Abstract—One of the major challenges faced by the Web services platform is how to compose a more appropriate new value added web service according to the user’s requirements and preferences, and adapt it at runtime. To this end, we suggest in this paper a QoS-oriented approach to build more efficient composite web services and personalize this composition on-the-fly by taking into account user context, requirements, and preferences and also non-functional properties of web services involved by this composition. These non-functional properties concern the Quality of Service (QoS) as well as the context of service execution.

Keywords-personalization; context aware; QoS; web services

I. INTRODUCTION

Web services framework, which is an instantiation of SOA, is nowadays the most well-known connection technology for implementing SOA concept. It consists of XML vocabularies to express protocols, service interfaces, registration of services, service discovery, policies, security, privacy, etc. In this framework, a new added value web can be produced by composing the existing ones, using composition languages such as BPML[1], XLANG[2], BPEL4WS[3], etc. This characteristic has already gained enormous traction in the industry by creating enterprise application integration, business to business relationships, etc.

However applications on SOA introduce new challenging problems, especially the capacity to offer to users a flexible approach to create and personalize on-the-fly this new added value according to the user’s context evolution including their preferences and the real world features such as technological restrictions, mobility requirements, and resources constraints and also its evolution during the composed web service execution. For instance, certain users may wish to have access using their PDA to a web service to extract, browse and either personalize information published by YouTube website. In addition they may prefer that the required web service will be adapted on-the-fly according to continuous variations of the execution environment. For example if their PDA performance is decreasing, the composite web service will be adapted dynamically by eliminating video web service from the composition.

The aim of the work we present in this paper contributes to address this lack by proposing a flexible architecture able to compose a more appropriate new value added web service according to the user’s preferences and requirements, and also to adapt it on-the-fly if any change related to user’s context and/or web service context is detected. Generally, context is the information that characterizes the interaction between users, applications and the surrounded environment. User context refers for example to user configuration (e.g. configuration of a used terminal, availability of critical resources, etc.), whereas web service context refers to parameters that concern for instance its availability, network-related characteristics, etc. in addition to QoS parameters that refer to how well a service performs its behaviors. In general, each context contains both static and dynamic aspect. The significant changes related to dynamic aspect of both contexts are handled dynamically by specific component integrated in the proposed architecture.

As described above, personalization in web service framework presents a multitude of challenges. These challenges can be summarized as follow: (a) creation of appropriate composite web service according to user context, needs and preferences, (b) handling the diversity of QoS specification language of web service, (c) determination of the best basic web service involved by composition process and (d) dynamic adaptation and/or repair of composite web service regarding web service context and user context, requirements and preferences.

Moreover, these three challenges are dependent and make more difficult our task to determine a more required available service according to users needs.

In the literature, few studies are focused on these four challenges simultaneously. The work presented in [4] proposes a context based approach for personalizing web services composition, but it does not address (b) and (d) challenges. Another interesting work that takes into accounts both context and QoS in [5] is limited only to discovery and selection process of web service without focusing on composition process. In the same direction, several approaches like [6] have been proposed in this area, but not meet all evoked challenges simultaneously. Therefore this lack motivates us to focus our study to meet the required challenges faced by web service personalization.

This paper is organized as follows: section 2 presents the web services personalization views. Section 3 provides details on user context and web service context. Section 4 exposes the main component of the proposed architecture. Before concluding, Section 5 introduces an application case study considering a composed web service using YouTube as a web source.
II. WEB SERVICES PERSONALIZATION

Generally, 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 need [7,8]. User-information includes user context execution, interests, desires, needs, etc. Depending on cases, some or all of these informations need to be captured and then processed to meet user’s requirements and preferences.

Many ways are used to personalize interaction mode with users. For instance, it’s possible to determine a user’s interests from his clicks or by analyzing previous navigation. Another case is to construct a user profile that leads to the explicit representation of the user’s interests. Unfortunately, this method is efficient under the assumption that the profile may remain static and not change. On the contrary, we have to add learning methods to update the profile according to the evolution of the user’s interests. Alternatively, the web usage mining method learns about users from their behavior. In fact, history about user actions is collected and used to record a track of the used services, including which method is used and so forth. Another way is to use the preferences of a certain community to predict appropriate content or services. A crucial problem is to measure the effectiveness of a personalization web service that involves defining metrics and feedback techniques [9].

