Commitments to Regulate Social Web Services Operation

Zakaria Maamar, Noura Faci, Khouloud Boukadi, Quan Z. Sheng, Member, IEEE, and Lina Yao

Abstract—This paper discusses how social Web services are held responsible for the actions they take at run time. Compared to (regular) Web services, social Web services perform different actions, for instance establishing and maintaining networks of contacts and forming with some privileged contacts strong and long lasting collaborative groups. Assessing these actions’ outcomes, to avoid any violation, occurs through commitments that the social Web services are required to bind to. Two types of commitments are identified: social commitments that guarantee the proper use of the social networks in which the social Web services sign up, and business commitments that guarantee the proper development of composite Web services in response to users’ requests. Detecting commitment violation and action prohibition using monitoring results in imposing sanctions on the “guilty” social Web services and taking corrective actions. A system for commitment management in terms of definition, binding, monitoring, and violation detection is also discussed in this paper.

Index Terms—Social web service, social networking, commitment, monitoring, violation

1 INTRODUCTION

Web services are regularly hailed by IT academics and practitioners for their capacity of developing business processes that can cross organization boundaries at run-time. According to the W3C, 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”. Web services exhibit usually three properties [4]: 1) independent as much as possible from specific platforms and computing paradigms, 2) primarily developed for inter-organization cases, and 3) easy to integrate into existing applications so that developing complex adapters for composition needs is not required.

Over the last few years, the R&D community has looked into Web services from a computation perspective with focus on how Web services are described [40], published [16], discovered [47], composed together [19], made secure [20], made fault-tolerant [37], to cite just a few. Unfortunately other pending issues such as recommendation-based discovery and composition continue to undermine the benefits of adopting Web services as a technology of choice when developing cross-organization business-processes. To address some of these issues, we have been looking into Web services from a new perspective that draws its essence from social computing (exemplified by Web 2.0 applications like social networks and blogs). On different occasions we have demonstrated the synergy between social computing and service-oriented computing (exemplified by Web services) [22], [23], [28], [30]. On the 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 [43]. On the other hand, service-oriented computing is the development of applications upon the principle of “I offer services that somebody else may need” and “I require services that somebody else may offer”. Service offering and requirement illustrate perfectly what people experience (also how they behave) in their daily life (e.g., at work, at home, and so on).

The synergy between social computing and service-oriented computing has resulted into social Web services. Compared to (regular) Web services, they 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 due to ontology and policy disparities are minimized [28]. These new actions are made possible because of the social networks (e.g., collaboration, substitution, and competition) that social Web services can now sign up in. For instance, a social Web service uses a collaboration social network to recommend the peers that it prefers to work with in the case of composition and, also, uses a competition social network to remain aware of the peers that compete against it in the case of selection.

When a social Web service signs up in a social network, it becomes exposed first, to the authority responsible for managing this network and second, to the existing members (i.e., other peers) in this network. This social Web service also avails of the benefits of being a member of the network such as contacting other potential members...
based on the profile they post on the network. Because social Web services can now take different actions whose outcomes might “harm” peers in the same social network (e.g., revealing their private details), or even slowdown the operation of this network (e.g., broadcasting irrelevant details), we hold the social Web services accountable for their actions so that monitoring who did what and when is deemed necessary. To this end, we adopt commitments as a means to first, guarantee the compliance of social Web services with the social networks’ regulations and second, detect any violation of these regulations so that corrective actions are taken promptly. Regulations in a social network cover a wide range of aspects such as privacy, content sharing, registration, and payment. The appropriateness of commitments for social Web services is elaborated in Section 2.3.

Our preliminary research results on commitments to regulate social Web services operation have been reported in [26], where we discuss the general architecture supporting this operation and present some commitment examples. In this paper, we motivate further the adoption of commitments and build upon this architecture to

1. detail two main categories of commitments known as social and business,
2. develop compositions that are driven by commitments,
3. structure commitments and detect their violations,
4. and illustrate the deployment of this architecture using a running scenario.

The rest of this paper is organized as follows. Section 2 is an overview of social Web services and commitments in the literature and a discussion of the motivations to adopt commitments. Section 3 discusses the approach to regulate the operation of social Web services using commitments. The different types of responsibilities that are mapped onto commitments and the different types of sanctions in the case of violation or prohibition are also presented. Section 4 reports on the implementation of a system for commitment management. Finally concluding remarks and future work are reported in Section 5.

2 BACKGROUND

This section provides an overview of social Web services and commitments and concludes with a set of motivations that back the value-added of commitments to regulate social Web services operation.

2.1 Overview of Social Web Services

Social Web services are at the cross-road of two main research streams: social computing and service-oriented computing. Existing research works either adopt Web services to develop social networks of users or develop social networks of Web services to address issues such as recommendation-based discovery of Web services. The blend of social computing with service-oriented computing is quite new. As a result, several research opportunities are still un-taped.

In the category of social networks of users, we cite the research works of Maaradji et al. [31], Xie et al. [45], Al-Sharawneh and Williams [1], Wu et al. [44], Nam Ko et al. [5], and Bansal et al. [3]. Maaradji et al. propose a social composer (a.k.a SoCo) that advises users on the next actions to take in response to specific events like selecting specific Web services. Simply put, users check what their friends in a network did in the past so that they do the same if appropriate. Xie et al. introduce a framework for semantic service composition based on social networks. Al-Sharawneh and Williams mix semantic Web, social networks, and recommender systems technologies to help users select Web services with respect to their functional and non-functional requirements. Wu et al. rank Web services using non-functional properties and invocation requests at run-time. The popularity of a Web service as analyzed by users is the social element that is used for ranking. Nam Ko et al. discuss the social Web in which a new type of services called “social-networks connect services” help third parties develop social applications without having to build social networks. Last but not least, Bansal et al. examine trust in the context of Web services discovery. On top of Web services’ functional and non-functional properties, users’ trust in the Web services’ providers can be used as an additional selective criterion. Trust rating is assessed using social network analysis, which is the process of mapping and measuring the relationships between a social network’s nodes, for instance providers. This rating reflects the centrality of a provider in a social network using three levels known as degree, betweenness, and closeness.

