Context for Personalized Web Services

Zakaria Maamar
Zayed University
United Arab Emirates
zakaria.maamar@zu.ac.ae

Soraya Kouadri Mostéfaoui
Fribourg University
Switzerland
kouadris@acm.org

Qusay H. Mahmoud
Guelph University
Canada
qmahmoud@uoguelph.ca

Abstract

The paper discusses the way context is used for Web services personalization. A Web service is an accessible application that other applications and humans can discover and trigger to satisfy various needs such as car rental. Context is the information that characterizes the interactions between humans, applications, and the surrounding environment. Web services are personalized so that users' preferences can be accommodated. Preferences are of different types varying from when the execution of a Web service should start to where the outcome of this execution should be delivered. Besides users' preferences, this paper highlights that the resources on which the Web services are performed have an impact on Web services personalization.

1. Introduction

A *Web service* [21] is an accessible application that other applications and humans can discover and trigger to satisfy various needs (e.g., currency conversion). One of the major strengthens of Web services (also called services in this paper) is their capacity to be composed into high-level business processes known as *composite services*. Composition addresses the situation of a user's request that cannot be satisfied by any available service, whereas a composite service obtained by combining a set of available services might be used [4]. For example, reviewing a paper for a scientific journal requires identifying the appropriate referees, assigning the paper to the referees, and finally collecting the referees' comments back for decision-making and notification.

Because users' expectations and requirements constantly change, it is important to include their *preferences* in Web services. Indeed, some users would like receiving answers to their personal requests directly submitted to their personal email instead of office's email. Some users would like having certain services, such as traffic conditions, automatically triggered once they get into their cars. These basic situations shed the light on *personalization* and its impact on making applications adaptable. Personalization can be of type explicit or implicit [17]. Explicit personalization calls for a direct participation of users in the adjustment of applications. Users clearly indicate the information that needs to be treated or discarded. Implicit personalization does not call for any user involvement and can be built upon learning strategies that track users' habits, interests, and behaviors. Personalization is dependent on the features of the environment in which this personalization is expected to happen. These features can be related to users (e.g., stationary user, mobile user), time periods (e.g., in the afternoon, in the morning), and physical locations (e.g., meeting room, cafeteria). Being aware of the features of an environment raises the importance of having a structure for collecting, tracking, and storing these features. We denote this structure by *context*. Context is the information that characterizes the interaction between humans, applications, and the surrounding environment [7]. The field of Web services is a very active area of research and development [21]. However, *very little* has been accomplished to date regarding their personalization using context. Several obstacles still hinder personalization such as: (i) current Web services act as passive components rather than active components that can be embedded with context awareness mechanisms, (ii) existing approaches for service composition (e.g., WSFL and BPEL) typically facilitate choreography only, while neglecting contextual information on users and services too, and (iii) lack of support techniques for modeling and specifying the integration of personalization into Web services. In this paper, we present our work on personalizing Web services. The major features of this work are as follows:

- Three types of context exist and correspond to *U*-context of *U*ser context, *W*-context of *W*eb service context, and *R*-context of *R*esource context.
- Different types of *policies* are developed to manage the integration of personalization into Web services. The use of policies guarantees that the Web services still do what they are supposed to do despite personalization.

The remainder of this paper is organized as follows. Section 2 overviews some concepts e.g., Web services and context. Section 3 presents the approach for personalizing Web services. Section 4 discusses the value-added of policies to manage personalization. The implementation of Web services personalization is discussed in Section 5. Section 6 presents some related work. The development trends happening in the field of Web services are discussed in Section 7. Section 8 concludes the paper.

2. Background

A Web service is an accessible application that other applications and humans can discover and invoke. Benatallah et al. suggest the following properties of a Web service [3]: (i) independent as much as possible from specific platforms and computing paradigms; (ii) primarily developed for inter-organizational situations; and (iii) easily composable so that developing complex adapters for the needs of composition is not required.

Context is any information that is relevant to the interactions between a user and an environment [8]. This information is about the circumstances, objects, or conditions by which the user is surrounded. Many researchers have tried defining context such as Schilit et al. who decompose context into three categories [23]: computing context, user context, and physical context.

