Towards an Approach for Weaving Preferences into Web Services Operation

Zakaria Maamar
Zayed University, Dubai, U.A.E
Email: zakaria.maamar@zu.ac.ae

Quan Z. Sheng
The University of Adelaide, Adelaide, Australia
Email: qsheng@cs.adelaide.edu.au

Yacine Atif
United Arab Emirates University, Al-Ain, U.A.E
Email: yacine.atif@uaeu.ac.ae

Sujith Samuel Mathew
The University of Adelaide, Adelaide, Australia
Email: sujith@cs.adelaide.edu.au

Khouloud Boukadi
University of Sfax, Sfax, Tunisia
Email: khouloud.boukadi@fsegs.rnu.tn

***Abstract*— Existing approaches on Web services privacy dominate solutions from a users' perspective, giving little consideration to the preferences of Web service providers. The integration of service providers' preferences into Web services' operations is discussed in this paper. A Web service provider indicates peer Web services that it could interact with as well as the data that they could exchange with. We focus on Privacy and (trust) Partnership preferences based on which, we develop a Specification for Privacy and Partnership Preferences (S3P). This specification suggests a list of exceptional actions to deploy at run-time when these preferences are not met. An integration model of these preferences into Web services design is illustrated throughout a running scenario, and an implementation framework proves the S3P concept.**

***Index Terms*—Composition, Partnership, Privacy, Preference, Web service.**

I. INTRODUCTION

Web services play a major role in the development of loosely-coupled business applications that can cross organization boundaries at run-time. This role is witnessed from the widespread adoption of Web services in different initiatives [4, 15, 16, 17, 20, 25]. Composition of Web services handles users' requests that cannot be satisfied by any single, available Web service, which requires combining the available Web services.

In response to the dynamic nature of today's environments, e.g., sudden drop in network bandwidth, mobility of computing resources, and high rate of cyber attacks, we enhanced in the past Web services with mechanisms that allow them for example to reject processing users' requests due to their current heavy loads, and ask for better rewards due to the pressing nature of these requests [13]. In this paper, we continue this enhancement with emphasis on why and how providers of Web services need to express the *preferences* of their Web services. By preference, we refer to the conditions and terms that regulate the proper (and expected as well) use of a Web service. We consider two types of preferences: *partnership* that is geared towards composition, and *privacy* that is geared towards controlling the data flow in composition. As a result, privacy becomes critical when independent Web services are put together in the same composition.

Although there is no substantial research on partnership issues in compositions (issues like semantic disparity and policy incompatibility are assumed properly addressed in this paper), research on privacy issues through the Platform for Privacy Preferences (P3P, www.w3.org/P3P) and the Enterprise Privacy Authorization Language (EPAL, www.zurich.ibm.com/security/enterprise-privacy/epal) initiatives, is still confined to users, only, who interact with Web sites [2, 24, 26]. A user would like to know the purpose of submitting her credit card number to a Web site, how long this Web site will retain this number, how she could verify that this number was really deleted, etc. This way of analyzing privacy overlooks the concerns of

Manuscript received July 3, 2011; revised September 1, 2011; accepted October 29, 2011.
Corresponding author: Z. Maamar, Zayed University, Po Box 19282, Dubai, U.A.E.

doi:10.4304/jsw.7.7.1429-1439

providers of Web services in terms of (i) what data their Web services can receive, (ii) when their Web services can forward data, and (iii) what data their Web services can store. Similar questions can be asked when partnership is analyzed, e.g., with whom Web services can interact and for how long. For illustration purposes, let us assume two Web services s_1 and s_2 along with their respective partnership and privacy preferences. In compliance with these preferences s_1 invokes s_2 during an agreed upon time period (e.g., 2pm-4pm only) and s_1 submits data to s_2 because s_2 guarantees the deletion of these data within 48 hours. If these preferences cannot be satisfied, either s_2 is invited to review its preferences or the search for another peer that will interact with s_1 is initiated.

Previous research on Web services focuses on privacy from a user's perspective and always guarantees the automatic and continuous participation of Web services in compositions. This should not be the case, as discussed in this paper. First, the providers question the data that their Web services consume and exchange. Second, the providers question the compositions that their Web services take part in. The same questions may apply to security issues as well. However, standards in Web service security have been extensively addressed in the literature. We focus in this paper on privacy issues to illustrate the accommodation of preferences in Web services compositions. The approach we propose is extensible to additional preferences. Our contributions are strictly dedicated to Web services and built upon a Specification for Privacy and Partnership Preferences (S3P). S3P uses *tags* to represent partnership preferences of component Web services and a *privacy flow* to represent the restrictions on the data flow between component Web services. Main contributions are summarized as follows:

- Identify arguments that reflect Web services' partnership and privacy preferences.
- Develop a set of corrective actions to take when partnership or privacy preferences are unsatisfied at run-time.
- Provide graphical means to illustrate partnership and privacy preferences during the modeling of component and composite Web services.

The remainder of this paper is organized as follows. Section 2 is an overview of some related work. Section 3 discusses preference integration into Web services operation through the adoption of the S3P. Examples of preference arguments and satisfaction of these arguments are, also, discussed in this section. Section 4 provides a proof of concept to test the feasibility of the S3P. Finally, Section 5 draws some concluding remarks and identifies some future research work.

II. RELATED WORK

Web services provide unique opportunities to extend Web applications dynamically, but face some challenges that compromise their effectiveness to cross organization boundaries and computing platforms [20]. These challenges include automated discovery of services, dynamic service reconfiguration, end-to-end security, privacy, to cite just a few. Our literature review on the particular issue of Web services privacy includes a good number of research projects such as [3, 5, 6, 8, 10, 14, 21, 23, 24, 26, 27, 28]. We found that [21] is the only project that addresses this issue from the perspective of providers of Web services and not from the perspective of users of Web services.