In the category of social networks of Web services, we cite the works of Maamar et al. [24], [27], [29], [30] and Chen and Paik [9]. In [24], Maamar et al. develop a method to engineer social Web services. Questions addressed in this method include 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. In [27], Maamar et al. discuss the intertwine of social networks of users with social networks of Web services to compose, execute, and monitor composite Web services. Each network provides details that achieve this intertwine and thus, complete these operations. Three components are developed: composer, executor, and monitor. The social composer develops composite Web services considering relations between users and relations between Web services. The social executor assesses the impact of these relations on these composite Web services during execution. Finally, the social monitor replaces failing Web services to guarantee the continuity of these composite Web services at run-time. In [29], Maamar et al. use social networks of Web services to tackle the “thorny” problem of Web services discovery. Different social networks permit to capture the situations (e.g., collaboration and recommendation) that Web services come across at run time. Web services should not be treated as isolated components that respond to queries, only. Contrarily, they 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.

1. Readers are referred to [25] for more details on how to hold social networks responsible for the actions affecting their members, namely social Web services.
Commitments have also emerged as a key concept in modeling e-business systems with the work of Xu et al. [46] and e-contracting applications for the semantic Web with the work of Grosof and Poon [14]. Xu et al. discuss how to identify the agents that will be blamed using interconnected commitments. Grosof and Poon define a contract as a set of business activities, a coordination mechanism that manages these activities between multiple agents, exceptions that are violations of inter-agent commitments, and handlers that manage these violations.

As stated earlier, commitments can be linked to SLAs [17] as well. Indeed SLAs are a form of commitments from a provider to future consumers that a service will be made available for use at a certain level (e.g., response time and security level). Commitments adopted in this paper establish, however, relationships from consumers to providers that the regulations of these providers will be “obeyed”. In case of violation providers take actions against consumers. In SLAs consumers take actions against providers.  

2.3 Motivations
The increasing complexity of business processes calls for more than (regular) Web services that respond to users’ queries, only. Indeed organizations adjust their processes regularly, develop new products and offer new services, engage in partnership relationships, etc. To deal with this complexity, the research community has been examining different ways of enhancing Web services with additional capacities so that they become more responsive to the environment. A first option consists of making Web services aware of the environment using context [10], [21], [32]. A second option consists of making Web services join communities for better exposure [6]. A third option upon which this work is built consists of weaving social elements into Web services operations like discussed in Section 2.1. To ensure the success of this weaving, social Web services need to be held accountable for the actions that they take with respect to the regulations of the social networks in which they sign up. These regulations cover a wide range of aspects such as privacy, content sharing, registration, and payment. Social Web services can take different actions (e.g., establishing and maintaining contacts with peers) whose outcomes might “harm” peers in the same social network (e.g., revealing their private details), or even slowdown the operation of the network (e.g., broadcasting irrelevant details). Different questions raise such as, are these actions taken on purpose, are there guidelines to prevent similar actions from happening, and what are the consequences of these actions? To guarantee social Web services’ full compliance with social networks’ regulations, we adopt commitments to specify and implement the operation of social Web services. The following reasons motivate this adoption:

- Commitments describe well how real life business scenarios occur in terms of negotiation, delegation, cancellation, and discharge, which cannot be handled through simple SOAP-based binding (or invocation) operations.
- Commitments are associated with states that permit to follow up the interaction progress between the parties involved in these commitments known as debtor and creditor. Each state is the result of executing operations.
- Commitments can be used in conjunction with other elements like trust and argumentation to develop complex business-scenarios [12].

3 COMMITMENT-BASED APPROACH FOR SOCIAL WEB SERVICES
This section discusses first, the architecture upon which our commitment-based approach is built. Afterwards it details the commitments in terms of structure and monitoring.
Fig. 1. Commitments for social Web services operation.

3.1 Architecture

Fig. 1 illustrates the architecture of our approach to specify and implement social Web services operation using commitments. The architecture includes two worlds: computational and operational. On the one hand, the computational world hosts a pool of Web services and multiple social networks. These two reside in this world’s service and social levels, respectively. A Web service is referred to as social when it signs up in a social network. On the other hand, the operational world hosts a pool of user requests and different composite social Web services. These two reside in this world’s request and composition levels, respectively. Examples of requests include train booking and currency exchange. The completion of complex requests like vacation planning requires developing composite social Web services that consist of component social Web services.

Web services, social Web services, social networks, user requests, and composite social Web services are the constituents upon which our approach is built. Interactions between some of these constituents lead into establishing and managing commitments. We identify two types of commitments: social and business. Social commitments are confined into the borders of the computational world and arise when Web services sign up in specific social networks. These commitments hold the social Web services responsible for the actions that they take when using the social networks as a communication platform to reach out to other peers in these networks. This should guarantee the compliance of the social Web services with the regulations of the corresponding networks, otherwise corrective actions like sanctions are taken. Contrarily business commitments connect the two worlds and arise when social Web services take part in compositions. User requests are assigned to the social Web services for processing. These commitments hold the social Web services responsible for the actions that they take when taking part in ongoing compositions. This should guarantee the participation of these social Web services in compositions and proper behaviors at run time. Contrarily to regular composition scenarios [2], [15], [32], we rely on social networks to let social Web services recommend the peers that they would like to collaborate with in the case of composition, recommend the peers that can substitute for them in the case of failure, and be aware of the peers that compete against them in case of selection [28]. In general, a Web service can sign up in the following three social networks:

- Collaboration social networks. By combining their respective functionalities, social Web services have the capacity to work together on complex user requests. Consequently, a social Web service manages its own network of collaborators, so that it decides if it likes collaborating with peers based on previous experiences. A social Web service can also recommend peers to join under-development compositions.
- Substitution social networks. Although social Web services compete against each other, they can still help each other when they fail as long as they offer similar functionalities. Consequently, a social Web service manages its own networks of substitutes, so that it can meet its SLAs when failure occurs. It can then identify the best substitutes in response to users’ non-functional requirements.
- Competition social networks. Social Web services 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 social Web service learns about its own network of competitors, so that it can attempt to improve its non-functional properties with respect to these competitors [2].