Personalization involves a process of gathering user information during interaction with the user, which is then used to deliver appropriate content and services, tailor made to the user's needs. The aim is to improve the user's experience of a service [5]. Personalization is to integrate users' preferences into the process of delivering any information-related content or outcome of service computing. It is shown for instance that the needs of mobile users regarding information access are quite different from the needs of stationary users [19]. Mobile users' needs are not about browsing the Web, but about receiving personalized content that is highly sensitive to their immediate environment and respective requirements.

3. Context-based personalization

3.1. Deployment

Figure 1 illustrates the approach backing Web services personalization. In this approach, the core concept is context from which three sub-contexts are obtained: *U*-context, *W*-context, and *R*-context.

Muldoon et al. define user context as an aggregation of his location, previous activities, and preferences [17]. Su adopts the same definition and even adds physiological information to user context [24]. We define the Web-service context of a Web service as an aggregation of its simultaneous participations in composite services, locations of execution, times of execution, and constraints during execution. In addition, we define the resource context of a resource as an aggregation of its current status, periods of non-availability, and capacities of meeting the execution requirements of Web services.

In Figure 1, *U*-context, *W*-context, and *R*-context are interconnected. From *R*-context to *W*-context "execution adjustment" relationship identifies the execution constraints on a Web service (e.g., execution time, execution location, flow dependency) *vs.* the execution capabilities of the resource (e.g., next period of availability, scheduling policy) on which the Web service will be performed. A resource has to check its status and assess its current commitments before it agrees on supporting the execution of a service. From *U*-context to *W*-context, provisioning personalization relationship identifies the preferences of users (e.g., when and where a Web service needs to be executed, and when and where the execution's outcome needs to be returned) *vs.* the capabilities of a Web service to accommodate these preferences (e.g., can a service be executed at a certain time or in a certain location). A Web service needs to check its status before it agrees on handling a user's needs. In Figure 1, *W*-context is the common element between *U*-context and *R*-context. A Web service has to reconcile between what a user wishes and what a resource permits.

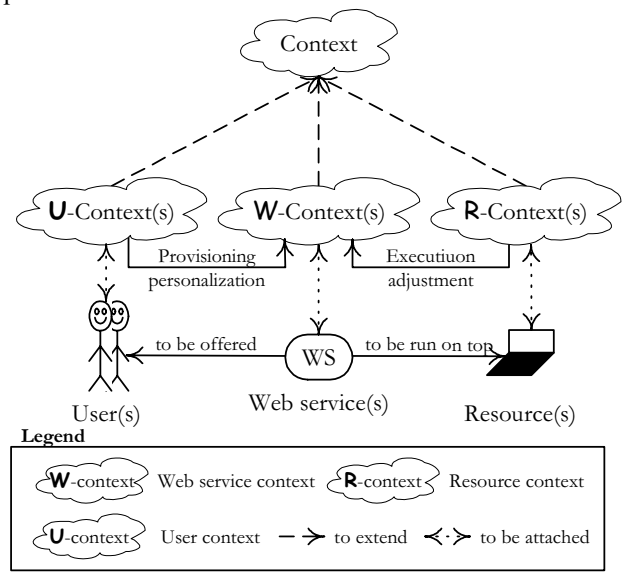

Figure 1 Personalization of Web services

3.2. Types and roles of context

Figure 1depicts three types of context. Each context is attached to a specific component whether user, Web service, or resource. User is the most dynamic component. His needs, requirements, and preferences always change. Resource is nearly the most stable component. The computing features of and constraints on a resource can to a certain extent be known in-advance. Therefore, the capabilities of a resource can be tuned in order to meet certain requirements such as communication access reliability and efficiency of security mechanisms. The Web service component is between user and resource. A Web service is responsible for finding a compromise between what users prefer and what resources permit.

***U*-context.** The role of *U*-context is to track the current status of a user and reflects his personal preferences in terms of execution location and execution time of services. The following parameters define *U*-context (Table 1): label, previous locations/services, current location/services, next locations/services, previous periods of time/services, current period of time/services, next periods of time/services, and date. Previous locations and previous periods of time parameters illustrate the Web services that were executed in the past. Next locations and next periods of time parameters illustrate the Web services that will be executed in the future.