In [3], Benbernou et al. develop a privacy agreement model for Web services. Despite the increasing number of privacy policies that organizations post on their Web sites, individuals are generally reluctant to disclose their personal data to these Web sites. In response to this reluctance, Benbernou et al.'s privacy-agreement model adopts the WS-agreement specification [1], to stress out the importance of defining rights and obligations of users towards organizations.

In [6], Chafle et al. discuss the centralized orchestration of Web services composition with focus on constraints on the data flows in this composition. In this orchestration, the data are routed through a central coordinator that has access to the input/output data of all the component Web services. Chafle et al. note that (i) in certain business scenarios, Web services may have restrictions on the source (resp., destination) of the data they receive (resp., send) and (ii) handling these restrictions using current security mechanisms (encryption, authentication) is sometimes inefficient. The solution of Chafle et al. uses three modules (decentralizer, topology filtering, and deployment) and splits a composite Web service into a set of partitions, one partition per component Web service. A partition is like a proxy that processes, transforms, and manages the incoming/outgoing data in compliance with the restrictions imposed on a component Web service and the data requirements of a composite Web service.

In [8], Hamadi et al. develop privacy-aware protocols for Web services. Like other researchers, they note that (i) Internet users have concerns about their personal data being collected and managed by various organizations, and (ii) a small number of Web sites offer real Web services that could be used to investigate privacy and its impact on Web services acceptance by the IT industry and users. To remedy this lack of real Web services, Hamadi et al. study some B2C Web sites/portals like Amazon.com along with their privacy policy documents. Their response to privacy is a modeling technique (based on state chart) that (i) captures privacy abstractions while describing the operation of a Web service and (ii) weaves these abstractions into this operation.

In [24], Xu et al. note that privacy concerns of users need to be handled while the development of composite Web services is in progress. The number of people who access the Web continues to grow, which has exacerbated these concerns. To address this exacerbation and P3P shortcomings, Xu et al. develop privacy-conscious composite Web services. When a user submits data to a Web service, the user would make sure that these data are managed according to her privacy preferences. To this end, the user requests the model of a Web service so that

she knows how this Web service processes and shares data. In their work, automated techniques check the compliance of a Web service's model with a user's privacy preferences. If the check succeeds, the user forwards her request to the Web service for processing. Otherwise, the user forwards the violation as an obligation to the composite Web service for further actions.

In [27] Liu et al. emphasize that the increased use of Web services has meant that more and more personal information of consumers is being requested and shared with these Web services' providers. Thus it is critical to guarantee that the private data of consumers are collected, used and disclosed according to strict policies. The authors suggest developing a minimal privacy authorization that still permits achieving the functional goals. Authorization policies to specify privacy privileges and trust relationships among services are used.

Although the aforementioned approaches offer a snapshot of the initiatives on Web services' preferences with emphasis on privacy, there is no clear vision that articulates how these preferences should be looked at from the particular perspective of providers of Web services. The work of Rezgui et al. is, to a certain extent, the only one that embraces this perspective by highlighting the concerns of providers in terms of data usage, storage, and disclosure [21]. However questions like what privacy preferences are appropriate for Web services, how these preferences are reviewed in case of no-satisfaction at run time, and how these preferences are modeled, are left unanswered and solutions are provided on a case-by-case basis.

III. PREFERENCE INTEGRATION INTO WEB SERVICES THROUGH S3P

This section consists of three parts. First, we propose some arguments that show Web services' partnership and privacy preferences. Then, we illustrate these arguments using a running example. Finally, we work out an S3P instance of this example based on these arguments.

A. Preference arguments

In Section 1, partnership and privacy are introduced as types of preferences. In the following, we suggest some arguments per type of preference and show how the operation of a Web service is restricted if these preference arguments turn out unsatisfied at run-time. It should be noted that preference arguments should be defined using a dedicated ontology but this is outside this paper's scope.

Partnership preferences are related to the compositions that Web services take part in. Some examples of partnership arguments are as the following:

- *Participation-duration* argument: because Web services can engage in long-running compositions that last days and even weeks [12, 19], a Web service sets the maximum time that it will remain committed to a composition whether this composition is complete or not. By doing this, the Web service disengages automatically from the compositions that last more than expected and participates in other compositions should this become possible.
- *Invocation-period* argument: to maintain a certain level of QoS [18, 22], a Web service sets different time periods (e.g., off peak, peak) to process requests. These periods are based on business hours, computing resources availabilities, etc.
- *Payment-mode* argument: in return to processing requests, a Web service is compensated either (i) instantly after these requests are complete or (ii) deferred until the successful completion of the composition in which this Web service participates now. In case of composition failure, the Web service requests compensation/cancelation charges on top of its regular charges. If the Web service turns out the source of the failure, then it will be subject to financial penalties.

Privacy preferences are related to the data that Web services exchange in compositions. The following are examples of privacy arguments:

- *Data-source* argument: a Web service sets a list of peers from which it accepts data without checking their "credentials" [7, 11].
- *Data-destination* argument: a Web service sets a list of peers for which it forwards data without checking their "credentials" [7, 11].
- *Data-retention-period-at-destination* argument: a Web service sets a time frame for the destination peers to retain its data whether these data are updated or not. Afterwards, these data should be either deleted or forwarded. In the case of data forward, the privacy preferences of both sender and destination peers need to be satisfied. To counter-balance *data-retention-period-at-destination* argument that a sender Web service announces, each recipient Web service announces its *data-retention-period-at-reception* argument as well.
- *Data-disclosure-distance* argument: a Web service sets the maximum distance (e.g., number of edges that correspond to dependencies) for its data to be disclosed from one peer to another without seeking its direct approval. For example, in Figure 1 (we adopt state chart in our work [9]; states and transitions correspond to component Web services and dependencies between these component Web services, respectively) *data-disclosure-distance* for s_1 is set to 2, which means data of s_1 are disclosed to its direct connected peers (i.e., s_2) and the next direct connected peers (i.e., s_3 and s_4). To counter-balance *data-disclosure-distance* argument, each recipient Web service announces its *data-destination* argument so that the sender Web service approves the peers included in this argument.