The personalization represents either one of the major challenges faced by Services Oriented Architecture, especially in Web service framework. Personalization becomes more important when applied to composite Web services. Composite web services are services that invoke other web services contrarily to basic web services that do not rely on any other web service. They represent a new added value that can be produced by composing the existing ones, using composition languages.

In this direction, personalizing composite web service means that process business processes, representing a composite Web services, usually have to be updated and evolved over the time in order to be adapted to the changing requirements. So without an appropriate approach, the task of integrating dynamic personalization of business process may become a tough task.

To address Web services composition regarding to the context constraint evolution, our approach is based on BPEL language, which is currently the preferred standard for Web service composition and implemented by many vendors. More over there is currently several open source implementation of BPEL specifications.

According to the personalization taxonomy described in our previous work [10], the setting up of an architecture for a dynamic personalization must take into account a variety of changes. These changes may concern not only requirements such as defining some preferences on the execution environment, associating a provided application to a specific terminal, and so on, but also it must take into account the context of the used Web service to more satisfy the client request. In this direction, we have integrated in our approach some policies in XML format representing contexts related to the user and either to Web services. These policies are described more precisely in the following subsection.

III. USER AND WEB SERVICE CONTEXTS

A. Context background

To provide adequate service for the users, applications and services should be aware of their contexts and automatically adapt to their changing contexts-known as context-awareness [11]. Therefore context is very important, since it provides information about the present status of applications, services and devices in the environment.

Context is defined by Dey [11] as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including location, time, activities, and the preferences of each entity.

A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use [12]. One goal of context-aware systems is to acquire and utilize information on the context of a device in order to provide services that are appropriate to the particular people, place, time, event, etc.

In our case, we consider two types of context: user context and web service context.

B. User’s Context

User context covers all information related to user device, its preferences, the state of the execution environment, etc. at a given time. Therefore and in order to offer to users a personalized composite Web services on-the-fly towards technological restriction such as using for example a particular terminal and/or a specific Network Access Point, and so on, we have introduced a XML structure named UserConfiguration that will permit to specify all features bound to the corresponding configuration needs for each user.

An example of such structure is described as follows:

```
<Configuration>
  <Terminal type = PDA />
  <parameters name="Resolution" type="LCD" value = "160x160" unite="pixel"/>
  <parameters name="memory" values="2" unite="MB"/>
  <parameters name="CPU" value = "300" unite="Mhz"/>
  <parameters name="OS" value="Windows CE"/>
</Terminal>
```

To state the execution environment related to the user context, we have placed a monitoring framework, which is composed by a collection of probes, to detect at runtime all the required information about the user context execution. This information concerns for instance the knowledge of the availability of certain resources to decide if some reconfigurations and/or adaptations may be performed, etc. All these properties are expressed in a XML format, named UserExecutionContext. An example of such politic is as follow:

```
<context-policy name="user-context-execution-policy">
  <property-value name="/system/network.bandwidth" value="40000" unite="Kbps"/>
  <property-value name="/system/battery-life" value = "360" unite="s"/>
</context-policy>
```
In addition, users may wish to express their preferences to get a personalized display, a required language, etc. To express all these constraints, the user uses a special XML file, named UserPreferences.xml. The following code shows a simple example:

```
<preference-policy>
    <multimedia>yes</multimedia>
    <qos>high</qos>
    <language>English</language>
    <output>html</output>
</preference-policy>
```

C. Web service Context

In a competitive business environment, the service providers become increasingly confronted with the customer's requirements. In this fact, they try to ensure a high quality of their provided services in order to gain the confidence of their customers. In this direction, it is necessary to publish not just a WSDL description of their Web services in UDDI registry but also its associated characteristics related to its execution context. This context includes all important parameters of the required resources that are recommended to invoke and run such Web service. These parameters concern for instance which kind of terminal is to use, the required memory or CPU, and so on. The described information concerns QoS description of the published web service. In this fact, this way becomes a main key for differentiating between providers offering the similar Web services.