Table 1 Description of *U*-context's parameters

Parameter & Description
Label: corresponds to the identifier of the user.
Previous locations/services: keeps track of all the locations, as indicated by the user, that have featured in the past the execution of services (null if there are no predecessor locations.
Current location/services: indicates the current location, as indicated by the user, that should feature now the execution of services.
Next locations/services: indicates all the locations, as indicated by the user, that will feature the execution of services (null if there are no next locations).
Previous periods of time/services: keeps track of all the periods of time, as indicated by the user, that have featured the execution of services (null if there are no predecessor periods of time).
Current period of time/services: indicates the current time, as indicated by the user, that should feature now the execution of services.
Next periods of time/services: keeps track of all the periods of time, as indicated by the user, that will feature the execution of services (null if there are no next periods of time).
Date: identifies the time of updating these parameters.

***W*-context.** The role of *W*-context is to be aware of the current status of a Web service and the execution constraints on the Web service. These constraints are tightly dependent on users' preferences of type execution time requested and execution location requested. A Web service is triggered each time it receives an invitation of participation in a composite service (details are given in [13] on what an invitation is). Before a service accepts an invitation it performs some verifications among them (i) number of current participations *vs.* number of allowed participations, (ii) expected completion time of current participations, and (iii) features of the newly-received invitation with regard to execution time and execution location. It happens that a Web service refuses an invitation of participation in a composite service because of multiple reasons: (i) period of unavailability for some maintenance work or (ii) resource unavailability.

The following parameters define *W*-context (Table 2): label, status per participation, previous services per participation, next services per participation, regular actions, start time per participation (requested and effective), location per participation (requested and effective), and date. Previous services per participation parameter illustrates the predecessor Web services to the current Web service that were executed in the past. Next services per participation parameter illustrates the successor Web services to the current Web service that are expected to be executed in the future. It should be noted that per participation in the aforementioned list of parameters stands for each composite service in which a Web service participates. Mechanisms that allow a Web service to participate in several composite services are detailed in [14].

Table 2 Description of *W*-context's parameters

Parameter & Description
Label: corresponds to the identifier of the Web service.
Status per participation: informs about the current status of the service with regard to each composite service in which the service takes part. Status can be of type in-progress, suspended, aborted, or terminated.
Previous services per participation: indicates whether there are services before the service with regard to each composite service (null if there are no predecessors).
Next services per participation: indicates whether there are services after the service with regard to each composite service (null if there are no successors).
Regular actions: illustrates the actions that the service normally performs.
Start time per participation (requested and effective): informs when the execution of the service should start, as requested by the user, and has effectively started with regard to each composite service.
Location per participation (requested and effective): informs where the execution of the service should happen as requested by the user (i.e., user-related) and has effectively happened (i.e., execution-related) with regard to each composite service.
Date: identifies the time of updating these parameters.

In *W*-context of Table 2, time-requested and location requested parameters are user-dependent. By user dependent, it is meant that the user has to assign values to both parameters. Time-effective and location-effective parameters are execution-dependent, (i.e., when and where the execution has really happened). Values to assign to time effective parameter can be obtained from the resource on which a service was executed, whereas values to assign to location effective parameter are obtained from users (we argue in Section 3.3 why a manual detection of the user's location is adopted).

To verify that time and location preferences of a user have been properly considered during the deployment of a

component service, the value of time-requested or location requested parameters should respectively be equal to the value of time-effective and location-effective parameters (a slight difference is also acceptable). Any discrepancy between a parameter of type requested and parameter of type effective is an indication that the user's adjustment in term of execution location or execution time of a Web service has not been correctly handled. The user needs to be informed about the discrepancy so that he could update (or through a third component acting on his behalf) the relevant parameters of *U*-context. These parameters are previous locations/services and previous periods of time/services. In addition, the Web service needs to find out the reasons of the discrepancy with regard to what was requested and what has effectively happened. With regard to location preference, we recall that users have to explicitly announce their location. A user who forgets announcing his location is a reason for delaying the execution of a service.

***R*-context.** The role of *R*-context is to track the current status of a resource. Before a resource accepts the support of a service execution, it performs some verifications including (i) number of Web services currently executed *vs.* maximum number of Web services under execution, (ii) approximate completion time of current executions, and (iii) execution time of the newly-received request. It happens that a resource rejects a request of executing a Web service because of multiple reasons: (i) period of unavailability due to some upgrade work or (ii) potential overloaded status.

The following parameters define *R*-context (Table 3): label, previous periods of time/services, current period of time/services, next periods of time/services, previous locations/services, current location/services, next locations/ services, and date. Previous periods of time parameter illustrates the periods of time with regard to a particular period of time that have featured the execution of Web services on a resource in the past. Next periods of time parameter illustrates the periods of time with regard to a particular period of time that will feature the execution of Web services on a resource in the future.