It should be noted that *data-source* and *data-destination* arguments are critical in peer-to-peer-based composition. This is not the case in centralized-based composition where Web services might not know with whom they interact. Interactions in this composition are routed through a central component.

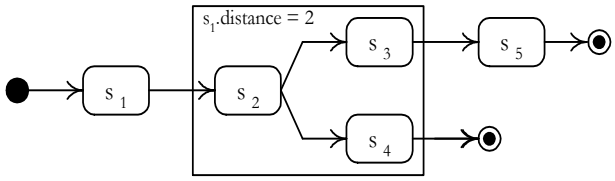

Figure 1. Illustration of *data-disclosure-distance* argument

B. Running example: cookout party

Our running example identifies a university student who organizes a cookout party for her recent graduation. The list of Web services implementing this party includes:

1. *CateringWS*: looks for and contacts catering companies according to criteria such as budget allocated, number of guests expected, and type of cuisine.
2. *GuestWS*: sends invitees invitations, keeps track of the confirmed ones, and follows-up on the unconfirmed ones through reminders.
3. *PlaceBookingWS*: looks for a place to host the cookout party, books the place, and completes the necessary paperwork like payment.
4. *WeatherWS*: checks the weather forecast for the day of the cookout party. In case of bad weather, the party takes place at the student's place.

Figure 2 represents the specification of the business logic that underpins the cookout-party composition. Some dependencies include: the party does not take place without checking the weather forecast on a specific date, and the quantity of food to prepare depends on the number of guests confirmed. For illustration purposes, we instantiate the preference arguments of *CateringWS* and *PlaceBookingWS*.

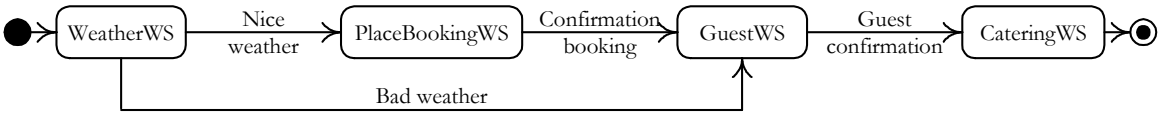

Figure 2. Specification of the cookout-party composition

CateringWS's partnership preferences are as follows:

- *Participation-duration* argument: 48 hours -- if the execution of the cookout-party composition lasts more than 48 hours, *CateringWS* will disengage from the composition. A remedy to this "expected" disengagement needs to be planned by the composition engineer by for example negotiating a longer engagement period with *CateringWS*.
- *Invocation-period* argument: null.
- *Payment-mode* argument: deferred -- *CateringWS* expects payment after the composition completes successfully. In case of failure that leads into cancelation, *CateringWS* charges additional fees because of the penalty included in the agreement with the catering company.

PlaceBookingWS's privacy preferences are as follows:

- *Data-source* argument: null.
- *Data-destination* argument: *GuestWS*.
- *Data-retention-period-at-destination* argument: up to 1 month from date of receipt.
- *Data-disclosure-distance* argument: 2 -- Data of *PlaceBookingWS* are transferred through *GuestWS* up to *CateringWS* without the approval of *PlaceBookingWS*.

C. S3P Establishment

In Section 2, we mentioned how Hamadi et al. inject privacy details into the specification (which is based on state chart) of a Web service [8]. Unfortunately, this injection does not comply with the *separation-of-concerns* principle since the revised specification of this Web service is strongly coupled to privacy details. As a result, changes in these details affect this specification and *vice-versa*. To address this limitation, our approach for handling Web services' preferences takes two inputs, namely the specification of a composition and the preferences of each component Web service in this composition, and produces one output, which is the S3P of this composition. An S3P is independent from the specification of a composition (i.e., *loosely coupled*). In an S3P, *tags* anchored to component Web services correspond to partnership preferences and the *privacy flow* corresponds to the application of privacy preferences on the data flow between the component Web services. In the following, we establish the S3P for the cookout-party using *CateringWS* and *PlaceBookingWS*.

Partnership preferences. They are represented with tags in the S3P. Each tag is structured as follows (Table 1): (i) argument name, (ii) preference type, (iii) corrective actions to take (*shown in italic*) if the preference is unsatisfied at run time, and (iv) the authority that executes the corrective actions. For example in Table 1, Tag #2 *invocation-period* argument, *CateringWS* receives an invocation request from the composite Web service. However, this request does not fall within the invocation period that was agreed-upon between both. As per the corrective actions for this argument, *CateringWS* either *rejects* the request or *applies* extra fees if it accepts to process this request. The extra fees are on top of the regular fees that *CateringWS* charges and reports using *payment-mode* argument (Tag #3).

TABLE I.
STRUCTURE OF TAGS ANCHORED TO WEB SERVICES

#	Argument name	Preference type	Corrective actions	Authority
1	*participation duration*	partnership	**If** participation-duration exceeded **Then** *replace* component WS	Composite WS
2	*invocation period*	partnership	**If** request falls outside the agreed upon period **Then** *reject* invocation **xor** *apply* extra fees on the composite WS	Component WS
3	*payment mode*	partnership	**If** late payment **Then** *apply* penalties on the composite WS	Component WS
4	*data retention period (at destination)*	privacy	**If** retention-duration exceeded **Then** *apply* penalties on the destination WS	Component WS (source)

Privacy preferences. Because of the use of state charts (in case of Petri-Nets, places and transitions will be adopted instead of states and transitions) to specify compositions (Figure 2), the privacy flow of the S3P is obtained by (i) adding *new direct links* (i.e., transitions) between the component Web services (i.e., states), or (ii) adding *generic* Web services between the component Web services. A generic Web service is limited to conveying data from one Web service to another without acting on these data. Except *data-retention-period-at-destination* argument that is handled using a tag (Table 1, Tag #4), handling the other privacy arguments calls for developing a dedicated flow (Figure 3):

