Journal of Universal Computer Science, vol. 20, no. 7 (2014), 944-963
submitted: 31/1/14, accepted: 30/6/14, appeared: 1/7/14 © J.UCS

A User-Aware Approach to Provide Adaptive Web Services

Chiraz El Hog, Raoudha Ben Djemaa, Ikram Amous
(MIRACl ISIMS, Sfax University, Tunis Road Km 10, 3021 Sfax, Tunisia
elhog.chiraz@gmail.com
Raoudha.Bendjemaa@isimsf.rnu.tn, Ikram.Amous@isecs.rnu.tn)

Abstract: Web services are rapidly gaining acceptance as a fundamental technology in the web fields. They are becoming the cutting edge of communication between the different applications all over the web. Because of today's wide diversity of devices together with the variety of the user's preferences, context-aware web services are becoming a fundamental challenge that must be targeted. This issue is a part of the Human Computer Interaction (HCI) discipline and it aims at adapting the web service behavior according to the user's context such as his specific work environment, language, type of Internet connection, devices and preferences. Many solutions have been proposed in this area. Nevertheless, the adaptation was carried out only at the runtime and it partially covered the user's general context. In this paper, we introduce a new context-aware approach that provides adaptive web services. Our approach allows to express requirements by taking into account potential user's profile in addition to the functional one. While the latter ensures the description of the web service-functionalities, adaptation expresses the ability of a service to be self-adapted to runtime context changes. Our approach deals with adaptation from the very beginning of the modeling step of a web service. Furthermore, it upgrades description and publication usual methods in order to support profile specification.
Key Words: web service, HCI, adaptation, wsdl, uddi, uml
Category: H.3.5, D.2.2, I.5.2, H.2.3

1 Introduction

Service Oriented Architecture (SOA) is an emerging software design paradigm for building flexible and loosely-coupled distributed systems. SOA is based on services that could be used individually or composed with other services in order to offer a complex functionality. One of the most widespread SOA implementations is web Service. It uses Internet as a communication infrastructure between software components and takes the risk of using highly popular and inexpensive standards. Web services enable universal interoperability and focus on messages.

The web service provider exposes the features of the service via an abstract interface described by an XML-based language called web Service Description Language (WSDL). Published web services are then discovered through registries and repositories built upon a standard called Universal Description Discovery and Integration (UDDI). Thanks to this registry, businesses worldwide and services can be listed and dynamically located on the Internet. It is designed to be interrogated by SOAP (Simple Object Access Protocol) messages and to provide access to WSDL documents which describe the protocol bindings and message formats required to interact with the concerned web service.

With the emergence of new technologies and the diversity of users, web services should be upgraded to fulfill the user's needs. The offered services should be able to meet each user's profile regarding his context (location, device, connection) and his preferences (content and presentation).
For instance, more and more mobile devices, such as smart phones, I pads, pcs have been recently used to access web services via mobile communication. Nevertheless, most of the existing web services are not originally designed for mobile devices and did not take into consideration the user's interaction environment. Therefore, context-aware web service has become an important field of study. It relies on the intersection of the web service technology and the Human Interaction Computer discipline and it aims at enhancing the user's satisfaction. Context-aware Web service is based on the adaptation principle that makes services able to dynamically adjust their behavior to the user's profile (context and preferences) without human intervention. Consequently, tools and mechanisms are needed to afford a set of designed and developed user-centered Web services.

Some approaches have been proposed in the literature to support the development of adaptive web services. However, proposed solutions focus on improving the service-use while they do not deal with the development process. [El Asri et al. 2009] have integrated the notion of user's context in the web service description level. [Parimala and Saini 2011] proposed X-WSDL as an extension of the wsdl standard by adding new attributes 'criteria name' and 'description' to the service element. [Benaboud et al. 2010] adopted semantic web solution by enriching the user's query with his preferences. A refinement of the query is then performed according to the user's preferences. [Azmeh 2011] used the concept lattice approach to propose a discovery and selection method based on user's context and desired QoS. All these proposals are ad-hoc solutions for a particular domain or at a specific level of the web service life cycle. Whereas, adaptation must be formalized earlier in the web service life cycle, from the very beginning of the designing step, in order to provide a faithful adaptation framework. In addition, they restrict the user's profile to few characteristics of his environment and preferences. They mainly deal with human users of the service and focus on the adjustment of web service features. In fact, there are gaps in the existing software component development methodologies as they do not include the adaptation requirements and contextual information. This leads to a somehow low quality and irrelevant query results while it helps discovering and locating services with the existing service description and publication techniques. So the major challenge is to support service adaptation through all the service life cycle levels and to provide a coherent adaptation framework.

The present paper, proposes a new user-centred approach for the development of context-aware web services. Our proposed solution AWSF (Adaptable web service Framework) deals with the main steps of the web service development life cycle (the modeling, the description and the publication steps). It offers the ability to represent and provide contextual information in addition to web service features. Thus, it provides web services that are liable to adjust their behavior according to a customer's needs dealing with his preferences (color, text, image) and context (screen size, memory, wifi, 3G,...).

The remainder of this paper is structured as follows. Section 2 gives an overview of our proposed framework. Section 3 describes our proposal in the modeling layer. The description layer is presented in section 4. Section 5 describes the publication layer of our framework. To reveal the validity of our framework we propose in section 6 an illustration of use. Section 7 discusses some related works and section 8 summarizes the main results and gives some directions for ongoing and future work.

2 AWSF: Adaptable web Service Framework

The need of managing adaptive web services impels the incorporation of facilities to deal with user's profile in all the web services life cycle. We define the user's profile as a set of context information composed of device characteristics, connection, location and a set of preferences in terms of content and presentation. To achieve this objective, we propose in this section a framework that supports the development, the description and the publication of adaptive web service (AWS). Our goal is to offer relevant and suitable information depending on the user's particular profile. Therefore, developers have to integrate software facilities dealing with the profile characteristics. Then, profile annotations must be used to complement WSDL descriptions. Moreover, UDDI must be extended to include contextual information of the web service in addition to the web service information currently stored in UDDI registries. In the case of the customer, selection tool must be able to detect his context and to recover his preferences with the aim of discovering and selecting the suitable web service. Our framework is organized in a layered architecture and it consists mainly of three layers providing modeling, description and publication solutions.
Figure 1 illustrates an overview of our framework's main layers. Each one describes our proposal to a given stage of the web service life cycle.