Table 3 Description of *R*-context's parameters

Parameter & Description
Label: corresponds to the identifier of the resource.
Previous periods of time/services: keeps track of the periods of time, as indicated by the user, that have featured the execution of services with regard to each composite service (null if there are no predecessor periods of time). The effective periods of time of the execution of services are also reported in this parameter.
Current period of time/services: indicates the current time, as indicated by the user, that should feature now the execution of services with regard to each composite service.
Next periods of time/services: keeps track of all the periods of time, as indicated by the user, that will feature the execution of services with regard to each composite service (null if there are no next periods of time).
Previous locations/services: keeps track of the locations, as indicated by the user, that have featured the execution of services with regard to each composite service (null if there are no predecessor periods of time). The effective locations of the execution of services are also reported in this parameter.
Current location/services: indicates the current location, as indicated by the user, that should feature now the execution of services with regard to each composite service.
Next locations/services: keeps track of all the locations, as indicated by the user, that will feature the execution of services with regard to each composite service (null if there are no next periods of time).
Date: identifies the time of updating these parameters.

3.3 Operation

Figure 2 illustrates the interactions that occur during context-based personalization of Web services. When a user selects a Web service, he proceeds next with its personalization according to his time and location preferences. Time preference is organized along two parts: (i) when the execution of the service should start, and (ii) when the outcome of this execution should be returned to the user. A user could request the start of the execution of a Web service at 2pm and the delivery of the result after 4pm as he will be in a meeting from 2 to 4pm. It happens that execution time and delivery time are equal if the time that the execution lasts is excluded or negligible. Location preference is organized along two parts: (i) where the execution of the service should occur, and (ii) where the outcome of this execution should be returned to the user. A user could ask the start of the execution of "mall map" service once he enters the company's meeting room and the delivery of the execution result to his mobile phone when he is in the vicinity of the mall. It also happens that execution location and delivery location are both the same.

Once the user's preferences are submitted to the Web service, this one ensures that the dates and locations are valid and no conflicts might happen during deployment. For instance, the delivery time cannot occur before the execution time of a service. Moreover, the user has to be continuously reminded that he has to explicitly identify his current location so that execution location and delivery location are both properly handled[1]. Priori to

1 While a manual feeding of the current location of users present some limitations, this type of feeding allows a much better handling of the privacy issue. Indeed, users only

identifying the resources on which it will be executed, the Web service checks its *W*-context with regard to (i) number of Web services currently under execution *vs.* maximum number of Web services under execution and (ii) next period of unavailability.