3.2 Social Commitments Towards Social Networks

Social commitments are usually responsibilities contracted by one agent (called debtor) towards another (called creditor), raising the expectation that the debtor will act to satisfy these responsibilities [8]. We first, identify the responsibilities that a social Web service binds to after signing up in a social network and then, discuss how the commitments related to these responsibilities are modeled, managed, and enforced in order to avoid sanctions.

3.2.1 Responsibility Definition

In the following we assume that each network is led by an authority component (\(\text{sn}_{\text{auth}}\)) that does many things such as connect new Web services to existing members in the network, assess the weights of edges connecting the Web services, and enforce the regulations of the network.

To address the lack of relevant works on social networks of Web services, we looked at how users’ rights and responsibilities are specified in some online social applications like Facebook\(^3\) and LinkedIn\(^4\) so that we draw “similar” rights and responsibilities for social Web services. Moreover, in preparation for linking responsibilities to commitments (Section 3.2.2), we represent each Responsibility (\(\text{Resp}\)) with three elements: 1) either obligation or permission\(^5\), 2) actions to perform, and 3) possible conditions that authorize the execution of actions. Our proposed responsibilities are listed below:

- \(\text{Resp}_1\), Collecting any detail \(d\) in a social network would require indicating the purpose \(p\) of this collection to this detail’s owner \(o\), represented as \(\text{Permission(\text{Collect}(d,o,\text{valid}(p)))}\). \(\text{Collect}\) is the action, \(d\) is for instance a non-functional property like response time and is either public (made

---

4. www.linkedin.com/static?key=privacy_policy&trk=hb_ft_priv
5. Readers are referred to [34] for additional illustrations on obligation, permission, and prohibition use.
available to all members of a social network, protected (made available to the social network’s authority component, only), or private (not available), o is the owner of d for instance social Web service, p is the rationale of collecting d, and valid is a function that checks p. To identify the different purposes that requestors can use we studied the Platform for Privacy Preferences Specification (P3P). This resulted in identifying two purposes: collaboration (col) to support the development of composite Web services and substitution (sub) to support the execution continuity of Web service-based business processes in the case of failure.

- Resp2. Posting any detail (d) on a social network should be correct, represented as Obligation(Post(d,true)). Post is the action and true is the veracity of d.
- Resp3. Collecting any detail (d) from a social network should not be tampered afterwards, represented as Obligation(not-Tamper(d,o, collection(d))). not-Tamper is the action and collection is a function that checks if collecting d is approved in compliance with Resp1.
- Resp4. Signing off from a social network would require the completion of all the pending assignments (ass), represented as Permission(Sign-off(status(ass))). Sign-off is the action and status is a function that assures the progress (e.g., ongoing, complete, and failed) of ass.
- Resp5. Revealing any public detail (d) to the non-members (not(m)) of a social network should not be authorized indefinitely, represented as Obligation(not-Reveal(d,o,m, collection(d))). not-Reveal is the action, m corresponds to the non-members of a social network, and collection is a function that checks if collecting d is approved in compliance with Resp1.

3.2.2 Commitment Structure and Management

Largely studied in the multi-agent community [8], [41], Fornara and Colombetti note that "...intuitively a social commitment is made by an agent (the debtor) to another agent (the creditor), that some fact holds or some action will be carried out (the content)" [13]. Conditions can also be linked to commitments resulting in conditional commitments. Fig. 2 shows the life cycle of a commitment proposed in [41] using null, conditional, base, satisfied, and violated as states and create, detach, discharge, cancel, and expire as operations that change commitments’ states. We first, propose a structure for our commitments and then, describe their management with respect to the interactions that occur between debtors (e.g., social Web service) and creditors (e.g., social network’s authority component). violated state is detailed in Section 3.2.3.

Structure. We adopt the formalism of Fornara and Colombetti to structure our commitments with respect to the aforementioned list of responsibilities (Resp1...5). This formalism is as follows [13]: 

\[ C_{Resp}(debtor, creditor, content||condition) \]

where \( C_{Resp} \) is a commitment associated with Resp and [ ] means optional. For the sake of simplicity, we map a responsibility onto one commitment, only.

- \( C_{Resp}(swsi,swsj,\{collect\}(d,swsj)|valid(p_d)) \) is a conditional commitment by swsi to swsj, if valid(p_d) holds then Collect(d,swsj) will be satisfied.
- \( C_{Resp}(swsi,swjauth, Post(d_{act}) \) is a commitment by swsi to swjauth that Post(d_{act}) will be satisfied. self refers to swsi.
- \( C_{Resp}(swsi,swsj,not-Tamper(d_{public},swsj) |collection(d_{public}) \) is a conditional commitment by swsi to swsj that if collection(d_{public}) holds then not-Tamper(d_{public},swsj) will be satisfied.
- \( C_{Resp}(swsi,swjauth, Sign-off(status(ass))) \) is a conditional commitment by swsi to swjauth, if status(ass) holds then Sign-off() will be satisfied.
- \( C_{Resp}(swsi,swjauth, not-Reveal(d_{public},o, nm) |collection(d_{public}) \) is a conditional commitment by swsi to swjauth that if collection(d_{public}) holds then not-Reveal(d_{public},o,nm) will be satisfied.

Management. We adopt the life cycle of Fig. 2 to illustrate commitment changes in response to, first, the interactions that occur between debtors and creditors and, second, the monitoring that creditors perform over the commitments to assess either their satisfaction or their violation at run-time.