– Modeling layer: It focuses on the design of the application behind the web service. In fact, in this layer we propose a modeling language called AWS-UML [EL Hog et al. 2011] and a corresponding tool named ArgoAWS-UML [EL Hog et al. 2013a] which supports the new design elements proposed.

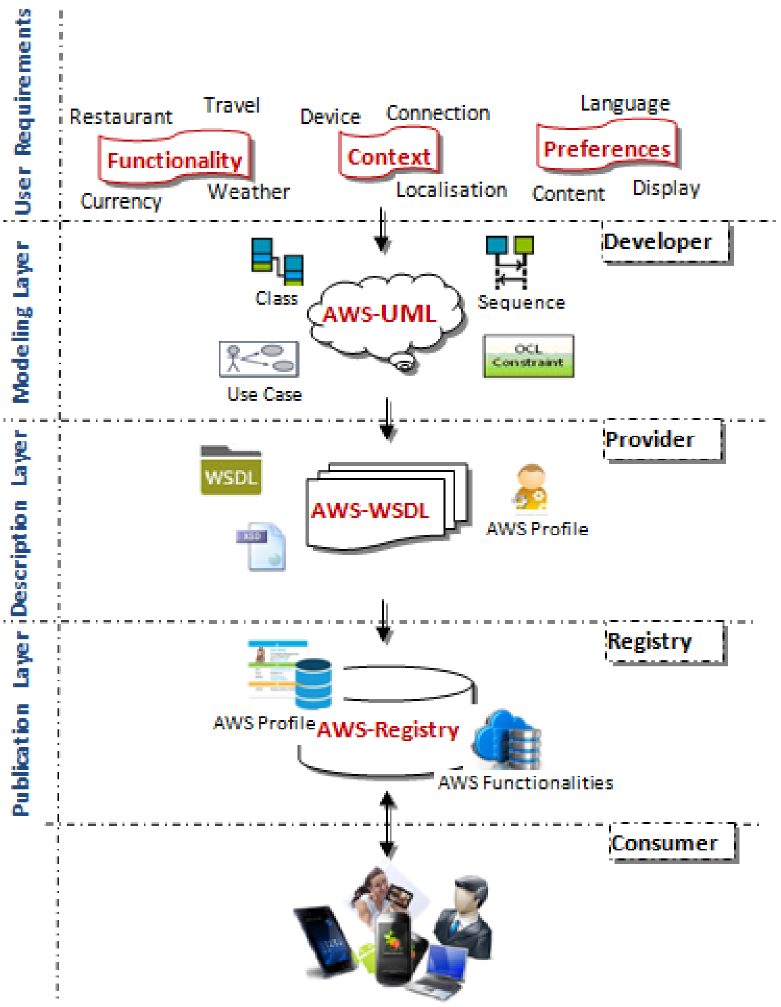


Figure 1: Adaptable web Service Framework's main Layers

– Description layer: It concentrates on the annotation of the WSDL description document with contextual information managed by the web service. The adaptable description named AWS-WSDL [EL Hog et al. 2012] is automatically generated from the service implementation using our AWS-WSDL generator.

– Publication layer: It gathers web services features and context supported on a UDDI extended registry named AWSRegistry [EL Hog et al. 2013b].

The following sections will concentrate on a detailed description of each layer.

3 Modeling layer

Software development methodologies, like object-oriented and component-based ones, cannot be blindly applied to SOA and web services. Creating wrappers on existing components to act as web services are not sufficient. In fact, most of existing applications take serious redesign effort to properly deliver components functionality through a web service. Moreover, the use of standard UML notation, as they are defined, seems to be too general for the purpose of describing web services and needs clarification and standardization of even basic terms like provider, consumer, etc... It also lacks semantics to express the web service non-functional aspects such as the ability of adaptation to the final user profile. To overcome these gaps, we proposed in [EL Hog et al. 2011] an UML profile named AWS-UML. Our proposal helps complete the UML models with semantics, stereotypes and constraints. It also allows the easy design of Web service characteristics together with the ability of adaptation. We defined adequate diagram extensions and meta models in our previous work [EL Hog et al. 2011]. We have also implemented a modeling tool which supports the new design elements that we named ArgoAWS-UML [EL Hog et al. 2013a]. It is based on the open source Argo-UML tool. Figure 2 shows inputs and the output delivered by the modeling layer.