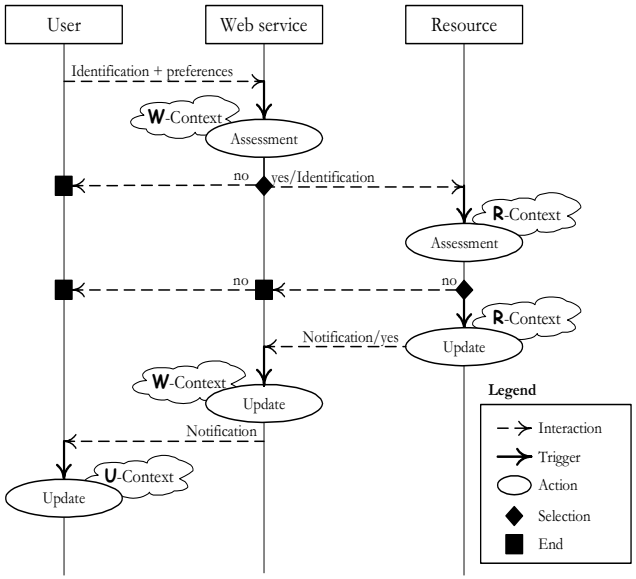

Figure 2 Interactions during Web services personalization

After a positive verification of the *W*-context, the identification of a resource is now launched. It is assumed that a mechanism supporting resource identification is available. A resource needs mainly to accommodate two things: (i) the start time of a service execution, and (ii) the time that the execution of a service lasts since the outcome of this execution needs to consider the delivery time as per user indication. To this purpose, a resource checks its *R*-context with regard to (i) the number of Web services currently under execution *vs.* maximum number of Web services under execution and (ii) next period of unavailability. After a positive check of the *W*-context, the identification of a resource can now be launched. It is assumed that a mechanism supporting resource identification is available. A resource needs mainly to accommodate two things: (i) the start time of a service execution, and (ii) the time that the execution of a service lasts since the outcome of this execution needs to consider the delivery time as per user indication. To this purpose, a resource checks its *R*-context with regard to (i) the next periods of time that will feature the execution of Web services and (i) next period of maintenance. After a positive check, the resource notifies the service, which itself notifies the user. User notification means the update of the following parameters: (i) next locations/services and next periods of time/services of *U*-context (Table 1), (ii), next services per participation, start-time requested per participation, and location requested per participation of *W*-context (Table 2), and (iii) next periods of time/services and next locations/services of *R*-context (Table 3). Updating these parameters and mainly the parameters that correspond to the next actions to take is a good indication of the assessment that occurs in terms of (i) which Web services are involved and in which composite services, (ii) which resources are considered, and (iii) which locations or periods of time will feature the execution of Web services. This type of assessment enables predicting the situations that will happen and preparing the corrective plans in case of exceptions.

reveal to external systems the locations they wish to be known. An automatic feeding of the location of users is doable and can be based on satellite-based techniques.

4. Policies for personalization management

In Section 1, the use of policies for managing the integration of personalization into Web services was suggested. In fact, policies provide many benefits such as reusability, extensibility, and context-sensitivity [26].

Because personalization is dependent on specific preferences, policies aim at specifying what, when, and where to track, and how to perform the tracking so that these preferences are properly handled. Besides the widely adoption of policies as authorization mechanisms, the value-added of policies has been reported in different works. In [15] Maamar et al. used policies for the dynamic management of multiple UDDI registries in a wireless environment of Web services. Policies are also used in conversations as reported in [11,13]. Because of users' preferences and resources' availabilities, a Web service is adjusted so that it accommodates these preferences and availabilities. To ensure that the adjustment of a Web service is efficient, we developed three types of policies (owners of Web services are responsible for developing the policies). The first type, called *consistency*, checks the status of a Web service after being personalized. The second type, called *feasibility*, ensures that a personalized Web service finds a resource on which it can be executed according to the constraints of time and location. Finally, the third type, called *inspection*, ensures that the deployment of a personalized Web service complies with the adjusted specification.

The consistency policy guarantees that a Web service still does what it is supposed to do after personalization. Personalization may alter the initial specification of a service when it comes for instance to the list of regular events that trigger the service. Indeed, time- and location-related parameters are new events that need to be added to the list of regular events. Moreover, because of QoS-related parameters (e.g., availability, response time, and throughput) of a Web service [16,25], it is important to

verify that these QoS parameters did not change and are still satisfied despite the personalization.

The feasibility policy guarantees that an appropriate resource is always identified for the execution of a personalized Web service. Because services have different requirements (e.g., period of requests, period of deliveries) and resources have different constraints (e.g., period of availabilities, maximum capacity), an agreement has to be reached between what services need in terms of resources and what resources offer in terms of capabilities. Furthermore, the feasibility policy checks that the new operations of the personalized Web service are properly handled by the available resources.

The inspection policy is a means by which various aspects are considered such as what to track (time, location, etc.), who did ask to track (user, service itself, or both), when to track (continuously, intermittently), when and how to update the different arguments of the different contexts, and how to react if a discrepancy is noticed between what was requested and what has effectively happened. The inspection policy is mainly tightened to the parameters of type requested and effective of the *W*-context of a Web service (Table 2). If there is a discrepancy between the requested and effective parameters, the reasons have to be determined, assessed, and reported. One of the reasons could be the lack of appropriate resources on which the personalized service needs to be executed.

Based on consistency, feasibility, and inspection policies, we define what we refer to as *scope* of personalization and *scope* of adjustment. On one hand, the scope of personalization of a Web service represents the elements of the service that can be adjusted without affecting or altering the consistency of the service itself. This scope is identified with "provisioning personalization" relationship in Figure 1. On the other hand, the scope of adjustment of a Web service represents all the new elements in terms of operation and binding that are added to a personalized service and guaranteed to be performed by the available resources. This scope is identified with "execution adjustment" relationship in Figure 1. Enforcing the validity of both scopes is among the objectives that the inspection policy achieves.