To describe QoS information about web services, service provider may use a specific QoS specification language. However several QoS specification languages can be used, such as WSLA, WS-Policy, WS-Agreement, etc. This diversity of languages introduces a new problem of analysing QoS information heterogeneity to determine which published service is more adequate to the user context and preferences at the moment of sending his request.

In this case, two challenges are exposed (1) How to handle this diversity of QoS Specification? (2) How to identify QoS criteria for each QoS specification since that there is no standard QoS specification? In this way, we have introduced a specific component in our architecture which is responsible for introducing required parsers for each QoS specification that will be loaded at runtime according to QoS specification used by the service provider. In the other hand and since service providers use divers QoS specification to describe QoS information related to theirs published services, we have interested to parse only those are more important common criteria for service as a first objective. These criteria concern for example network bandwidth, memory, cpu capacity, etc.

Another aspect of web service context, which is qualified dynamic, concerns for example availability of web service (related to the number of session that can be managed by a server), response time of a service to a user request.

IV. THE PROPOSED ARCHITECTURE

This section presents the main features and components of our proposed architecture for context-and QoS aware personalization for Web Services. This architecture has the advantage to deliver a required Web service by composing Web services according to the users' context, requirements and preferences. This composition is guided by the user to order the sequence of execution services composed just for the first time (semi-automatic composition). After that the system based on our architecture will make the decision itself.

The proposed architecture has in addition the ability to adapt dynamically this composition by selecting the most appropriate Web services related to the evolution of the used context. This context takes into account many factors such as changes related to user configuration needs, user context execution, and the Web service context.

The proposed architecture includes six main modules:
1. an extended UDDI repository that adds an entry for QoS information, mainly expressed as an UDDI Tmodel (technical model).
2. Queries Analyzer Module (QAM).
3. User’s Context Module (UCM) which is a collection of software sensors for user context deduction.
4. WS Discovery & Composition Module (WSDCM).
5. BPEL orchestration engine for invoking the composed WS.

This architecture is shown in Figure 1.

A. Description of the different modules of the architecture

The QAM holds primitives required for analyzing user request to determine which Web services are needed to be composed to deliver a required Web service.

To provide a better Web service to the user according to his/her needs, the UCM is introduced to determine user’s configuration and also to measure at runtime the important parameters of its execution context such as CPU usage, battery life, and so on. This module is based on a collection of software sensors that informs dynamically the system if the value of some parameters is changed and is less than the value recommended by a required Web service include by the composition process.
The WSDCM represents the central and important module of our architecture. Its main function is to receive: (1) which Web services are required to be composed from QAM (2) which sequence of Web services is to compose which guided by the user (semi-automatic composition) and (3) the user context and configuration information from the UCM to compose an appropriate composite Web services according to the user’s requirements. The appropriate composition is determined by this module by selecting the better Web service among several Web services registered in the enriched UDDI delivering the same functionalities, by comparing user context information and the registered QoS part of the published web service context. Once the selection is done, the WSDCM generates a BPEL process of the composed Web service and then asks the BPEL engine to handle it.

To be in harmony with the significant evolution of user’s and Web service context, we have introduced a Module, named Personalization Monitoring Module, which has the advantage to detect continuously parameters of the current context (dynamic part of user context and web service context (cf. Section 3)) to detect if the composite WS is still conform to the actual user context used or not. If not, it will notify the WSDCM to reconfigure the used composition.

B. Behavior of the proposed architecture

The aim if this subsection is to present how our proposed architecture works by describing links between modules:

(1) A user requests for a Web service.

(2) The UCM will collect information about user context including its preferences and the real world features as technological restrictions, mobility requirements, resources constraints, etc.

(3) The Web service QAM analyzes the user request to determine which Web services are needed to be composed to deliver a required Web service and then orders the WSDCM to perform the required composition taking into account the user context.

(4) After receiving the order from QAM to compose a required composite Web services, the WSDCM asks the UCM to have the information about the user context and also the user to get the required sequence of web services to be composed.