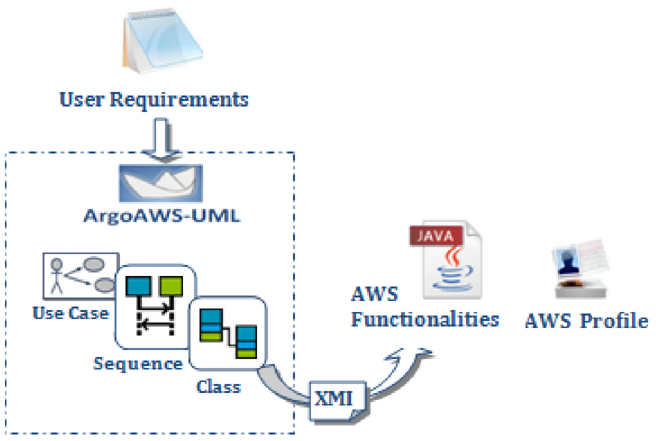


Figure 2: Modeling layer of the AWSF

The goal of user-requirements capture is to find out the informational and functional requirements of the web service. In our purpose, these requirements can be captured by Use Cases diagrams. This diagram is the starting point of our new notation. By analogy to UML, this diagram is represented by the two extended concepts: Actor and Use case. For instance, based on the web service participants, we classified web service actors into four categories: *Human consumer, Provider, Composite web Service and Applications.* We included these categories in the UML use case meta model. We also proposed new icons to represent these actors [EL Hog et al. 2011]. In addition, web service functionalities have been partitioned on the basis of interaction between web service participants. The use cases are organized into four categories: *publication, interaction, search and composition.* These groups of use cases could manage all the web service's life cycle. In the case of class diagram, AWS-UML uses the concept of VUML (View based Unified Modeling Language) introduced by Nassar ([Nassar 2005]) and offers the ability to model service accepted profiles when modeling web service features. In our work, we define: "a multi-view service as a first class modeling entity that highlights the actors' needs and requirements early in the life-cycle of the web Service development". In order to retrieve the most relevant results with the user's profile, we should discover all elements that influence the result. These elements are enclosed on actor's profiles. The *Base* class allows the representation of features required by all kinds of users. In contrast, the *View* class allows the representation of the features required by a specific user. We distinguish one *Base* and three kinds of *Views* related to users categories: *Composite web service View, web Application View, Provider View and Human Consumer View.* Every view is related to a profile modeling entity which describes profile characteristics supported by the web service features. The AWS-UML class diagram meta model provides stereotypes to represent adaptation rules associated with each profile cf figure 3.

Concerning scenarios, we proposed an enrichment of the exchanged messages with profile information. In fact, we need more precision to model objects and Messages exchanged according to their behavior. Therefore, we proposed in AWS-UML five kinds of objects described by new icons: *ObjectMultiviews, ObjectBase, ObjectProvider, ObjectHumanConsumer, ObjectCompositeWebService and ObjectApplication.* These objects interact while exchanging messages that can be simple messages or profile messages (containing profile features) labeled by the corresponding type: *PublicationMessage, SearchMessage, InteractionMessage, ProfileSearchMessage, ProfileInteractionMessage.*

To allow these diagram's extensions, we integrated all the new notions in the meta-model of the argoUML tool. Every diagram is enriched with new stereotypes, tagged values and OCL constraints. Given the user requirements, the designer and the developer could use our tool ArgoAWS-UML [EL Hog et al. 2013a] to model an adaptive multiview web service.

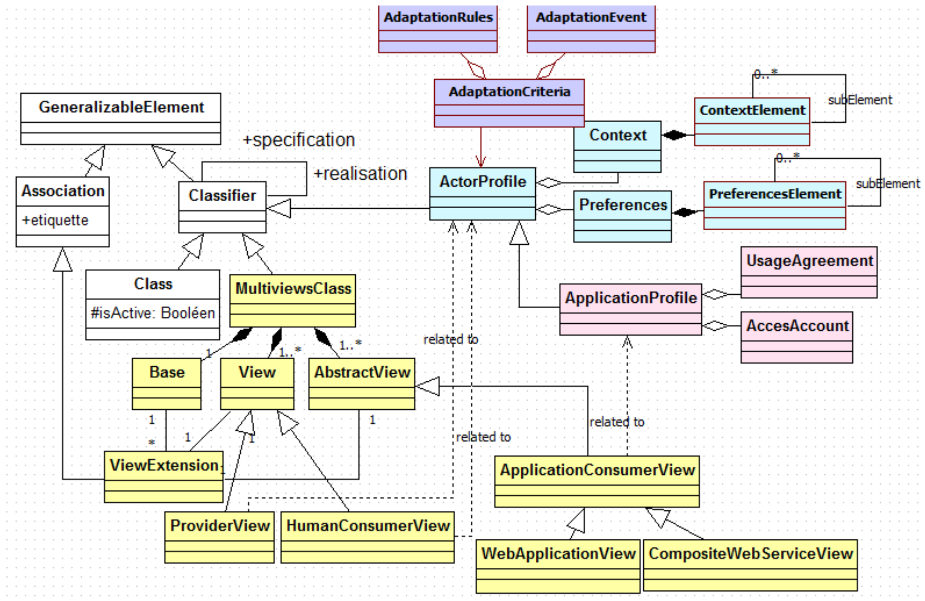


Figure 3: AWS-UML class diagram meta model

Description of a web service is generated after its deployment in a chosen web server. The description is provided by the standard WSDL and contains essentially a list of features offered by the service and links invocation. As we defined above a modeling approach taking into consideration the user's profile, we feel the need to extend the service description in order to accommodate extensions proposed in the modeling layer.

4 Description Layer

In order to allow the provider to publish adaptation criteria on the one hand and to allow the consumer to choose from a list of published Web services the most appropriate description according to his needs and profile on the other hand , we suggest additional information in the description file. We therefore propose an extension of the standard WSDL having the ability to describe service profile information. Our extension, named AWS-WSDL and based on the MDA [1] approach, is detailed in [EL Hog et al. 2012]. The AWS-WSDL document is the output of our description layer. It is obtained through a model transformation

[1] MDA: Model Driven Architecture

from the adaptive design described in the previous section. Figure 4 illustrates the transformation process in the description layer.

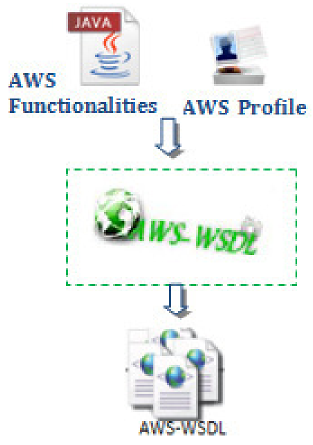


Figure 4: Description Layer of the AWSF

Below, we briefly show the process followed to achieve model transformation for generating the AWS-WSDL document based on the OMG specification of the MDA architecture. To allow the description of service profile, we combine in the same document the description of functional elements (operations, messages, types, ...) and profile information elements. Thus, we introduce new elements in the web service description document using meta-model extension. First of all, we elaborate the WSDL meta model using a class diagram. Then, we extend obtained meta model with profile model. As *PortType* is the interface for the description of operations offered by the web service, we propose the addition of a *UserCategory* and a *view* attribute. The *UserCategory* attribute is used to specify the user's profile which could invoke offered operations. This new attribute refers to an instance of the *UserProfile* element describing context and preferences. The *view* attribute is used to limit access rights to operations. It refers to the active *view* by the current *PortType*. The operations described by a *PortType* are the ones offered by the *view* in correspondence with the class of user and access rights and containing adapted operations to the user profile defined by the attribute *User Category*. Besides, the *UserProfile* class is composed of *AccessRights*, *Context* and *Preferences*. All these new elements added to the standard WSDL must be defined in the description schema. Therefore, we extend the WSDL schema by defining a new namespace http://localhost:8080/AWS-

WSDL/aws-wsdl. This schema in figure 5 will be stored in an XSD file and will be used to validate AWS-WSDL documents.