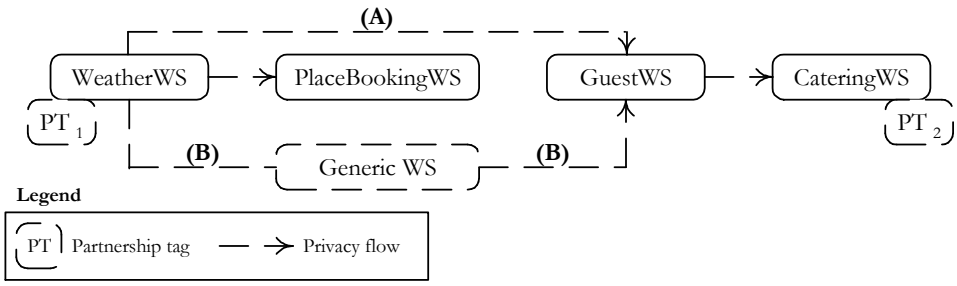

Figure 3. Handling Web services' preferences

- *Case of adding a new link* (Figure 3-(A)): *WeatherWS* sets *data-disclosure-distance* argument to 1, i.e., data to disclose up to *PlaceBookingWS* and *GuestWS* (in case of bad weather). However, *GuestWS* requires data input from *WeatherWS* in case of fine weather so that it informs the invitees of the location of the party (in case of fine weather, there is no direct link between *WeatherWS* and *GuestWS*). This location is a data input for *GuestWS*. To satisfy this preference, a direct link (a transition) that forms the privacy flow is added to the S3P from *WeatherWS* to *GuestWS* (Figure 3-(A)). Adding this link requires that *GuestWS* satisfies *data-destination* argument of *WeatherWS*
- *Case of adding a generic Web service* (Figure 3-(B)): *GuestWS* does not satisfy *data-destination* argument of *WeatherWS*, so the exchange of data through the existing link between these two Web services violates this preference. To deal with this violation, two options exist: (i) submit data via *PlaceBookingWS*, which is the current case in Figure 2, or (ii) introduce a generic Web service from *WeatherWS* to *GuestWS*. In either case, it is required that *data-disclosure-distance* argument is greater to one. Otherwise, this privacy preference cannot be satisfied.

In the following, we present two algorithms to handle privacy preferences with focus on *data-disclosure-distance* and *data-retention-period-at-destination* arguments. We map a composition specification (e.g., Figure 2) onto a graph *G=(N,E)*. In this graph the nodes *N* and edges *E* correspond to Web services and dependencies between these Web services, respectively. Each edge is a couple of the form $<s_i,s_j>$ where the edge is directed from s_i to s_j. Furthermore, the graph has two unique nodes: *START* and *END*. *START* node has no predecessors whereas *END* node has no successors. The graph is supposed to meet two basic conditions: (i) every node in the graph is directly or indirectly reachable from *START* node, and (ii) *END* node is reachable from every node in the graph.

In the algorithm for handling *data-disclosure-distance* argument (Figure 4), the following functions are used: *Indirect-Neighbor*(s_i), *Input-Data*(s_i), *Output-Data*(s_i), *Distance*(*Path*(s_i,*1[s]n*,s_j)), *Connect*(s_i,s_j), and *Connect*(s_i,s,s_i). This algorithm checks the data dependencies between Web services and establishes, if necessary, new connections either direct or indirect, between these Web services so that *data-disclosure-distance* argument is satisfied at run-time.

1. *Indirect-Neighbor*(s_i): returns the set of Web services that are indirectly connected to s_i through other Web services *(0,n)* with *0* and *n* standing for minimum and maximum,

respectively. This set permits forming paths *(Path(s_i,0[s]n,s_j))*, needs to be pruned from duplicate paths, and could be empty. The set of all the paths is stored for later use. If s_i and s_j are directly connected, *Indirect-Neighbor(s_i)* is equal to *Ø*, i.e., zero services between them.

2. *Input-Data(s_i)*: returns the set of data that s_i requires for functioning.
3. *Output-Data*(s_i): returns the set of data that s_i returns after functioning.
4. *Distance(Path(s_i,1[s]n,s_j)):* returns a set of numerical values that represent the numbers of Web services that separate s_i from s_j (distance at least greater or equal to one). These numbers illustrate the shortest and longest paths between s_i and s_j.
5. *Connect(s_i,s_j):* permits to form a new direct transition between s_i and s_j. This transition is added to meet some privacy requirements.
6. *Connect(s_i,s,s_j)*: permits to form a new indirect transition between s_i and s_j through a generic Web service s. This indirect transition is added to meet some privacy requirements.

In the algorithm for handling *data-retention-period-at-destination* argument (Figure 5), the following functions are used on top of *Input-Data(s_i)* and *Output-Data(s_i)* that were introduced earlier: *Direct-Neighbor(s_i)*, *Check-Duration(s_i,s_j)*, *Pass(s_i,s_j)*, and *Relax-Duration(s_i)*. This algorithm checks the data dependencies between Web services and either authorizes the flow of data between these Web services or invites some Web services to review their retention periods of the data they receive.