5. Implementation

A proof of concept implementation is under construction using the Sun Microsystems' J2EE 1.4 to create Web services, and their reference implementation of JSR 188 (Composite Capability/Preference Profile Processing Specification)[2] for context information representation and processing. The Composite capability/Preferences Profile (CC/PP)[3] is an industry standard developed by the W3C that provides a way for client devices to transmit their capabilities and user preferences. CC/PP is based on the Resource Description Framework (RDF), which is an approach for representing statements each of which containing a subject, predicate, and an object. CC/PP uses the XML serialization of RDF. A concrete implementation and extension of the CC/PP is the UAProf[4], which was developed by the Open Mobile Alliance (OMA) for WAP-enabled devices.

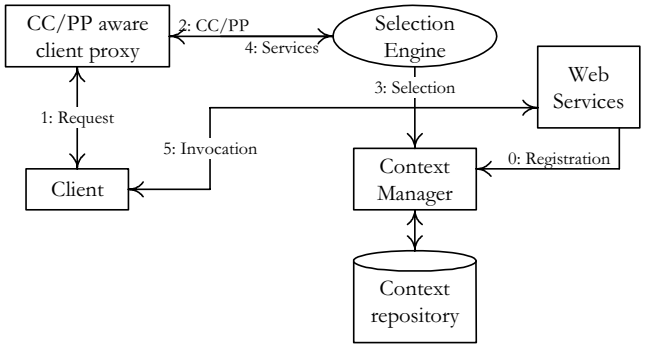

Figure 3 Prototype architecture

In order to ensure that any Web browser can be used with our prototype implementation, we have devised a multi-tier architecture. Instead of using a custom CC/PP aware Web browser, we have developed a CC/PP aware client proxy that receives the client request and inserts CC/PP headers that correspond to the client profile. Reference profiles and user's preferences are sent as part of the proxy request to the selection engine. Figure 3 illustrates this architecture and its components.

The client can be any Web browser that sends and receives HTTP requests. The CC/PP aware client proxy acts as an HTTP proxy server, which inserts CC/PP headers about the client profile and user's preferences. The request is then forwarded to a CC/PP aware server that acts as a Selection Engine. This one interacts with the Context Manager to select the most appropriate services and send their addresses to the client to interact with them. The Context Manager interacts with the Context Repository that keeps track of *U*-, *W*-, and *R*-contexts. In order to improve performance, client and user preferences profiles are cached only for the duration of the session.

6. Related Work

Web services are a very active area of research and development [21]. However, to our knowledge none of the research projects have aimed at personalizing services using context. To cope with this lack of related work, we

[2] www.jcp.org/en/jsr/detail?id=188.
[3] www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115.

4 www.wapforum.org/what/technical/SPEC-UAProf-19991110.pdf..

present the following some of the works that supported our context-based personalization of Web services.

Barkhuus and Dey have identified three levels of interactivity for context-aware applications [2]: personalization, active context-awareness, and passive context-awareness. For both authors, personalization, also referred to customization and tailoring, is motivated by the diversity and dynamics featuring nowadays applications. For active context-awareness, it concerns applications that, on the basis of sensor data, change their content autonomously. Whereas in passive context-awareness applications merely present the updated context to the user and let the user specify how the application should change. In this paper, we adopted a passive context-awareness style with the manual feeding of the user's current location. While we mentioned that this type of feeding presents some limitations *vs.* an automatic feeding, this enables however an efficient handling of the privacy concern of users.

In Table 1, parameters of type requested *vs.* parameters of type effective overlaps with the QoS of type advertised *vs.* QoS of type delivered. Ouzzani and Bouguettaya report that a key feature in distinguishing between competing Web services is their QoS, which encompasses several qualitative and quantitative parameters that measure how well the Web service delivers its functionalities [18]. A Web service may not always full advertised QoS parameters, due to various fluctuations related for example to the network status or resource availability. Therefore, some differences between QoS advertised and QoS delivered values occur. However, large differences indicate that the Web service is suffering performance degradation in delivering its functionalities. The same comment is made on parameters of type requested *vs.* parameters of type effective when it comes to service personalization. A major difference between values of respective parameters indicate that a user's personal preferences were not considered.