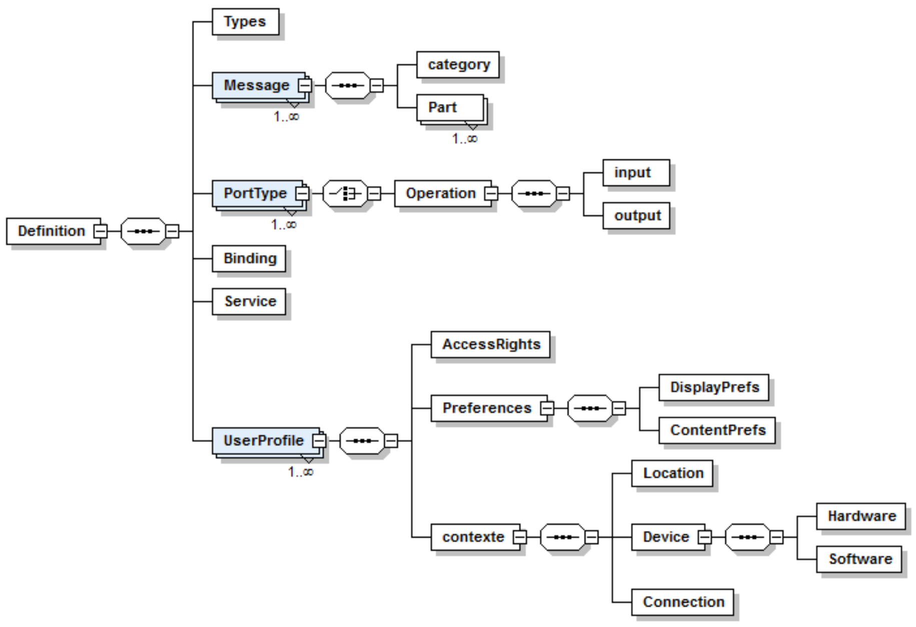


Figure 5: Extract from the AWS-WSDL schema

Having the new extension of the proposed standard WSDL in details, we realize it through a description generator allowing to generate the AWS-WSDL file from the service implementation. Our generator is used by the service provider and it is offered as an eclipse plug-in named *AWS-WSDL PLUG-IN* as illustrated in figure 6.

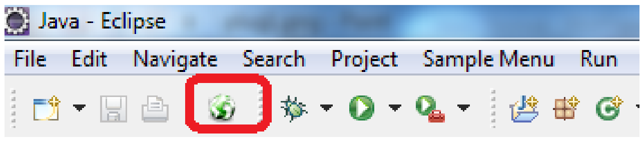


Figure 6: AWS-WSDL plug-in icon added to the Eclipse tool bar

It is a GUI tool that helps the provider to annotate WSDL document with service profile's information. We use the following list of open sources (Plugs/API): WSDL4J, Apache Axis2, JAVA2WSDL, Swing, JDOM to elaborate our tool. The web service description is published in the *UDDI* registry, from where the customer can reach the required service. In this case, the web service is described using AWS-WSDL. It should be published in an adequate registry supporting additional features. In the following section, we aim to define an *UDDI* extension.

5 Publication layer

A service provider publishes its service at a service registry using a publishing interface. The public interfaces and binding information of the registered services are clearly defined in the WSDL standard language. A registry organizes the published services and provides a query interface that enables a service consumer to search for a needed service, and obtain its provider's location information. A service consumer then interacts with a service provider through the SOAP protocol. The most known registry is UDDI which is an XML-based standard that was proposed for allowing providers to publish their web services, so that they can be located afterwards by consumers. A UDDI registry is structured into three components: *BusinessEntity* or white pages, *BusinessService* or yellow pages and *BindingTemplate* or green pages. These components allow the search for a suitable web service in the UDDI[2] registry according to the three data types. Unfortunately, web service discovery through UDDI-based registries is insufficient to carry out consumer adaptation requirements. This necessity is increasing every day with the emergence of portable devices and various users preferences. Hence, the provider and the consumer want to have a registry providing a fairly precise publishing and search capabilities to render the use of user-aware web services more efficient. To overcome these needs, we propose a new web service registry offering the ability to save features information as well as the adaptation criteria supported by each service. Our registry is called Adaptable web Service Registry AWSRegistry [EL Hog et al. 2013b] and it supports the publication of the AWS-WSDL description presented in section 4. We extend the UDDI data structure by an *AdaptationCriteria* element which makes reference to the item *UserProfile* in the description file AWS-WSDL.

When a supplier requests the publication of his service at the registry, an analyzer module retrieves the AWS-WSDL description file. Then features and access points description will be transferred to the AWSFunctionalities database and those corresponding to the adaptation criteria will be transmitted to the *AWSProfile* database.

[2] UDDI: $http://uddi.org/pubs/uddi_v3.htm$

To implement our registry, we propose an extension of the JuddiV3[3] API. Therefore, we use the apache Tomcat as a web server and the MySQL to implement the UDDI data Base. We offer a user-interface allowing the provider to specify enterprise and service information and to load the description file of its service. Then, a parser analyzes the AWS-WSDL file. It allows the extraction of the functional description carried out by the elements (types, portype, message, binding, service) and stores it in the database *AWSProfile*. Then it extracts the profile description contained in the element *UserProfile* and saves it in the database *AWSFunctionalities*. The $db - context$ table stores the profile of the web service. This table refers to the $binding - template$ table that stores web service instances. Our extended registry is compatible with the basic UDDI and both of them could coexist in the same environment. Publication process is shown in figure 7.