1. *Direct-Neighbor(s_i)*: returns the set of Web services that are directly connected to s_i.
2. *Check-Duration(s_i,s_j)*: verifies that *data-retention-period-at-destination* argument of s_i is in agreement with *data-retention-period-at-reception* argument of s_j.
3. *Pass(s_i,s_j):* submits data from s_i to s_j.
4. *Relax-Duration(s_i)* : is an invitation to the provider of s_i to relax its *data-retention-period-at-reception* argument.

```
Proc Data-Disclosure-Distance(si)
Input: INeighsi: set of all indirect neighbors to si
Input: Pathsi: set of all paths that come out of si
Auxiliary: i, j: integer
Begin
  INeighsi ← Ø
  Pathsi ← Ø
  INeighsi ← Indirect-Neighbor(si)
  Pathsi ← Indirect-Neighbor(si)
  For each sj in INeighsi do
    If Output-Data(si) ∩ Input-Data(sj) <> Ø then
      //sj needs data from si
      If Data-Disclosure-Distance(si) < Distance((Path(si,1[...]n,sj))) then
        //sj is not supposed to receive data from si
        Connect(si,sj)
        //Figure 3-(A) case, establishes a new dependency between si and si
        //this assumes that si accepts to interact directly with sj
        //as per data-destination privacy preference
      Else
        If (Path(si,1[...]n,sj) exists)
          and (Distance((Path(si,1[...]n,sj))) <= Data-Disclosure-Distance(si)) then
          //Find a path that already connects si and sj and
          //verify if this path does not violate data-disclosure-distance preference
          Use(Path(si,1[...]n,sj))
        Else
          Connect(si,Generic-WS,sj)
          //No path exists so establish a new dependency between si and sj through a generic WS
          //Figure 3-(B) case
        End if
      End if
    End if
  End for
End
```

Figure 4. Algorithm for handling *data-disclosure-distance* argument

```
Proc Data-Retention-Period-at-Destination(s_i)
Input: DNeigh_si : set of all direct neighbors to s_i
Auxiliary: i, j: integer
Begin
  DNeigh_si ← Ø
  DNeigh_si ← Direct-Neighbor(_si)
  For each s_j in DNeigh_si do
    If Output-Data(s_i) ∩ Input-Data(s_j) <> Ø then
      //s_j needs data from s_i
      If Check-Duration(s_i,s_j) then
        //s_j and s_i data durations are in agreement
        Pass(s_i,s_j)
      Else
        Relax-Duration(s_j)
        //s_j and s_i data durations are not in agreement
        //s_j is invited to relax its data retention duration
      End if
    End if
  End for
End
```

Figure 5. Algorithm for handling *data-retention-period-at-destination* argument

D. Formalization

This section formalizes the concepts and definitions given in the previous sections.

1. Based on Figure 2 that shows a state chart-based specification of a composite Web service, we define this specification as a 5-tuple $CWS = <WS,L,T,ws^0,F>$ where:
   - WS is a finite set of states that correspond to Web services' names;
   - ws^0 is the initial Web service in WS;
   - $F \subseteq WS$ is the set of final Web services;
   - L is a set of labels;
   - $T \subseteq WS * L * WS$ is the transition relation. Each transition $t=(ws^{src},l,ws^{tgt})$ consists of a source Web service $ws^{src} \in WS$, a target Web service $ws^{tgt} \in WS$, and a transition label $l \in L$.

   Example 1: Figure 2 is a state chart of the specification of the cookout-party composite Web-service. Several states like *WeatherWS* (initial state) and *CateringWS* (final state) and several transitions like *(WeatherWS, NiceWeather, PlaceBookingWS)* are represented. In this transition example, *WeatherWS* and *PlaceBookingWS* are the source and target states, respectively, and *NiceWeather* is the transition's label.
2. A preference model, *PM*, is denoted as $PM = <PAP,PRP>$ where:
   - *PAP* is the set of partnership preferences. Given a composite Web service specification *CWS*, a partnership preference *pap* of a component Web service *WS* in *CWS* is a tuple $pap_{WS} = (name,value,description,c.action,authority,Ont)$ where:
     i. *name* is the name of the partnership preference.
     ii. *value* is a value (numerical, string, etc.) assigned to the partnership name.
     iii. *description* is a narrative description of the partnership preference.
     iv. *c.action* is a list of corrective actions to take when the partnership preference is unsatisfied.
     v. *authority* is the body in charge of executing the list of corrective actions when the partnership preference is unsatisfied.
     vi. *Ont* refers to the ontology defining the partnership preference.
   - *PRP* is the set of *P*rivacy *P*references. Its definition is similar to *PAP*.
3. A privacy flow, denoted as *PF*, of a composite Web service *CWS* is a 5-tuple $PF_{CWS} = <WS_{PF},L_{PF},T_{PF},ws_{PF}^0,F_{PF}>$ where:
   - WS_{PF} is a finite set of states that correspond to Web services' names; three exclusive cases could exist ($|P$ represents the cardinality of the set P):
     i. $|WS_{PF}| = |WS|$; the number of Web services in the privacy flow is equal to the number of Web services in the specification of the composite Web service.
     ii. $|WS_{PF}| < |WS|$; the number of Web services in the privacy flow is less than the number of Web services in the specification of the composite Web service. The privacy flow requires less Web services (Figure 3-(A)).

iii. $|WS_{PF}| > |WS|$; the number of Web services in the privacy flow is greater than the number of Web services in the specification of the composite Web service. The privacy flow requires more Web services (Figure 3-(B)).

- ws_{PF}^{0} is the initial Web service in WS_{PF};
- $F_{PF} \subseteq WS$ is the set of final Web services;
- L_{PF} is a set of labels; like the three cases that feature the relationship between WS_{PF} and WS, similar cases apply to L_{PF} and L.
- $T_{PF} \subseteq WS_{PF} * L_{PF} * WS_{PF}$ is the transition relation. Each transition $t_{PF} = (ws_{PF}^{src}, l_{PF}, ws_{PF}^{tgt})$ consists of a source Web service $ws_{PF}^{src} \in WS_{PF}$, a target Web service $ws_{PF}^{tg} \in WS_{PF}$, and a transition label $l_{PF} \in L_{PF}$.

