantic descriptions of services, a similar marshaling of the runtime behaviors of socially enhanced services can be exploited to meet runtime execution objectives.

The following example illustrates how services are used with no reference to social elements. To find the definition of a word in English, translate it into French, and then email it, Alice creates a mashup consisting of `Dictionary`, `Translator`, and `PostTwitter` services. To find the weather forecast, translate it

 Published by the IEEE Computer Society 1089-7801/11/$26.00 © 2011 IEEE

into French from English, and post it on her Twitter profile, Alice creates another mashup using `Weather`, `Translator`, and `PostTwitter` services. Here, Alice is using `Translator` for the second time without questioning how it behaved/operated in the presence of `Translator` when it was used for the first time. Indeed, she connected `Translator` and `PostTwitter` again without paying attention to the outcome of the first connection — was it successful or not? This detail isn't reported anywhere unless Alice decides to keep track of everything that she did, which is neither appropriate nor doable. Could `Translator` "step in" smoothly during the mashup development to advise Alice not to use `PostTwitter`, for example? Another user, Bob, wants to create a mashup that finds the weather description for his city so that he can post it on his blog. He uses the following services: `MyLocation`, `Weather`, `Translator`, `BlogPost`, `tinyURL`, and either `PostTwitter` or `Email`. Suppose Carol also wants to create a new mashup based on a weather forecast service. Could she benefit from any of Bob's service mashups? Traditionally, the answer has been "no." Successful service invocations and compositions aren't saved for later use, nor are unsuccessful ones.

Services are treated as isolated components despite their previous interactions with other peers when complex services are built. Capturing service interactions using, for example, social networks could be beneficial for software engineers who can capitalize on the known successful interactions as needs arise. The first interaction concerns the selection that led into identifying, in this case, `Weather` over another peer service such as `WeatherForecast`. Both services are in competition because they do the same job, which is to provide weather information. The second interaction concerns the execution dependencies between services that can become recurrent over time. `Translator` and `PostTwitter` have participated in several joint compositions. Finally, the third interaction concerns service reliability. When `PostTwitter` fails, `Email` takes over automatically. If social networks could capture all these interactions, a (SOA-compliant) service would

- recommend the peers with whom it would like to collaborate in case of compositions, such as `Weather` and `Translator`;
- recommend the peers that can substitute for it in case of failure, such as `PostTwitter` and `Email`; and
- be aware of the peers that compete against it in case of selection, such as `Translator` and `TranslatorWS`.

Collaboration, substitution, and competition are some of the links that can connect Web services together. To make full use of these links, we describe in another work some steps that software engineers can adopt when building Web services' social networks:[3] identify these networks' components, analyze Web services' similarities and differences to identify in which networks these Web services can sign up, manage these networks' growth, navigate through these networks to collect necessary details, and maintain these networks in case of changes in Web services.

The Value of Adding Social Networks to Web Services

When enterprises discover and engage Web services for business needs, they're included in service compositions based on both the functionality they offer and the quality of service (QoS) they can guarantee, which implies the need for contracts. However, when consumers engage and compose services, it's much more informal and dynamic, much like how people download iPhone apps. But unlike iPhone apps, which are monolithic and operate independently of each other, Web services are intended to be composed, and their functionality and QoS are interdependent with other services. Moreover, they execute remotely and with some degree of autonomy. Their discovery and subsequent engagement thus become social activities, much like the collaboration and competition supported in social networks.

Social networks exemplify the tremendous popularity of Web 2.0 applications, which help users become proactive; colloquially, we can refer to users now as *prosumers*, providers and consumers at the same time.[4] Prosumers post definitions on wikis, establish groups of interest, and share tips and advice. These various operations illustrate the principles of "I offer services that somebody else might need" and "I require services that somebody else might offer" upon which SOA is built. Service offerings and requests demonstrate perfectly how people behave in today's society, imposing a social dimension on how Web services must be handled in terms of description, discovery, binding, and composition. What if this social dimension is the missing link? It could serve as an additional ingredient to the formal methods that support SOA needs, namely, service description, discovery, binding, and composition.

Weaving social elements into Web service operation means new social Web services (SWSs) that will

- establish and maintain networks of contacts;
- put users either explicitly or implicitly in the heart of their life cycle, enabling additional functionalities through collaboration;
- rely on privileged contacts when needed;