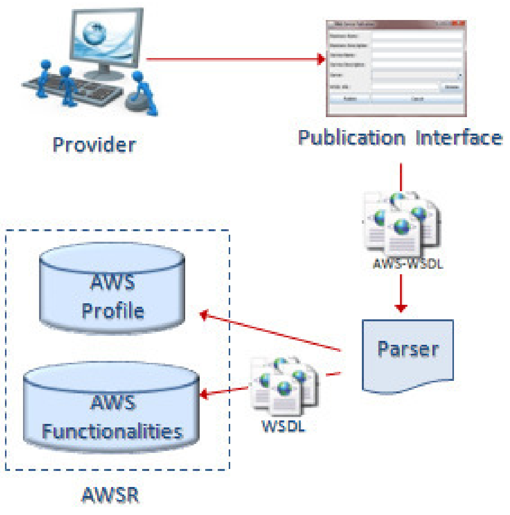


Figure 7: Publication process in the AWSRegistry

The AWSRegistry contains also a query analyzer module used in the selection and discovery phase. This module communicates with the database *AWSFunctionalities* and *AWSProfile* in order to select the suitable list of web services meeting user's query in terms of functionality and adaptation. The query analyzer allows the customer to:

– Extract the needs in terms of functionality through keywords,

[3] JUDDI: $http : //juddi.apache.org$

– Extract the needs in terms of adaptation through a comparison between the user's profile and the service profile stored in the AWSRegistry. The user's profile is automatically detected by a detection module named W-EDM (WildCat Extension Detected module) which is an extension of the WildCat API [4] and it is stored in an XML file.

Web service finding process over AWSRegistry is depicted by figure 8.

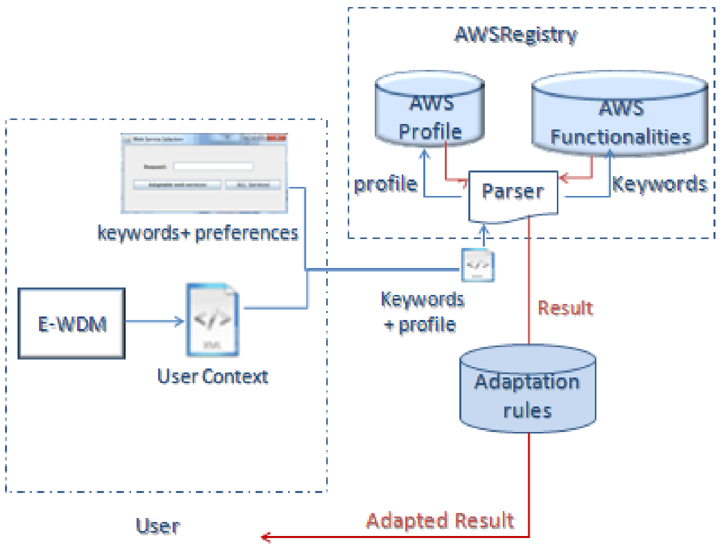


Figure 8: Finding web service process in the AWSRegistry

6 Illustration of Use

In this section, we illustrate the feasibility of our proposed framework AWSF through the example of a *Travel Agency* web service. The service client wants to book a room in a hotel on the basis of his profile characteristics. We provide corresponding solution to each layer.

6.1 Modeling layer

This layer's focus is on the design of the *Travel Agency* web service features as well as the web service adaptation ability. Figure 9 shows an extract of the class diagram in ArgoAWS-UML.

[4] http://wildcat.ow2.org/

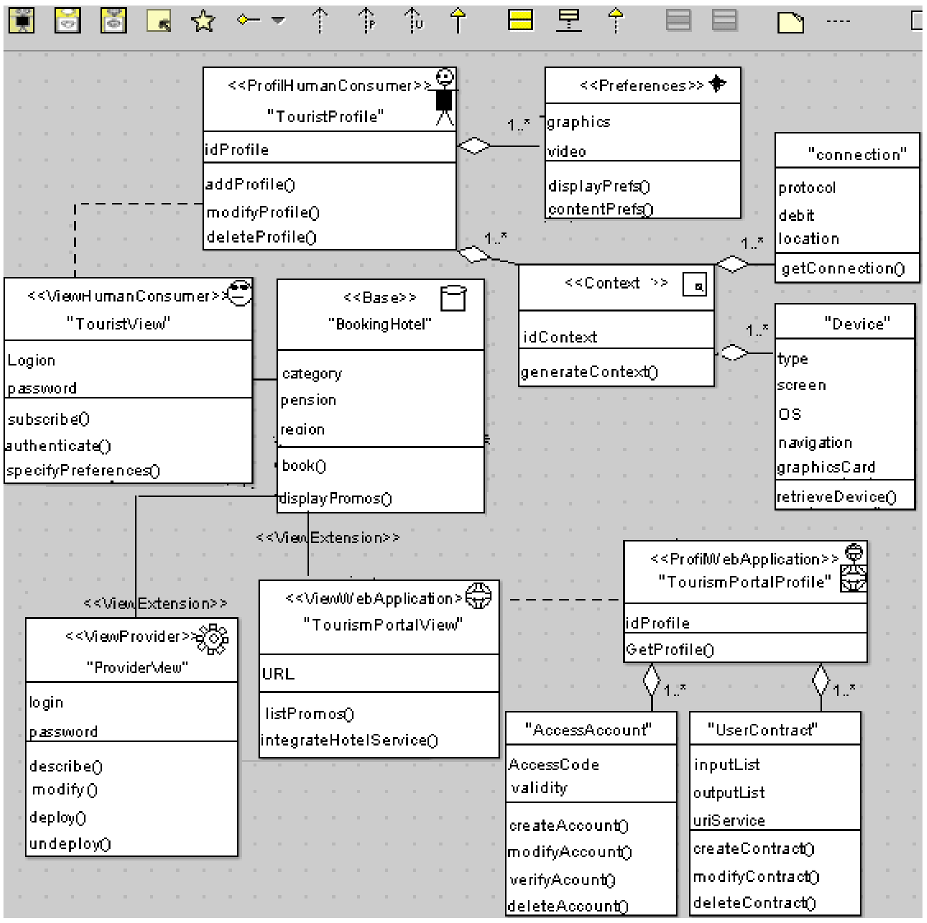
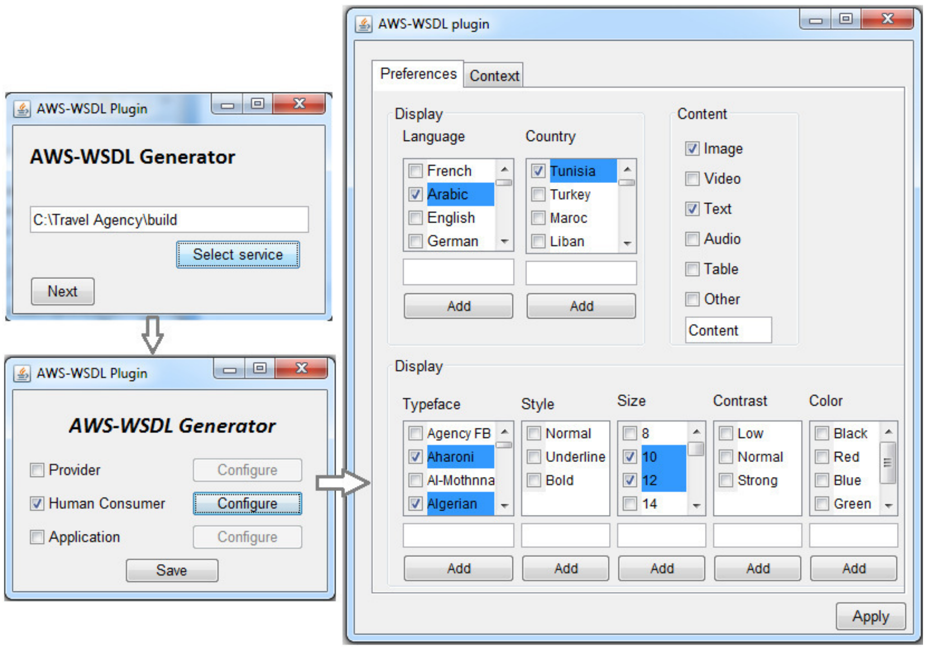