(5) Once the WSDCM has in its possession either a list of Web services to compose and the actual context of the user, it will search in the enriched UDDI, including the QoS of Web Service, which one is the most appropriate among the published Web services to the current context of the user. Recall that QoS information is obtained by using the appropriate XML parser for the used QoS specification language.

(6) The WSDCM generates a corresponding BPEL process related to the composed Web service and then invokes the BPEL engine to perform it.

(7) To be aware of either related to the user context and the Web service context, the Personalization Monitoring Module has the responsibility to detect at runtime significant changes of user and Web service contexts evolution.

(8) If PMM detects that user and/or Web service context is/are decreasing, it will notify the WSDCM to change dynamically the used composition related to the current user and/or Web service context. This can be done by substituting some Web services by other and/or eliminating some Web services from the used composition. After that, the architecture will restart the steps (4) and (6).

V. YOUTUBE’S COMPOSED WEB SERVICE CASE STUDY

To illustrate more our approach, we present a scenario based on YouTube information that is wrapped within a composed web service. This later is personalized according to user's different contexts. This scenario is presented as follow:

- **User's context**: a user connects and specifies his preferences. In addition, the system will instantiate the user's other parameters considering his device, network features, etc. All these parameters represent a user context and its description look as the following code:

  `<context user="JZQM">`
  `<preference>`
  `<multimedia>yes</multimedia>`
  `<qos>high</qos>`
  `<language>English</language>`
  `<output>chTML</output>`
  `</preference>`
  `<configuration>`
User's Query: Consider that the user's Query is "Georges Bush". Consequently, we will search videos related to Georges Bush from the web source YouTube.

Services discovery: using the UDDI, the following services are discovered:
- YouTube information extraction
- Translation from English to French
- Information representation
- Video streaming
- View generator

YouTube information extraction WS: YouTube

Results for the query "Georges Bush" is an html document:

See [13,14] for more details on web services dedicated to information extraction from the web.

- Translation from English to French WS: Translate the English description to French: "Amitié pour la majeure partie d'une décennie, les tensions américano-russe ont grandi .... Al Jazeera Anglais George Bush, Vladimir Poutine Jonah Hull USA Russie".

- Information representation WS: XML representation of the previous item extracted:

View generator WS: View generation according to the user's context including his device. Web page generation using the specific language like html, xml or wap. This page is then returned to the user.

In conclusion, the new composed service is then executed. The BPEL generated by the WSDCM is as follow:

<sequence>
<invoke partnerLink="YouTubeInformationExtractionPartner"
portType="taw: InterfaceOfLoadPageService "
operation=" YouTubeInformationExtraction" inputContainer="ytieic"
outputContainer = "ytieoc">
QoS aware personalization in SOA environment, more used after the first composition. XML relating to contexts of both user and web services detected by monitoring process and those expressed in the current values of the parameters of the resource issue composition. It is deducted automatically by comparing the context of all web services included in the Personalization Monitoring Module because it is aware of composition. This decision was taken by the streaming video service is stopped and deleted from constraints defined by the new context. Consequently, the system discovers this performance decreases.

- Discover new services according to the new context properties:

  There is no video streaming services with the current constraints defined by the new context. Consequently, the streaming video service is stopped and deleted from composition. This decision was taken by the Personalization Monitoring Module because it is aware of the context of all web services included in the composition. It is deducted automatically by comparing the current values of the parameters of the resource issue detected by monitoring process and those expressed in XML relating to contexts of both user and web services used after the first composition.

VI. CONCLUDING REMARKS

The contribution of this paper deals with context-and QoS aware personalization in SOA environment, more precisely in web services framework. The suggested approach takes into account processes of publishing, discovering and selecting web services regarding to the user context, requirements and preferences. This is done by integrating in the proposed architecture a monitoring framework for detecting continuously variations in user’s context.

In addition, the proposed architecture has the capability to offer a required web service in harmony with continuous variations of the execution environment including both user and web service contexts. This capability is made by adapting the composition process used to build an appropriate composed web services by eliminating, replacing and/or adding web services into the already used one.

Concerning future works, we are currently working on developing the proposed architecture and enhancing different policies presented with more parameters. Also we are interesting to integrate a composition on-the-fly rather than semi-automatic composition.

REFERENCES