Using Fig. 2, we consider a Web service (wsi) as a debtor and a social network through either its authority component (swjauth) or one of its members (swsj) as a creditor. We define the interactions between debtors and creditors with performatives like those defined in the speech act theory [38]: performative(from sender;to:receiver;content:action;statement). Performative is a speech act that the sender uses to express either 1) a directive (e.g., request and advice) that makes the receiver take a particular action, 2) commissive (e.g., promise) that makes the sender commit to some action in the future, or 3) assertive that makes the sender commit to the truth of a statement.

Initially wsi and swjauth exchange request (from : wsi; to : swjauth; content : sign up(reasons)) and inform (from : swjauth; to : wsi; content : approval/refusal) where reasons include supporting arguments (e.g., high reputation/trust level) for wsi to join the network. In the case of approval (criteria like improving the welfare of all members of a social network or arguments given by wsi) can be used to either approve or reject
commitments to regulate social web services operation

We list in the following how the five previous commitments \((C_{Resp_i})\) are subject to either violation, prohibition, or both. We associate violation and prohibition with commitment content and condition, respectively \((C_{Resp}(debtors, creditors, content|condition))\). We also associate compensations and sanctions with creditors and debtors, respectively.

- \(C_{Resp_i}\): violation arises when collection occurs over a non-public detail, and prohibition arises when the purpose of a detail collection is neither composition nor substitution.
- \(C_{Resp_i}\): violation arises when incorrect details are posted on a social network.
- \(C_{Resp_i}\): violation arises when a collected detail is tampered, and prohibition arises when a detail is collected without approval.
- \(C_{Resp_i}\): violation arises when signing off from a social network happens without approval, and prohibition arises when an assignment is left incomplete.
- \(C_{Resp_i}\): violation arises when a detail is revealed to a social network’s non-member, and prohibition arises when a detail is collected without approval.

Whether violation or prohibition, monitoring the actions of social Web services is required. We assign the monitoring to the social network’s authority component \((snauth)\) as well as to the members \((swsj)\) of this network. For each commitment we suggest ways of monitoring and the compensation and sanction operations in response to the violations or prohibitions:

\(C_{Resp_i}\):

- Violation monitoring requires that \(swsj\) reports to \(snauth\) recurrent, tentative accesses to its non-public details from \(swsj\). If these tentative are confirmed using logs for example, this will be a violation to accessing non-public details on \(swsj\). Sanctions consist of reviewing the trust/reputation levels of \(swsj\) if first time. Otherwise, eject \(swsj\) from the social network if these levels go below a threshold.
- Prohibition monitoring requires that \(snauth\) checks if \(swsj\) was really used either as a component in an under-development composition or as a substitute in an under-execution composition for the purpose that \(swsj\) mentioned to \(snauth\) so that it collects details on \(swsj\). If \(swsj\) was not used as expected, this would be a prohibition to collecting details on \(swsj\). Compensations include informing \(swsj\) of what happened as well as giving it more access privileges like tracking all the peers that request its details.

\(C_{Resp_i}\):

- Violation monitoring requires that \(snauth\) checks the veracity of the self-details that \(swsj\) posts. To this end it tracks \(swsj\’s\) operations over time. If this veracity is not confirmed, this will be a violation to posting valid details. Compensations include forcing \(swsj\) to review its details.

### 3.2.3 Violation and Prohibition Handling

Fig. 2 includes violated state to indicate that social Web services might not honor the commitments that they bind to for reasons such as being malicious or temporary shortage of computation resources. Detecting violations is possible using monitoring [33], which can be coupled to compensation in order to take corrective actions. Besides commitment violation, it may happen that social Web services carry out actions that are prohibited. This raises the importance of setting sanctions like decrementing their reputation levels and/or revoking some access privileges.

"permit between first, nature, this requires additional performative exchanges commitments take on now referred to as social Web service swsi to the desire that conditional

Fig. 2 includes violated state to indicate that social Web services might not honor the commitments that they bind to for reasons such as being malicious or temporary shortage of computation resources. Detecting violations is possible using monitoring [33], which can be coupled to compensation in order to take corrective actions. Besides commitment violation, it may happen that social Web services carry out actions that are prohibited. This raises the importance of setting sanctions like decrementing their reputation levels and/or revoking some access privileges.

We list in the following how the five previous commitments \((C_{Resp_i})\) are subject to either violation, prohibition, or both. We associate violation and prohibition with commitment content and condition, respectively \((C_{Resp}(debtors, creditors, content|condition))\). We also associate compensations and sanctions with creditors and debtors, respectively.

- \(C_{Resp_i}\): violation arises when collection occurs over a non-public detail, and prohibition arises when the purpose of a detail collection is neither composition nor substitution.
- \(C_{Resp_i}\): violation arises when incorrect details are posted on a social network.
- \(C_{Resp_i}\): violation arises when a collected detail is tampered, and prohibition arises when a detail is collected without approval.
- \(C_{Resp_i}\): violation arises when signing off from a social network happens without approval, and prohibition arises when an assignment is left incomplete.
- \(C_{Resp_i}\): violation arises when a detail is revealed to a social network’s non-member, and prohibition arises when a detail is collected without approval.

Whether violation or prohibition, monitoring the actions of social Web services is required. We assign the monitoring to the social network’s authority component \((snauth)\) as well as to the members \((swsj)\) of this network. For each commitment we suggest ways of monitoring and the compensation and sanction operations in response to the violations or prohibitions:

\(C_{Resp_i}\):

- Violation monitoring requires that \(swsj\) reports to \(snauth\) recurrent, tentative accesses to its non-public details from \(swsj\). If these tentative are confirmed using logs for example, this will be a violation to accessing non-public details on \(swsj\). Sanctions consist of reviewing the trust/reputation levels of \(swsj\) if first time. Otherwise, eject \(swsj\) from the social network if these levels go below a threshold.
- Prohibition monitoring requires that \(snauth\) checks if \(swsj\) was really used either as a component in an under-development composition or as a substitute in an under-execution composition for the purpose that \(swsj\) mentioned to \(snauth\) so that it collects details on \(swsj\). If \(swsj\) was not used as expected, this would be a prohibition to collecting details on \(swsj\). Compensations include informing \(swsj\) of what happened as well as giving it more access privileges like tracking all the peers that request its details.