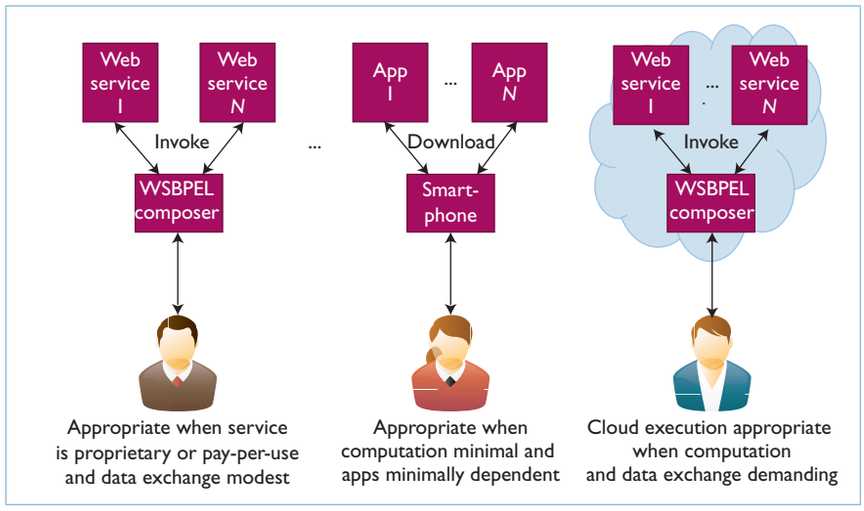

Figure 1. Architectures for service composition and execution. Different execution configurations are suggested depending on the nature of components (Web services or app) involved and users' nonfunctional requirements.

- form with other peers strong and long-lasting collaborative groups; and
- know with whom to partner to minimalize ontology reconciliation.

We see SWSs as the result of blending social computing with service-oriented computing. On one hand, social computing is the computational facilitation of social studies and human social dynamics as well as the design and use of information and communication technologies that consider social context.[5] Social computing is also about collective actions, content sharing, and information dissemination in general. On the other hand, service-oriented computing builds applications on the principles of service offer and request, loose coupling, and cross-organization flow.[6] Blending social computing with service-oriented computing leads to SWSs that "know" with whom they've worked in the past and with whom they would potentially like to work in the future. These two timestamped elements constitute the "memory" of actions that SWSs can accumulate over time and apply in the future. In addition, they show the collective action of a group of SWSs that share respective experiences in response to requests for developing complex value-added composite services. SWSs are expected to take the initiative in advising users how to develop and reuse value-added services.

Social Web Services in Action

In many ways, smartphone apps are like Web services in that they're functional components that are discovered and executed. However, they have significant differences:

- apps are complete and stand-alone;
- apps are owned and executed locally;
- apps aren't composable, except as informal mashups; and
- apps have non-standard APIs.

Figure 1 illustrates some of the differences in the way that Web services and apps are composed and executed, leading to a variety of architectural possibilities. The appropriate choice of architecture is based primarily on QoS, such as the amount of computation required, bandwidth of the data exchange, the system's response time, the user's privacy requirements, and whether the service being provided is proprietary and charged per use. When "socialized," Web services can provide information about how they've behaved and been used in the past. Architecturally, services deployed in a cloud could have their social aspects exploited more easily.

Establishing and maintaining Web services' social networks can happen in three ways:

- *Collaboration*. By combining their respective functionalities, SWSs have the capacity to work together on complex user requests. Consequently, an SWS manages its own network of collaborators, so that it decides if it likes collaborating with peers based on previous experiences. It can also recommend peers.
- *Competition*. SWSs compete against each other when they offer similar functionalities. Their nonfunctional properties differentiate them when users' nonfunctional requirements must be satisfied. Consequently, an SWS learns about its own network of competitors, so that it can attempt to improve its nonfunctional properties with respect to other peers.[7]
- *Substitution*. Although SWSs compete against each other, they can still help each other when they fail if they offer similar functionalities.[8] Consequently, an SWS manages its own networks of substitutes, so that it can meet its service-level agreements (SLAs) when it encounters a potential failure. It can then identify its own best substitutes in response to users' nonfunctional requirements.

These three ways for maintaining social networks can be considered independently as a network of social behaviors. They can be the starting point of building more networks,

Table 1. Web service management strategies.

Comparative elements	Basic	Community (Web 2.0)	Social networks
User level			
Profile	User profile built following regular use of Web services	General profile built for whole community according to use of Web services; this profile is then distributed to all members and customized individually	User profile built following regular use of Web services and social relations that users maintain with others; relations are either explicit or implicit
Web service level			
Description	Web service description developed by provider and then made available to all users	Web service description made available by provider subject to enrichment through annotations by community members and then offered to other members for use; enriched description might suffer from discrepancies	Web service description made available by provider subject to possible enrichment through annotations by members of the same social network, increasing the enriched description acceptance by the rest of this social network
Discovery	Web service discovery after registry screening	Web service collective discovery after registry screening and discovery outcome shared with other community members	Web service discovery after registry screening driven by the needs of each social network's members
Composition	Web service composition driven by individual users familiar with composition techniques and constraints	Web service collective composition driven by some community members; composition outcome shared with other members; community interests prevail over individual interests	Web service composition driven by the needs and previous experiences of each social network's members
Trust	Trust directly established between user and Web service provider	Web service trusted by members of the community based on past experiences; ranking technique can be used	Trust mainly related to the strength of the social relations that users have on top of their experiences of Web service use
Enterprise level			
Advertising	Web service advertisement done by its provider	Web service use supports advertisement, but limited within community boundaries; limited use of Web services because of trust concerns	Web service advertisement taken care of by users via their social contacts; better use of Web services because of trust in these contacts

depending on the interactions that arrive between Web services such as delegation and supervision.