Figure 9: Extract of the class diagram of a Travel agency web service

In this example, we have a class stereotyped *Base* corresponding to the *BookingHotel* main feature provided to all kind of users. Classes stereotyped *Views* correspond to class of actors and roles that interact with the web service. We can distinguish the *TouristView* as a *HumanActorView* which invokes the service directly through its web interface. The *TourismPortal* as an *ApplicationView* that could use this service to retrieve the list of promotions or to add this service to its list of activities. A *ProviderView* is used to model supplier interaction with the web service for any update and deployment. To model adaptation criteria supported by the web Service, every view is related to a corresponding class profile (composed by context and preferences). The *ProfileHumanConsumer* class is aggregated with *Preferences* class allowing to represent the supported content and presentation preferences. Class *Context* allows to model environment characteristics. In the case of *WebApplicationProfile*, we have to model the usage agreement between the web service and the web application. In fact, interaction in that case could be dealt with a *servlet* sending a request containing the web application account (for authenticity), the user contract as input and output exchanged (in our case study it could be the credit card information) and the URI of the wanted web service.

6.2 Description layer

The *Travel Agency* web service modeled above using our tool ArgoAWS-UML is then implemented in Java. The next step is to generate the description file gathering all needed information about the available functionalities, access links and supported profiles. Figure 10 illustrates the AWS-WSDL generator interfaces.



AWS-WSDL Generator

C:\Travel Agency\build

AWS-WSDL Generator

Provider
[✓ Human Consumer]
Application

Display
Language: French, [✓ Arabic], English, German
Country: [✓ Tunisia], Turkey, Maroc, Liban

Content: [✓ Image], Video, [✓ Text], Audio, Table, Other
Content

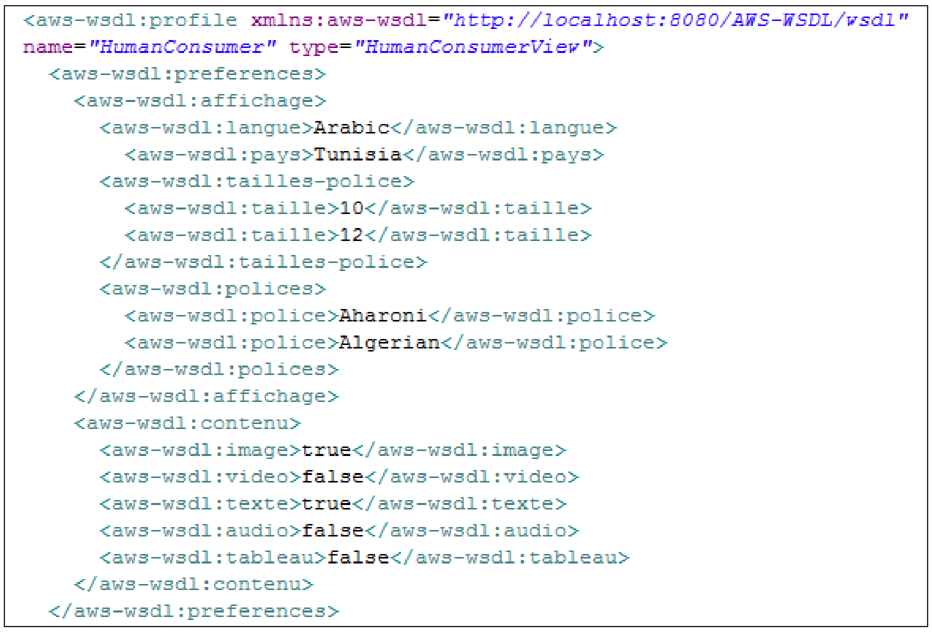
Display
Typeface: Agency FB, [✓ Aharoni], Al-Mothnna, [✓ Algerian]
Style: Normal, Underline, Bold
Size: 8, [✓ 10], [✓ 12], 14
Contrast: Low, Normal, Strong
Color: Black, Red, Blue, Green

Figure 10: AWS-WSDL generator GUI interfaces

As input, we provide the build of the compiled class of the *Travel Agency* multiview adaptable service. Then the provider could select, through the plug-in interface, a set of views to be included in the final description file. After that, the provider select suitable contextual information supported by the web service. In this example we choose to configure a human consumer view.