\(C_{Resp_i}\):

- Violation monitoring requires that \(snauth\) checks the veracity of the self-details that \(swsj\) posts. To this end it tracks \(swsj\’s\) operations over time. If this veracity is not confirmed, this will be a violation to posting valid details. Compensations include forcing \(swsj\) to review its details.
C_{\text{Resp}}:

- Violation monitoring requires that snauth checks how swsj uses the details it has collected from swsj. If the integrity of these details is not maintained, this will be a violation to collecting details on swsj. Compensations include either posting/refreshing new/existing details on swsj.

- Prohibition monitoring requires that swsj reports to snauth any detail that was collected by swsj without its approval. If this is confirmed using logs for example, this will be a prohibition to collecting details on swsj. Sanctions include revisiting the trust/reputation levels of swsj if first time. Otherwise, eject swsj from the social network if these levels go below a threshold.

C_{\text{Resp}}:

- Violation monitoring requires that snauth checks all the sign-off activities of swsj in a social network. If a sign off happens without approval, this will be a violation to leaving the social network. Compensations include enforcing the sign-off regulations and informing other peers of these regulations.

- Prohibition monitoring requires that snauth makes sure that all the assignments of swsj are complete according to a specific time frame. If this is not the case, this will be a prohibition to completing these assignments. Sanctions include revisiting the performance level of swsj first time. Otherwise, eject swsj from the social network if this level goes below a threshold.

C_{\text{Resp}}:

- Violation monitoring requires that snauth checks all the details that swsj posts on its social network on swsj that it is not part of this network making sure that swsj has the necessary approvals from swsj. In case these approvals are missing, this will be a violation to revealing details on swsj. Compensations include informing swsj of what happened as well as giving it more access privileges like tracking all the peers that request its details.

- Prohibition monitoring requires that swsj reports to snauth any detail that was collected by swsj without its approval. If this is confirmed using logs for example, this will be a prohibition to collecting details on swsj. Sanctions include revisiting the trust/reputation levels of swsj if first time. Otherwise, eject swsj from the social network if these levels go below a threshold.

### 3.3 Business Commitments Towards Compositions

Similar to how we analyzed social commitments in the computational world in terms of responsibility definition, commitment structure and management, and violation and prohibition handling, we proceed with the same for business commitments connecting the computational world to the operational world. These commitments have social Web services as debtors and compositions as creditors. We define the future responsibilities of social Web services when they engage in compositions in accordance with their already-defined responsibilities in the social networks.

#### 3.3.1 Responsibility Definition

We assume that each composition is driven by an orchestration component (comorch) that does among other things look for the necessary social Web services to append into the composition, fix the semantic mismatches between the social Web services in the composition, trigger the execution of the social Web services in the composition, and process the recommendations of social Web services in terms of expanding the composition with new peers or substituting the failing social Web services with other peers. We define a responsibility with three elements: obligation, actions to perform, and possible conditions that authorize the execution of actions. Our proposed responsibilities are listed as the following:

- **RespðdÞ.** Exchanging any detail (d) in a composition should require indicating the expected use (u) of this detail by the recipient (c). A detail corresponds to any data reported in a business process that a composition implements. As per our previous work on Web services’ control and operational behaviors [40], we identify two potential uses for a detail, namely control and operational behavior building. This responsibility is represented as Obligation(Submit(d, c, valid(u))). Submit is the action, d is for instance a data like price and is made available to the orchestration component and other members of a composition, c is the recipient for instance a peer, u is the rationale of exchanging d, and valid is a function that checks u.

- **RespðtÞ.** Sharing any detail (d) with all components of a composition including the orchestration component should be correct. This responsibility refers somehow to Resp2 where a social Web service is expected to post valid details on a social network, represented as Obligation(Share(d, true)). Share is the action and true is the veracity of d.

- **RespðrÞ.** Recommending a social Web service (sws) to the orchestration component should require stating its profile (f) in terms of functional and non-functional properties and role (r) in a composition either as a collaborator or as a substitute. This responsibility refers somehow to Resp1 since collecting details on a social Web service for possible participation in a composition requires its permission, represented as Obligation(Recommend(comorch, sws, f, valid(r))). Recommend is the action, sws is the recommended peer that comorch will either append into the composition or replace a failing component in the composition, f is the profile of sws, r is the role of sws, and valid is a function that checks r.

- **RespðsÞ.** Fulfilling the pending assignments (ass) of a failing peer should be taken care by the substitute social Web service (sws). RespðsÞ is a function that checks s.
post-effect to \( \text{Resp}_u \) fulfillment, represented as \\
\( \text{Obligation}(\text{Fulfill}(sws; ass)) \). \( \text{Fulfill} \) is the action, \( sws \) is the substitute social Web service that will join an ongoing composition, and \( ass \) is the list of pending assignments of the failing peer.

- \( \text{Resp}_{10} \). Guaranteeing the “proper” functioning (\( \text{func} \)) of a potential collaborator social Web service (\( sws \)) should be done by the recommending peer so that other peers’ operation continuity and safety are maintained. It is represented as \\
\( \text{Obligation}(\text{Guarantee}(sws; func)) \). \( \text{Guarantee} \) is the action, \( sws \) is the new social Web service that will be appended into an ongoing composition, and \( func \) represents its operation.

### 3.3.2 Commitment Structure and Management

We propose a structure for our business commitments and then, describe their management with respect to the interactions that occur between debtors (e.g., social Web service) and creditors (e.g., composition’s orchestration component).