Example 2: Figure 3 is a state chart of the specification of the privacy flow of the cookout-party composite Web-service. Several states like *WeatherWS* (initial state) and *CateringWS* (final state) and several transitions like *(WeatherWS,*$\underline{B_1}$*,GenericWS)* are included. In this transition example, *WeatherWS* and *IntermediaryWS* are the source and target states, respectively, and $\underline{B_1}$ is the transition's label.

IV. APPROACH VALIDATION

To validate the integration of preferences into Web services, we describe in this section the architecture of the system through a proof of concept which we implemented. The implementation is designed as a Web application based on JEE framework. JSP (Java Server Pages) is used to create interfaces for providers to design and compose Web services. Java Servlets are used for managing the flow of service composition.

A. System Architecture

The modules that constitute the architecture of the system are shown in Figure 6. These modules are: *ServiceDesignInterface*, *BusinessLogicModeler*, *InteractionPreferencesModeler*, and *ServiceManager*. The first module provides a Graphical User Interface for service engineers (or providers) to design Web services. The second module assists service engineers specify and edit the business logic of compositions. The third module takes the specification of a Web service and injects it with preferences. The last module manages the registration and repository of composite Web services.

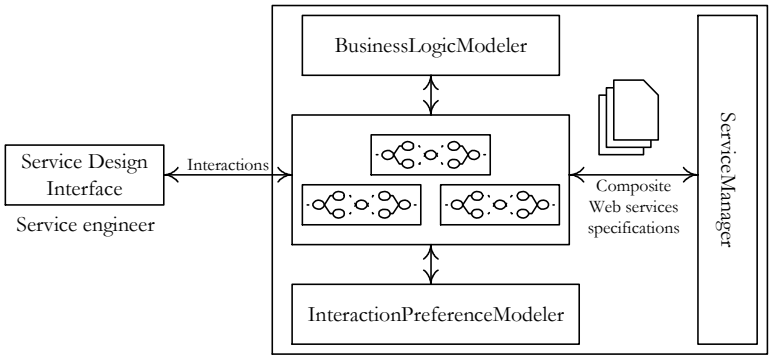

Figure 6. System Architecture

B. Implementation Prototype

The prototype is implemented with a two-fold objective which is to prove the architectural feasibility of injecting preferences into Web services and to validate the satisfaction of these preferences at run-time. The implementation is designed as a Web application. JSP (Java Server Pages) is used to create interfaces for providers to design and compose Web services. Operations of various modules are implemented with Java Servlets for managing the composition, flow of services and injecting preferences. For illustrative purposes we explain the *InteractionPreferencesModeler* module here.

The following assumptions are made: i) only one instance of each Web service is considered and ii) the flow of preferences for this implementation is as shown in Figure 3, without the branching to the *GenericWS.*

A set of preferences for participating Web services are defined where each individual preference has a *name*, *description* and *properties* as XML tags. The properties define attributes of a particular preference.

The *InteractionPreferencesModeler* module executes the functionalities of tagging the component Web services with partnership preferences and adding the privacy flow to the initial specification. This shows the consequences of applying privacy preferences on the data exchange between the component Web services. Once the preferences are set, these are injected by the *InteractionPreferencesModeler* into the respective component Web service.

The component Web service injected with the preferences will be positioned as a part of the Web composition based on the Business Logic given by the providers. An example of the preferences that could be injected is shown below.

```
<CateringWS>
…
 <Preferences>
  <preference prefId="124">
   <name>Partnership</name>
   <description> … </description>
   <properties>
     <Participation-duration> …
          </Participation-duration >
     <Invocation-period> …
          </Invocation-period>
     <Payment-mode> … </Payment-mode>
   </properties>
  </preference>
  <preference prefId="125">
   <name>Privacy</name>
   <description> … </description>
   <properties>
     <Data-source> … </Data-source>
     <Data-destination> …
          </Data-destination>
     <Data-retention-period> …
          </Data-retention-period>
     <Data-disclosure-distance> …
          </Data-disclosure-distance>
   </properties>
  </preference>
 </Preferences>
…
</CateringWS>
```

The flow diagram shown in Figure 7 for service composition describes an operation of *InteractionPreferencesModeler* module.

C. Discussion

With the design and implementation of the proposed system architecture, the various possibilities using S3P for Web service composition were explored. It was realized that the use of a standard protocol for specifying and injecting preferences, universally accepted, would enable the widespread use and control of Web service composition. It is evident that, the number of participating Web services and the respective preference parameters affect the turnaround time for the successful composition of Web services. The use of a business modeling language such as BPEL (Business Process Execution Language) would enhance the standardization of the architecture for integration of business processes with Web services. This also improves the possibilities of modeling preferences of participant behavior in business interactions. With the dynamic changes in preferences and the changes in policy we achieved varying the composition partnership and privacy information flow at runtime. Integrating the composition of Web services with the preferences of the providers using S3P was successfully demonstrated using this framework.

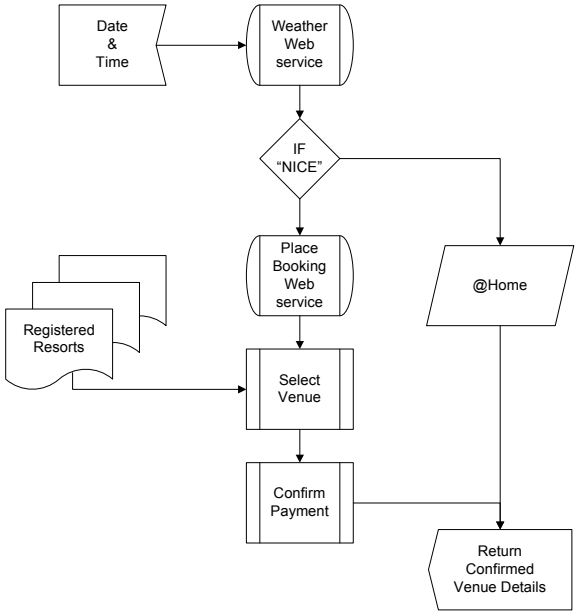

Figure 7. Service composition flow (1)