First, we select the multi-view service implementation. Technically, our AWS-WSDL plug-in checks whether multi-view and profile classes exist or not. Then, we choose the desired views (provider, human consumer or application) to be taken in the description file. According to the selected view, a corresponding User interface is proposed allowing to configure supported profile elements. In our case study we configure the preferences of the human consumer view. Clearly, for a chosen element, we can select multiple choices. For instance, police size could be ”10” or ”12”. Also, we can add new values in the proposed textfields. A temporary file is associated to every configured view. Then, they are gathered into the AWS-WSDL description final document.

An extract of the generated description file is illustrated in figure 11.

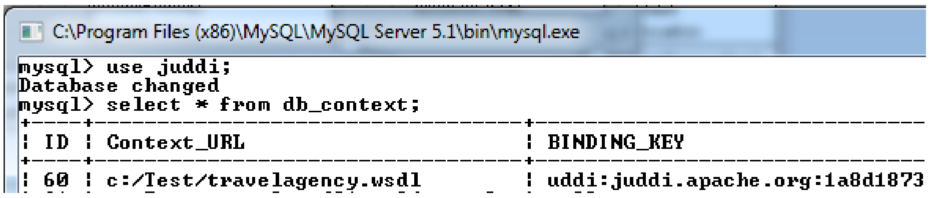


```
<aws-wsdl:profile xmlns:aws-wsdl="http://localhost:8080/AWS-WSDL/wsdl"
name="HumanConsumer" type="HumanConsumerView">
  <aws-wsdl:preferences>
    <aws-wsdl:affichage>
      <aws-wsdl:langue>Arabic</aws-wsdl:langue>
        <aws-wsdl:pays>Tunisia</aws-wsdl:pays>
      <aws-wsdl:tailles-police>
        <aws-wsdl:taille>10</aws-wsdl:taille>
        <aws-wsdl:taille>12</aws-wsdl:taille>
      </aws-wsdl:tailles-police>
      <aws-wsdl:polices>
        <aws-wsdl:police>Aharoni</aws-wsdl:police>
        <aws-wsdl:police>Algerian</aws-wsdl:police>
      </aws-wsdl:polices>
    </aws-wsdl:affichage>
    <aws-wsdl:contenu>
      <aws-wsdl:image>true</aws-wsdl:image>
      <aws-wsdl:video>false</aws-wsdl:video>
      <aws-wsdl:texte>true</aws-wsdl:texte>
      <aws-wsdl:audio>false</aws-wsdl:audio>
      <aws-wsdl:tableau>false</aws-wsdl:tableau>
    </aws-wsdl:contenu>
  </aws-wsdl:preferences>
```

Figure 11: Extract from the Travel Agency description file

6.3 Publication layer

Next step is to publish the AWS-WSDL description of the service in our AWSRegistry. The provider has to specify the enterprise information, the service information, the web server and the description file to load. The WSDL document enriched with the supported profile is saved in the *dbcontext* table (cf figure 12).



```
mysql> use juddi;
Database changed
mysql> select * from db_context;
+----+-----------------------------------+-------------------------------
| ID | Context_URL                       | BINDING_KEY
+----+-----------------------------------+-------------------------------
| 60 | c:/Test/travelagency.wsdl         | uddi:juddi.apache.org:1a8d1873
```

Figure 12: Publication of the Travel agency web service in the AWSRegistry

Below, we expose experiment results of the publication process in our AWSRegistry:

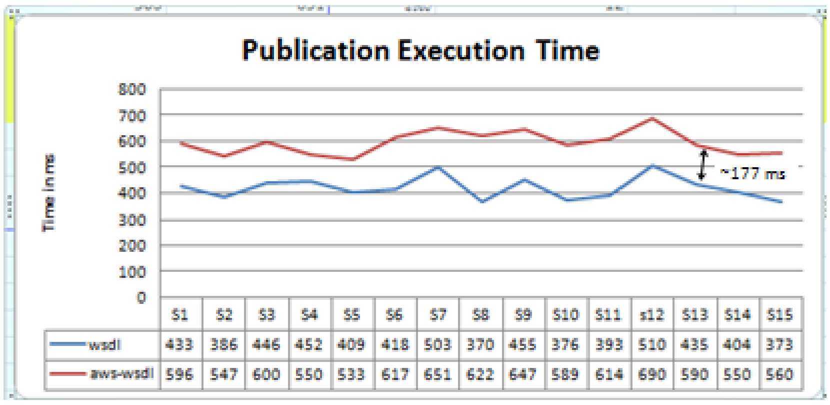
1. The first experiment determined the performance of the standard UDDI by measuring the speed of the publishing function dealing with WSDL document.

2. The second experiment determined the performance of the AWSRegistry by measuring the speed of the publishing function dealing with AWS-WSDL document.

Figure 13 demonstrates a comparison between the span of time taken to publish WSDL description and the one taken to publish AWS-WSDL description. WSDL documents used in these experiments were taken from the webservicex [5] portal.

It was noticed that the publication time are very close to an average spread of ≈ 177 ms.