**Structure.** We adopt the same formalism used earlier namely \\
\( \text{C}_{\text{Resp}}(\text{debtor}; \text{creditor}; \text{content}||\text{condition}) \).

- \( \text{C}_{\text{Resp}}(swsj; swsi; \text{Submit}(d, swsj); \text{valid}(udj)) \) is a conditional commitment by \( swsj \) to \( swsi \), that if \( \text{valid}(udj) \) holds then \( \text{Submit}(d, swsj) \) will be satisfied.
- \( \text{C}_{\text{Resp}}(swsj; swsj; \text{comporch} \odot swsj; \text{Share}(dself)) \) is a commitment by \( swsj \) to \( \text{comporch} \) that \( \text{Share}(dself) \) will be satisfied. \( \text{self} \) refers to \( swsj \).
- \( \text{C}_{\text{Resp}}(swsj; swsj; \text{comporch}; \text{Recommend}(\text{comporch}; swsj; f); \text{valid}(rswsj)) \) is a conditional commitment by \( swsj \) to \( \text{comporch} \) that if \( \text{valid}(rswsj) \) holds then \( \text{Recommend}(\text{comporch}; swsj; f) \) will be satisfied.
- \( \text{C}_{\text{Resp}}(swsj; swsj; \text{Fulfill}(swsj; assswsj)) \) is a commitment by \( swsj \) to \( swsj \), that \( \text{Fulfill}(swsj; assswsj) \) will be satisfied.
- \( \text{C}_{\text{Resp}}(swsj; swsj; \text{Guarantee}(swsj; func); \text{valid}(funcswsj))) \) is a commitment by \( swsj \) to \( swsj \), that \( \text{Guarantee}(swsj; funcswsj) \) will be satisfied.

**Management.** We consider this time a social Web service (\( sws \)) as a debtor and a composition’s orchestration component (\( \text{comporch} \)) as a creditor. The interactions between debtors and creditors are defined with performatives and are driven by the needs to either collaborate or substitute. For illustration we only discuss the collaboration interactions and assume that \( sws \) is already part of a composition.

Initially \( swsj \) and \( \text{comporch} \) exchange request (from : \( swsj \); to : \( \text{comporch} \); content : \( \text{recommend}(swsj) \)) and inform (from : \( \text{comporch} \); to : \( swsj \); content : \( \text{approval}/\text{refusal} \)). \( swsj \) suggests adding \( swsj \) to the composition led by \( \text{comporch} \). Upon approval, \( \text{comporch} \) makes \( swsj \) a member of the composition and informs other existing members if necessary. As a first, direct result \( swsj \) has to comply with all the responsibilities of this composition in terms of exchanging details and recommending collaborators and substitutes. The compliance begins when \( \text{comporch} \) executes create operation so that all the commitments related to \( swsj \) take on conditional state (for non-conditional commitments like \( \text{C}_{\text{Resp}}, \) base state is automatically taken after conditional state). Further to the conditional state that some commitments take, other states are taken as well as per the following cases:

1. If a commitment’s condition holds, e.g., \( \text{valid}(udj) \), then \( \text{detach} \) operation is executed making this commitment takes on base state. Different ways exist to assess the conditions of commitments depending on the nature of responsibilities.
   - \( \text{Resp}_u \): \( swsj \) checks that \( u \) is either for control-behavior building or for operational-behavior building.
   - \( \text{Resp}_u \): \( \text{comporch} \) checks that \( r \) is either collaborator or substitute.

2. If a commitment’s content is satisfied, e.g., \\
\( \text{Guarantee}(swsj; func) \), then \( \text{discharge} \) operation is executed making this commitment takes on satisfied state. Otherwise, \( \text{cancel} \) operation is executed making this commitment takes on violated state.

3. If \( swsj \) fails in a composition, then \( \text{expire} \) operation is executed making all the commitments related to \( swsj \) take on null state.

### 3.3.3 Violation and Prohibition Handling

We list in the following how the five additional commitments (\( \text{C}_{\text{Resp}_{10..}} \)) can be subject to either violation, prohibition, or both. Similar to what we did for the social commitments, we associate first, violation and prohibition with commitment content and condition, respectively and second, compensations and sanctions with creditors and debtors, respectively.

- \( \text{C}_{\text{Resp}} \): violation arises when detail submission occurs without the request of the recipient. And prohibition occurs when a detail is used though the sender of the details did not approve its use.
- \( \text{C}_{\text{Resp}} \): violation arises when incorrect details are exchanged in a composition.
- \( \text{C}_{\text{Resp}} \): violation arises when the profile of a recommended social Web service is not correct. And prohibition arises when a social Web service is appended into a composition though its role is not valid.
- \( \text{C}_{\text{Resp}} \): violation arises when a substitute social Web service does not fulfill the pending assignments of the failing peer.
- \( \text{C}_{\text{Resp}} \): violation arises when a collaborator social Web service is appended into a composition though its proper functioning is not guaranteed.

Whether violation or prohibition, monitoring the actions that social Web services take as components in compositions is required. We assign the monitoring to orchestration components (\( \text{comporch} \)) and social Web services (\( swsj \)) as well. For each commitment we suggest ways of monitoring and compensation and sanction operations in response to the violations or prohibitions.
Violation monitoring requires that \( \text{Comporch} \) checks if the \( \text{swsj} \)'s role is played as expected (i.e., collaboration or substitution). If this is not the case, there will be a prohibition to append \( \text{swsj} \) into the composition of \( \text{Comporch} \). Compensations include informing \( \text{swsj} \) that \( \text{swsj} \) was not appropriate for this composition.

**C_{Resp}':**

- Violation monitoring requires that \( \text{Comporch} \) checks if \( \text{swsj} \) fulfills the pending assignments of \( \text{swsi} \) in the composition. To this end it tracks \( \text{swsj} \)'s operations over time. If this fulfillment is not confirmed, there will be a violation to honoring its assignments. Sanctions include decreasing \( \text{swsj} \)'s trust level so that other peers do not consider it as a potential collaborator in their respective social networks.