V. CONCLUSION

In a dynamic environment like the Internet software components including Web services need to be given the opportunity of specifying their preferences: with whom they like to interact, what data they like to release, what requests they like to process, etc. Through the S3P we assisted Web services in defining and verifying their preferences at run-time. We suggested two types of preferences, partnership geared towards satisfying composition requirements, and privacy geared towards satisfying data exchange requirements. In terms of contributions, we identified arguments that illustrate Web services' preferences, developed corrective actions to take when these preferences are not satisfied, and last but not least provided graphical means to model the integration of these preferences into Web services design. These means correspond to tags that label Web services and a privacy flow that shows how data flow between Web services. The privacy flow complies fully with the separation of concerns principle. It is loosely coupled to the business logic of compositions, and hence can be amended with no impact on these compositions.

In term of future work, we plan to continue enhancing the corrective actions per type of restriction and further improve the prototype. Another direction is about the second algorithm concerns "data-retention-period-at-destination" privacy preference that aims at restricting the use of the sender's data beyond a certain time period. Checking the implementation of such restrictions assumes that the recipient is trustworthy and takes the needed actions in responses to the restrictions that are put on the data it receives. For instance, it could send notification when data are deleted or forwarded. In the opposite case, the recipient could retain data for longer periods of time, change data if it is of type task-driven, etc. In that case, the sender Web service could time-stamp its data with a validity period.

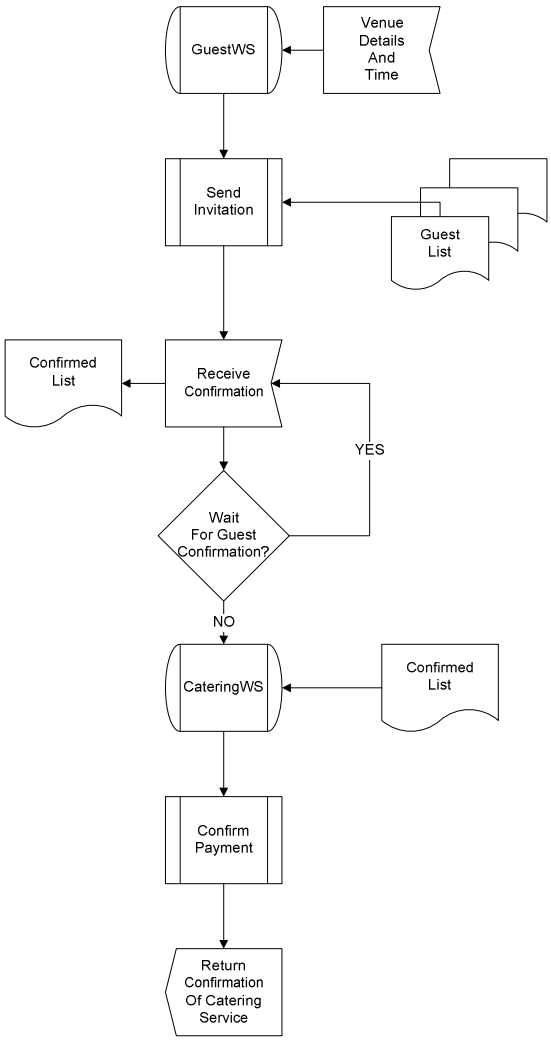

Figure 7. Service composition flow (2)

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their valuable comments and suggestions, which helped improve the paper.

REFERENCES

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. N. J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). Grid Resource Allocation Agreement Protocol (GRAAP) WG, March 2007.http://www.ogf.org/documents/GFD.107.pdf.

[2] S. Benbernou, H. Meziane, and M.-S. Hacid. Run-time monitoring for privacy-agreement compliance. In Proceedings of the Fifth International Conference on Service-Oriented Computing (ICSOC'2007), 2007.

[3] S. Benbernou, H. Meziane, Y. H. Li, and Hacid. M. S. A Privacy Agreement Model for Web Services. In Proceedings of the 2007 IEEE International Conference on Services Computing (SCC'2007), Salt Lake City, Utah, USA, 2007.

[4] J. Bentahar, Z. Maamar, D. Benslimane, and P. Thiran. An Argumentation Framework for Communities of Web Services. IEEE Intelligent Systems, 22(6), 2007.

[5] Carminati, B. and Ferrari, E. and Hung, P.C. K. Web service composition: A security perspective. In Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration (WIRI'2005) in conjunction with the 21st International Conference on Data Engineering (ICDE'2005), Tokyo, Japan, 2005.

[6] G. Chafle, S. Chandra, V. Mann, and M. Gowri Nanda. Orchestrating Composite Web Services under Data Flow Constraints. In Proceedings of The IEEE International Conference on Web Services (ICWS'2005), Orlando, Florida, US, 2005

[7] S. Elnaffar, Z. Maamar, H. Yahyaoui, J. Bentahar, and P. Thiran. Reputation of Communities of Web services - Preliminary Investigation. In Proceedings of the International Symposium on Web and Mobile Information Services (WAMIS'2008) held in conjunction with the 22nd International Conference on Advanced Information Networking and Applications (AINA'2008), Okinawa, Japan, 2008.

[8] R. Hamadi, H. Y. Paik, and B. Benatallah. Conceptual Modeling of Privacy-Aware Web Service Protocols. In Proceedings of the 19th International Conference on Advanced Information Systems (CAiSE'2007), Trondheim, Norway, 2007.

[9] D. Harel and A. Naamad. The STATEMATE Semantics of Statecharts. ACM Transactions on Software Engineering and Methodology, 5(4), October 1996.