While in this paper context is used for Web services composition and provisioning, other projects such as [6] have done the opposite by using Web services for managing context provisioning. Breener and Schiffers envision that context information will typically be provided by autonomous organizations (or context providers), which means heterogeneity and distribution challenges to deal with. Additional challenges are cited in [6] including (i) what is the optimal sequence for gathering and combining the required context information, (ii) how to secure the whole context provisioning process, and (iii) how is the cooperation between the providers of context achieved, and even enforced?

7. Development trends in Web services

While much of the work to date has focused on standards for announcing, discovering, and invoking Web services, there is a significant development happening in Web services. In this section, we overview some of the developments related to conversation-driven composition and semantic Web services.

7.1. Conversation-driven composition

A conversation is an exchange of messages between participants involved in joint operations. A conversation succeeds when the outcome that is expected out of that conversation is achieved. Further, a conversation fails when the conversation faced difficulties or the outcome that is expected is not achieved.

Using conversations helps define composite services at run-time instead of design-time. When a Web service is being executed, it has at the same time to initiate "freedom" to Web services to decide if they will take part in this orchestration. Conversations are more than just combining components; they rather promote the autonomy conversations with the Web services that are due for execution. The purpose of these conversations is to invite the Web services to join the composition process, and ensure that the Web services are ready for execution in case they accept the invitation [12].

Ardissono et al. observed that current Web services communication standards support simple interactions and are mostly structured as question-answer pairs [1]. These limitations hinder the possibility of expressing complex situations that require more than two turns of interactions (e.g., propose/counter-propose/accept_reject). In addition, Ardissono et al. proposed a conversational model, which supports complex interactions between clients and Web services, where several messages are exchanged before a Web service is completed. While the orchestration of Web services is a core component to any Web services integration effort, the use of conversations gives more "freedom" to Web services so they can decide if they will take part in this orchestration. Conversations are more than just combining components; they rather promote the autonomy of components that act and react according to their environment [9].

7.2. Semantic Web services

Another major trend is the integration of semantics into Web services. Heflin and Huhns argue that the goal driving the semantic Web is to automate Web-document processing [10]. The semantic Web aims at improving the technology that organizes, searches, integrates, and evolves Web-accessible resources. This requires the use of rich and machine-understandable abstractions to represent the resource semantics.

One of the core components to the widespread of the semantic Web is the development of ontologies that

specify standard terms and machine-readable definitions. Although there is no consensus yet on what an ontology is, most of researchers in the field of knowledge representation consider a taxonomy of terms and mechanisms for expressing the terms and their relationships. Samples of markup language for publishing and sharing ontologies on the 3W include RDF (Resource Description Framework), DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer), and OWL (Web Ontology Language) [22].

By combining efforts of Web services and semantic Web communities, it is expected that new mechanisms for enabling automated discovery, access, combination, and management for the benefit of semantic Web services will be developed. Paolucci and Sycara note that the semantic Web provides tools for explicit markup of Web content, whereas Web services could create a network of programs that produce and consume information, enabling automated business interactions [20].

8. Conclusion

In this paper, we presented our approach of context based personalization for Web services composition. Three types of context, namely *U*-context, *W*-context, and *R*-context, are the cornerstone of the approach by storing details related to personalization such as preferred execution time and preferred execution-location of Web services. The effect of changes of a Web service because of users' preferences and resources' availabilities has required the development of three types of policies referred to as consistency, feasibility, and inspection. Our use of policies has guaranteed that the Web services still do what they are supposed to do despite personalization. Some of the elements that could be identified through the use of the security context are: (i) regular security actions such as identification and encryption/decryption; (ii) types of violation to the security that have happened, and (iii) corrective security actions in case of any attempt to misuse a resource.

References

[1] L. Ardissono, A. Goy, and G. Petrone, "Enabling Conversations with Web Services", in *Proc .of The Second International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS'2003)*, Melbourne, Australia, 2003.

[2] L. Barkhuus and A. Dey, "Is Context-Aware Computing taking Control away from the User? Three levels of Interactivity Examined", in *Proc. of The Fifth International Conference on Ubiquitous Computing (UbiComp'2003)*, Seattle, Washington, USA, 2003.

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

[4] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, "A Foundational Vision for e-Services", in *Proc. of The Workshop on Web Services, e-Business, and the Semantic Web (WES'2003) held in conjunction with The 15th Conference On Advanced Information Systems Engineering (CAiSE'2003)*, Klagenfurt/Velden, Austria, 2003.