4 IMPLEMENTATION

4.1 System Design

Fig. 3 illustrates the architecture that supports install social Web services (in terms of connection to the social networks and interaction with each other) and managing commitments (in terms of creation, update, satisfaction, and deletion). The architecture is a stack of three levels, namely service, social, and monitoring. The first two levels correspond to the levels reported in the computational world in Fig. 1. At the service level, a Web service \( \text{(ws)} \) requests entry to the social network authority \( \text{(snauth)} \) of a social network (at the social level). Upon approval, the \( \text{snauth} \) installs a social Web service \( \text{(sws)} \) that will act on behalf of this \( \text{ws} \) in the network and updates two repositories denoted by commitments \( \text{(C_{Resp})} \) and SWSs details. The former stores commitments so that their statuses are known as per the life cycle of Fig. 2. The latter stores Web services’ functional and non-functional properties and the interactions’ outcomes between the \( \text{sws} \) and the \( \text{snauth} \) as well (e.g., rejecting a sign-off request). These two repositories are shared between the \( \text{snauth} \) and the MonitorManager. The MonitorManager and ObserverManager together form the monitoring level in Fig. 3.

Fig. 4 is the class diagram of the architecture of Fig. 3. Several classes are identified including \( \text{CMonitorManager}, \text{CObsManager}, \text{CObserverManager}, \text{CLifeCycle}, \text{CResp}, \text{CObserverManager} \) deploys instances of \( \text{CObserver} \) so that...
they report to CMonitorManager the events that occur in the social network such as message exchange between a SWs and the snauth, and operations in the social network (e.g., collecting private details). CMonitorManager processes these events together with respect to the instances of CLifecycle to determine when a concrete commitment (associated with CResp class) is created, satisfied or violated, or expired. In Fig. 5, CResp (with Condition and Content) describes a CRespk’s structure (Section 3.2.2). Let us consider $C_{Respk}(swsi, snauth, Post(d_{self}))$ as an example where $swsi$ and $d_{self}$ are variable names (i.e., any social Web service and any self-detail, respectively). These latter correspond to possible values for the debtor (resp. varParamA) attribute of $C_{Resp}$ (resp. Content). Upon acceptance of a new Web service ($sws_1$) in the social network, the snauth creates an instance of each $C_{Respk} = (1, 3, 5)$ where $swsi$ is replaced with $sws_1$ and stores the commitments in the commitment repository. $C_{Respk}$ is also associated with expCdt that represents the condition under which the snauth removes a $C_{Respk}$ instance from the repository of commitments. A commitment data-structure is generic; the list of pre-defined commitments can be enriched with no changes in the CMonitorManager functionalities.

Eclipse IDE for Java and PostgreSQL 8.4 were used to implement the system. First, we developed different Graphical User Interfaces (GUIs) to cater for the needs of providers of SNs and providers of WSs. We also decomposed the prototype’s functionalities as follows. The providers of SNs launch the deployment of three instances: CSNAuth, CMonitorManager, and CObserverManager. All these instances are implemented as concurrent threads. A WS provider submits to a SN details on its Web service (ws) like name, business domain, and execution time as well as its WSDL and the reputation level (Section 3.2.2). Then, CSNAuth compares this latter with some thresholds to decide whether it approves the request or not.

To carry out experiments, we generated in two different ways a list of observations (obsList) that corresponds to swss taking social actions as per their $C_{Respk}$ (e.g., Post(responseTime)). The first way is to select these actions through a GUI (Fig. 6). Here, CObservers are implemented as Java-Event Listeners. The second way is to automatically generate the social actions through the corresponding CObserver; for instance “HotelWS would like to collect listOfPlace from PlaceBookingWS for the purpose of collaboration”.

4.2 Experiments

When a social Web service becomes a member of a social network, an administration module is created to manage its life cycle based on the social commitments that it will bind to at run time (Fig. 6). For instance, social actions such as collecting and posting data are invoked through this administration module. As aforementioned, this module simulates CObserver to generate events and report them to CMonitorManager as well. This latter also has an administration module to select the violation/prohibition detection strategy from two alternative options (FIFO-based and priority-based), as well as violation/prohibition handling. These options schedule the handling of violations and prohibitions. The first option considers FIFO screening of the social commitment events with respect to their occurrence order, while the second considers priority screening of the social commitments. Priorities are established according to the social actions and can vary upon user’s request. In the current implementation, priorities establishment occurs as follows:

![Fig. 4. Architecture’s class diagram.](image)

![Fig. 5. Commitment data-structure.](image)
A detail in a user’s profile has a sensitivity level to indicate if permission is required before collection. For instance, a private detail may have a negative impact on the reputation level if leaked.

Social actions are sorted based on sanctions applied on Web services in case of commitment violation/prohibition. For instance, “Reveal private detail” (C.Resp₂) has a higher priority than “Collect public detail” (C.Resp₁). Violating C.Resp₃ (“Tamper details”) corresponds to more serious sanctions than C.Resp₄ (“Sign Off with pending assignments”) and C.Resp₂ (“Post invalid details”).

In the following, we illustrate some cases of violation and prohibition. We first consider that HotelWeb is about to collect data on PlaceBookingWeb where “Collect data” is a social action related to Resp₁ (i.e., “Collecting any detail (d) in a social network would require indicating the purpose (p) of this collection to this detail’s owner (o’).”). Let’s assume that d is listOfPlace and p is “collaboration”. As listOfPlace is private, unauthorized peers should not collect it even if p is valid. CMonitorManager updates the list of commitment status with this violation (Fig. 7). In Fig. 7, CMonitorManager also reports another violation related to C.Resp₅ (“Revealing any public detail (d) to the non-members (not(m)) of a social network should not be authorized”). In this case, WeatherWeb tries to reveal a GPS localization to BBCWeather that is a non-member of the network. This violation is ranked first in the priority list of CMonitorManager, since C.Resp₅ has a high priority than C.Resp₁.