Communities vs. Social Networks of Web Services

Communities can establish connections between Web services.[9] However, a community-based connection offers only a limited view of the activities required in managing Web services. In contrast, a social-based connection offers a wider view by stressing the interactions that occur between users, between Web services, and between users and Web services. Table 1 highlights Web service management by comparing basic strategies to community- and social-based strategies. The comparison criteria include user profiling, Web service description, Web service discovery, Web service composition, Web service advertisement, and trust between Web services and users.

A social network-based strategy for Web service management, as shown in this table, intends clearly to reinforce Web services' performance capabilities through a fine-grained consideration of analysis and reasoning[10] and consideration of "extra" information such as past experiences rather than just information related to Web services. A social network-based strategy offers better exposure, use, and follow-up of Web services compared to basic and community-based strategies. As an example, at the composition level, social networks can include recommendations based on particular users' interests (as well as their immediate social relatives) instead of considering "general" and static behaviors of the services' composition. Another example of the social network-based strategy's strength is

exemplified at the enterprise level by leveraging the diffusion property of a network for a better advertisement of Web services.

Web services have progressed significantly from their inception for addressing business problems to their subsequent democratization to their anticipated socialization. Social networks, with their underlying principles and metrics, can offer innovative solutions to some of the issues Web services face today. The growing number of initiatives reflecting the blend of social computing with service-oriented computing is certainly a positive sign of this area's growing importance.[8,10,11]

References

1. M. Papazoglou et al., "Service-Oriented Computing: State of the Art and Research Challenges," *Computer*, vol. 40, no. 11, 2007, pp. 38–45.
2. C. Petrie, "Practical Web Services," *IEEE Internet Computing*, vol. 13, no. 6, 2009, pp. 94–96.
3. Z. Maamar et al., "Using Social Networks for Web Services Discovery," to be published in *IEEE Internet Computing*, 2011.
4. C. Pedrinaci and J. Domingue, "Toward the Next Wave of Services: Linked Services for the Web Data," *J. Universal Computer Science*, vol. 16, no. 13, 2010, pp. 1694–1719.
5. F.Y. Wang et al., "Social Computing: From Social Informatics to Social Intelligence," *IEEE Intelligent Systems*, vol. 22, no. 2, 2007, pp. 79–83.
6. M.P. Singh and M.N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*, John Wiley & Sons, 2005.
7. M. Alrifai, D. Skoutas, and T. Risse, "Selecting Skyline Services for QoS-based Web Service Composition," *Proc. 19th Int'l World Wide Web Conf.* (WWW 2010), ACM Press, 2010, pp. 11–20.
8. Z. Maamar et al., "LinkedWS: A Novel Web Services Discovery Model Based on the Metaphor of Social Networks," *Simulation Modelling Practice and Theory*, Elsevier Science Publisher, vol. 19, no. 10, 2011, pp. 121–132.
9. L. Chen et al., "Towards a Knowledge-based Approach to Semantic Service Composition," *Proc. 2nd Int'l Semantic Web Conf.* (ISWS 2003), Springer-Verlag, 2003, pp. 319–334.
10. A. Maaradji et al., "Towards a Social Network-based Approach for Services Composition," *Proc. 2010 IEEE Int'l Conf. Communications* (ICC 10), IEEE Press, 2010, pp. 1–5.
11. M. Nam Ko et al., "Social-Networks Connect Services," *Computer*, vol. 43, no. 8, 2010, pp. 37–43.

Zakaria Maamar is a full professor in the College of Information Technology at Zayed University, Dubai, UAE. His research interests include Web services, social networks, and context-aware computing. Maamar has a PhD in computer science from Laval University, Quebec City, Canada. Contact him at zakaria.maamar@zu.ac.ae.

Hakim Hacid is a researcher at Bell Labs France (Alcatel-Lucent). His current research focuses on social interaction analysis to provide added value applications for users and service providers. Before joining Bell Labs, Hacid was a research associate at the University of New South Wales, where he worked with the service-oriented computing group. He has a PhD in computer science from the University of Lyon, France. Contact him at hakim.hacid@alcatel-lucent.com.

Michael N. Huhns holds the NCR Professorship and is chair of the Department of Computer Science and Engineering at the University of South Carolina. He has a PhD in electrical engineering from the University of Southern California. Huhns serves on the editorial boards for 12 journals, is a senior member of the ACM, and is a fellow of IEEE. Contact him at Huhns@sc.edu.