[10] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Limiting Disclosure in Hippocratic Databases. In Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB'2004), Toronto, Canada, 2004

[11] Z. Li, S. Su, and F. Yang. WSrep: A Novel Reputation Model for Web Services Selection. In Proceedings of the First KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications (KES-AMSTA'2007), Wroclaw, Poland, 2007.

[12] M. Little. Transactions and Web Services. Communications of the ACM, 46(10), October 2003.

[13] Z. Maamar, D. Benslimane, G. Kouadri Mostefaoui, S. Subramanian, and Q. H. Mahmoud. Towards Behavioral Web Services Using Policies. IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans, 38(6), 2008.

[14] Z. Maamar, D. Benslimane, and Q. Z. Sheng. Towards A Two-Layered Framework for Managing Web Services Interaction. In Proceedings of the 6th Annual IEEE/ACIS International Conference on Computer and Information Science (ICIS'2007), Melbourne, Australia, 2007.

[15] T. Margaria. Service is in the Eyes of the Beholder. IEEE Computer, 40(11):33–37, November 2007.

[16] B. Medjahed and Y. Atif. Context-based Matching for Web Service Composition. Distributed and Parallel Databases, Springer, 21(1), January 2007.

[17] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, F. Rosenberg, and S. Dustdar. A Context-based Mediation Approach to Compose Semantic Web Services. ACM Transactions on Internet Technology, Special Issue on Semantic Web Services: Issues, Solutions and Applications, 8(1), 2007.

[18] J. Myoung Ko, C. Ouk Kim, and I.-H. Kwon. Quality-of-Service oriented Web Service Composition Algorithm and Planning Architecture. Journal of Systems and Software, 81(11), November 2008.

[19] M. Papazoglou. Web Services and Business Transactions. World Wide Web, 6(1), 2003.

[20] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer, 40(11):38–45, November 2007.
[21] A. Rezgui, M. Ouzzani, A. Bouguettaya, and B. Medjahed. Preserving Privacy in Web Services. In Proceedings of the Fourth ACM International Workshop on Web Information and Data Management (WIDM'2002) held in conjunction with the Eleventh International Conference on Information and Knowledge Management (CIKM'2002), MclEan, Virginia, USA, 002.
[22] Y. Sun, S. He, and J. Y. Leu. Syndicating Web Services: A QoS and User-driven Approach. Decision Support Systems, 43(1), 2007.
[23] A. Tumer, A. Dogac, and I. H. Toroslu. A Semantic-Based User Privacy Protection Framework for Web Services. In Proceedings of the Workshop on Intelligent Techniques for Web Personalization (ITWP'2003) held in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI'2003), Acapulco, Mexico, 2003.
[24] W. Xu, V. N. Venkatakrishnan, R. Sekar, and I. V. Ramakrishnan. A Framework for Building Privacy-Conscious Composite Web Services. In Proceedings of the 2006 IEEE International Conference on Web Services (ICWS'2006), Chicago, Illinois, USA, 2006.
[25] Q. Yu, A. Bouguettaya, and B. Medjahed. Deploying and Managing Web Services: Issues, Solutions, and Directions. The VLDB Journal, 17(3):537–572, 2008.
[26] M. Zuidweg, J. G. Pereira Filho, and M. van Sinderen. Using P3P in a Web Services-based Context-Aware Application Platform. In Proceedings of the 9th Open European Summer School and IFIP Workshop on Next Generation Networks (EUNICE'2003), Balatonfured, Hungary, 2003.
[27] L. Liu, H. Zhu, Z. Huang, and D. Xie. Minimal Privacy Authorization in Web Services Collaboration, Computer Standards & Interfaces, 33(3), 2011.
[28] H. Meziane and S. Benbernou. A Dynamic Privacy Model for Web Services, Computer Standards & Interfaces, 32(5-6), 2010.

Zakaria Maamar is a full professor in the College of Information Technology at Zayed University in Dubai, U.A.E. His research interests are primarily related to service sciences theories and methods, context-aware computing, and enterprise systems interoperability. Dr. Maamar has published several peer-reviewed papers in journals and conferences and regularly serves on the program and organizing committees of several international conferences and workshops. Dr. Maamar graduated for his M.Sc. and Ph.D. in Computer Sciences from Laval University in Canada in 1995 and 1998, respectively.

Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales, Sydney, Australia. He is a senior lecturer in the School of Computer Science at the University of Adelaide. His research interests include service-oriented architectures, distributed computing, and pervasive computing. He is the recipient of Microsoft Research Fellowship in 2003. He is the author of more than 80 publications. He is a member of the IEEE and the ACM.

Yacine Atif received the PhD degree in Computer Science from Hong Kong University of Science and Technology (HKUST) in 1996. After graduation, he worked at Purdue University in the USA as a Post- Doc and then joined a faculty position at Nanyang Technological University (NTU) in Singapore. Since 1999 he is with the UAE University as faculty, then Program Chair at the College of Information Technology. Dr. Atif has made a number of research contributions particularly in the areas of Semantic Web and related Learning Technology applications. He is also involved in the Technical Programs of several research forums.

Sujith Samuel Mathew is a PhD student at the University of Adelaide with research interests in Ubiquitous computing, the Future Internet and Web Services. He has received his Master's degree in Software Engineering from the Visvesvaraya Technological University (VTU), India. He has over ten years of experience working both in the IT Industry and in IT Academia. He has held positions as Software Engineer, Technical Evangelist and Group Leader within the IT industry. He moved into academia when he joined the Faculty of IT, UAE University in 2006. Since then he has been teaching various IT related topics and pursuing his research interests in parallel.

Khouloud Boukadi is an associate professor in Computer Science in the Multimedia, InfoRmation systems & Advanced Computing Laboratory -Miracl (Faculty of Economics and Management of Sfax - Tunisia). Her research interests include service computing, context-aware computing, and agility of information systems. She has a Ph.D. in Computer Sciences from Ecole des Mines, Saint Etienne, France.