Still in Fig. 7 that shows C.Resp₁’s violation because of the private nature of listOfPlace and a prohibition because HotelWeb is neither a component in the under-development composition that includes PlaceBookingWeb nor a substitute for PlaceBookingWeb. These violation and prohibition are reported from CMonitorManager to the SN_Auth.

As per Fig. 6, HotelWeb is not engaged in any composition. In the second part of experiments, we engage HotelWeb in a composition referred to as TravelPlanner. The following describes briefly TravelPlanner in terms of component Web services along with some responsibilities converted into business commitments. On top of HotelWeb for hotel booking, it needs other components including FlightWeb for airline reservation, CarRental or TaxiWeb for...
ground transportation, CurrencyWS for currency exchange, WeatherWS for weather forecast, and BillWS for billing. HotelWS collects accommodation deals for the periods requested by customers. It can also suggest redeeming loyalty points in return of free-of-charges transport. WeatherWS returns weather forecast for the said periods so that either indoor activities like museum visits or outdoor activities like cruises are planned. BillWS proposes different payment options to customers. In case of credit card payment, BillWS contacts PaymentWS to authorize payment. For experimental needs, we assume that AccommodationWS is a substitute for HotelWS.

A customer through PaymentWS agrees to pay if the trip is planned and booked properly. After receiving the payment agreement from the customer, PaymentWS notifies FlightWS and HotelWS of the payment authorization. Following HotelWS’s unexpected failure due to server crash, AccommodationWS is a good candidate to substitute HotelWS so that the same quality of service is maintained. Before recommending any peer using HotelWS’s substitution social network and as per C_Resp9, HotelWS has the obligation to state this peer’s profile and role to the orchestration component in charge of executing TravelPlanner (Fig. 8a). HotelWS collects details on AccommodationWS with its permission as per C_Resp1 (a social commitment). The purpose of this collection is substitution. C_Resp1 is satisfied if AccommodationWS accepts to substitute HotelWS in TravelPlanner. In the same way, C_Resp9 is satisfied if the role stated to the orchestration component corresponds effectively to the purpose of this collection. Upon acceptance, AccommodationWS has the obligation to resume all the pending assignments of HotelWS (Resp9). AccommodationWS is thus committed to fulfilling this obligation (C_Resp9). However, AccommodationWS delays its participation due to other higher priority requests (Fig. 8b). Thus, violating C_Resp9 affects negatively its reputation causing the cancellation of hotel booking due to delay in confirming the booking. A potential side-effect is that FlightWS will be constrained to cancel the flight and request refunds with financial penalties.

5 Conclusion

In this paper, we have presented an approach to regulate social Web services operation using commitments. Two types of commitments have been identified: social and business. The former focus on the membership regulations of Web services in social networks so that these Web services can be referred to as social Web services. The latter focus on the participation of social Web services in compositions. Commitments have been deemed appropriate because of the actions that social Web services carry out compared to (regular) Web services such as establishing and maintaining networks of contacts, counting on their contacts when needed, and forming with other peers strong and long lasting collaborative groups. Each commitment is structured using three elements: debtor, creditor, and content that could be subject to conditions.

Monitoring violation and prohibition on commitments has also been examined in this paper. The objective is to develop compensations and sanctions in response to these violations and prohibitions, respectively. A system implementing the techniques proposed in this paper is reported. In particular, we discuss the commitment management in terms of definition, binding, monitoring, and violation detection.

There are several research directions in our future work. Firstly, we plan to develop techniques for checking the consistency of commitments so that deadlocks or conflicts are avoided. Secondly, we will identify additional commitment violation and action prohibition types. Thirdly, we plan to draw rights and responsibilities from real networks of social Web services. We also plan to develop strategies enabling the “guilty” social Web services to argue about their actions prior to sanctioning them.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that helped improve the quality of this paper.
Zakaria Maamar received the MSc and PhD degrees in computer sciences from Laval University, Québec, Canada, in 1995 and 1998, respectively. He is a Professor in the College of Information Technology at Zayed University in Dubai, United Arab Emirates. His research interests are primarily related to service-oriented computing, social computing, and system interoperability.

Noura Faci received the PhD degree in computer sciences from Reims University, Reims, France, in 2007. She has been an Associate Professor at the University Lyon 1, Lyon, France, since October 2008. Her research interests are fault tolerance in multi-agent systems, dependable e-business systems, and service computing. Prior, she was a Research Associate at King’s College London, London, U.K. in the Agents and Intelligent Systems Group where she led a work package as part of a CONTRACT European IT Research Project.

Khouloud Boukadi received the PhD degree in computer science from Ecole des Mines, Saint Etienne, France. She is an Assistant Professor in the Computer Science Department at the Faculty of Economics and Management of Sfax, Tunisia. She is a member of the Multimedia Information systems & Advanced Computing Laboratory (Miracl, www.miracl.mn.tn). Her research interests include service computing, Web services in the cloud, context-aware computing, and agility of Information systems.

Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales, Sydney, Australia. He is an Associate Professor in the School of Computer Science at the University of Adelaide, Adelaide, Australia. His research interests include service-oriented architectures, Web of Things, distributed computing, and pervasive computing. He was the recipient of the 2012 Chris Wallace Award for Outstanding Research Contribution and the 2003 Microsoft Research Fellowship. He is the author of more than 130 publications. Dr. Sheng is a Member of the IEEE and the ACM.

Lina Yao is currently a PhD student in the School of Computer Science at the University of Adelaide. She received the Master and Bachelor degrees, all in computer science, from the University of Adelaide, Adelaide, Australia, and Shandong University, Shandong, China, respectively. Her research interests include Web mining, knowledge management, Internet of Things, and service-oriented computing.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.