[5] M. Bonett, "Personalization of Web Services: Opportunities and Challenges", *ARIADNE*, 28, June 2001.
http://www.ariadne.ac.uk/, ISSN: 1361-3200.

[6] M. Breener and M. Schiffers, "Applying Web Services Technologies to the Management of Context Provisioning", in *Proc. of the 10th Workshop of the OpenView University Association (OVUA2003)*, Geneva, Switzerland, July 2003.

[7] P. Brézillon, "Focusing on Context in Human-Centered Computing", *IEEE Intelligent Systems*, 18,3, May/June 2003.

[8] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", *Human-Computer Interaction Journal, Special Issue on Context-Aware Computing*, 16,1, 2001.

[9] J. E. Hanson, P. Nandi, and D. W. Levine, "Conversation enabled Web Services for Agents and e-Business", In *Proc. of The 2002 International Conference on Internet Computing (IC'2002)*, Las Vegas, Nevada, USA, 2002.

[10] J. Heflin and M. Huhns, "The Zen of the Web", *IEEE Internet Computing*, 7,5, September/October 2003.

[11] F. Lin and D. H. Norrie, "Schema-based Conversation Modeling for Agent-oriented Manufacturing Systems", *Computers in Industry*, 46,3, October 2001.

[12] Z. Maamar, B. Benatallah, and W. Mansoor, "Service Chart Diagrams - Description & Application", in *Proc. of The Alternate Tracks of The Twelfth International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, 2003.

[13] Z. Maamar, S. Kouadri Mostéfaoui, and E. Bataineh, "A Conceptual Analysis of the Role of Conversations in Web Services Composition", in *Proc. of The 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*, Taipei, Taiwan, 2004.

[14] Z. Maamar, S. Kouadri Mostéfaoui, H. Yahyaoui, and W. J. van den Heuvel, "Towards an Agent-based and Context oriented Approach to Compose Web Services", in *Proc. of The 6th International Conference on Enterprise Information Systems (ICEIS'2004)*, Porto, Portugal, 2004.

[15] Z. Maamar, H. Yahyaoui, Q. H. Mahmoud, and F. Akhter, "Dynamic Management of UDDI Registries in a Wireless Environment of Web Services", in *Proc. of The Second International Conference on Wired/Wireless Internet Communications (WWIC'2004)*, Frankfurt (Oder), Germany, 2004.

[16] D. A. Menascé. "QoS Issues in Web Services", *IEEE Interne Computing*, 6,6, November/December 2002.

[17] C. Muldoon, G. O'Hare, D. Phelan, R. Strahan, and R. Collier, "ACCESS: An Agent Architecture for Ubiquitous Service Delivery", in *Proc. of The Seventh International Workshop on Cooperative Information Agents (CIA'2003)*, Helsinki, Finland, 2003.

[18] M. Ouzzani and A. Bouguettaya, "Efficient Access to Web Services", *IEEE Internet Computing*, 8,2, March/April 2004.

[19] C. Panayiotou and G. Samaras, "mPERSONA: Personalized Portals for the Wireless User: An Agent Approach", *Journal of ACM/Baltzer Mobile Networking and Applications, Special Issue on Mobile Commerce*, 2003.

[20] M. Paolucci and K. Sycara, "Autonomous Semantic Web Services". *IEEE Internet Computing*, 7(5), September/October 2003.

[21] M. Papazoglou and D. Georgakopoulos, "Introduction to the Special Issue on Service-Oriented Computing", *Communications of the ACM*, 46,10, October 2003.
[22] W3C Consortium, "Semantic Web Activity", www.w3.org/2001/sw.
[23] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications", in *Proc. of The IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, USA, 1994.
[24] J. Sun. "Information Requirement Elicitation in Mobile Commerce", *Communications of the ACM*, 46,12, December 2003.
[25] M. Tian, A. Gramm, M. Nabulsi, H. Ritter, J. Schiller, and T. Voigt, "QoS Integration in Web Services", in *Proc. of The Ph.D. Students Workshop on Technologies and Applications of XML*, Berlin, Germany, 2003.
[26] A. Uszok, J. Bradshaw, R. Jeffers, M. Johnson, A. Tate, J. Dalton, and S. Aitken, "Policy and Contract Management for Semantic Web Services", in *Proc. of The 2004 AAAI Spring Symposium on Semantic Web Services Series*, Stanford, CA, USA, 2004.