[5] http://www.webservicex.net/



	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	s12	S13	S14	S15
wsdl	433	386	446	452	409	418	503	370	455	376	393	510	435	404	373
aws-wsdl	596	547	600	550	533	617	651	622	647	589	614	690	590	550	560

Figure 13: Execution time required to publish web services on the AWSRegistry

The AWSRegistry contains also a query analyzer module in the selection and discovery phase. This module communicates with the database *ServiceFunctionalities* and *ServiceProfile*, and, according to profile information, it makes web service selection possible for end users. When looking for a service, the query analyzer offered by the AWSRegistry allows the customer to: Extract the needs in terms of functionality through keywords and in terms of adaptation regards user's context detected and specified preferences. To measure the impact of the required time for the service selection, the processing time of service requests, considering the user's context and preferences, was compared to the request processing time for the same services considering only functionalities (Basic).
A scenario of web services descriptions already published into the AWSRegistry was simulated using SQL requests. Even though the evaluation results have shown an increase in the average processing time, they render the Web service more relevant.

7 Related Works

In this section, we take a look at some research works interested in the possibilities of applying the profile adaptation on web Services life cycle. We provide an overview of some of these works.

Many researchers attempted to enhance web service description with non functional attributes. [Lin et al. 2012] extended the WSDL structure to deal with web service QOS needs. [Parimala and Saini 2011] proposed an extension, named X-WSDL, of the wsdl standard by adding new attributes 'criteria name' and 'description' to service element. Using this extension, the web service developer must specify the criteria along with the service in X-WSDL document. The criteria are also specified by the user when invoking a service. [El Asri et al. 2009] proposed a model driven approach for the modeling of user-aware web services on the basis of the multiview component concept. The multiview component is a class modeling entity that allows the capture of the various needs of service clients by separating their functional concerns. This work takes into account the user's access rights to the web Service functionality. Despite that, the user preferences, device capacity, network characteristic, localization ... are not taken into account. However, there is a major lack in considering adaptation notion when describing or providing web services.

A number of research efforts have studied web services discovery and selection adaptation. [Benaboud et al. 2010] developed a framework for web Services discovery and selection based on intelligent software agents and ontologies. Ontologies are used to describe web services, QoS, customer's preferences and experiences. But, the proposed framework did not take into account mobile devices with limited capabilities neither network characteristic. [Soukkarieh et al. 2008] proposed a context-aware adaptation architecture based on web services named CA-WIS. The classical web service architecture extended with the AHA architecture and an adaptation layer fitted with various components dedicated to context management. Later, [Laborie et al. 2013] proposed a framework to implement the adaptation solution presented in CA-WIS.

[Azmeh 2011] proposed a selection framework that assists consumer to select suitable web services with respect to QoS policies in addition to functional requirements. This framework gave response to the discovery, classification and selection of a needed web service and it is based on techniques of concept lattice.

The work of [Hafiddi et al. 2012] proposed an architecture, named ACAS, to support the development of Context-Aware Service Oriented Systems (CASOS). This architecture relies on a set of context-awareness services specifications and metamodels in order to enhance a core service to be context-aware. This enhancement is fulfilled by the Aspect Adaptations Weaver (A2W) which is based on the Aspect Paradigm (AP) concepts.

Most of the previous studies dealt separately with different steps of web services life cycle. However, some researchers were concerned on one particular form of adaptation, like semantic enrichment of the client request by his context [Benaboud et al. 2010]. This request enrichment is not enough to deliver adaptive web services. It is also needed to integrate in the web service description file, the context in which it is adapted. In addition, contextual information partially covers the user general context ([El Asri et al. 2009]). Other studies have presented specific solutions to a range of use or the type of equipment used.

8 Conclusions and Future Work

The expansion of the web and the wide spread of the web service as a basic technology of universal interoperability through Internet have led to an increasing need of user-aware adaptation. The same service should be able to change its supplied functionality according to the user's specific need regarding his profile (operating system, programming language, preferences...). Thus, the users are an important participant in the web service's architecture. They search and select Web services using keywords and expect a suitable adapted result with all simple text and hypermedia. In the literature, multiple adaptation solutions have been proposed but most of them are restricted to the search and selection step of the web service life cycle. In this paper, we introduced an Adaptable web Service Framework (AWSF), that integrates adaptation to all the steps of the web service life cycle namely the modeling, description and publication steps. In each step, we proposed extensions of existing technologies in order to integrate adaptation components and notions. We defined a new user-centered design language that we called Adaptable Web Service UML (AWS-UML). AWS-UML extends the UML language with new stereotypes and constraints to model service profile together with functionalities. We then introduced a modeling tool that we called ArgoAWS-UML in order to model newly added concepts. The obtained model is the input of the description step, so we introduced a description language named AWS-WSDL that is derived from the WSDL language. Adaptable web services descriptions are then published in a specific registry that we called AWS Registry. It supports web service saving and searching according to adaptation criteria. We tested the performance of our AWSR and compared its robustness versus the standard UDDI registry.
To improve the user's interaction with the web service, we intend in the near future to design and implement an interactive web service selection tool. This tool provides human interface enabling users to specify their query and preferences. In addition it contains sensors that capture the user's context. Moreover, we aim at developing an adaptive web service generator [Djemaa et al. 2009] gathering all the AWSF opportunities. An ontology will be used to describe adaptation rules used to return the adaptive interface to the user.

References

[Azmeh 2011] Azmeh, Z.: "A web Service Selection Framework for an Assisted SOA"; Thesis, (2011).

[Benaboud et al. 2010] Benaboud, R., Sahnoun, Z., Maamri, R.: "User's preferences and experiences based web services discovery using ontologies"; Fourth International Conference on Research Challenges in Information Science, (May 2010), 121-126.

[Djemaa et al. 2009] Djemaa, B.R., Amous, I., Hamadou, B. A.: "Adaptability and adaptivity in the generation of web applications"; International Journal of Information Technology and web Engineering, 4, 2 (2009), 20-44.

[El Asri et al. 2009] El Asri, B., Kenzi, A., Nassar, M., Kriouile, A., Barrahmoune, A.: "Multiview Components for User-Aware web Services"; 11th International Conference on Enterprise Information Systems, (May 2009), 196-207.

[EL Hog et al. 2011] ElHog, C., Ben Djemaa, R., Amous, I.:"Towards an UML based modeling language to design adaptive web services"; International Conference on Semantic web and web Services, (July 2011), 38-44.

[EL Hog et al. 2012] ElHog, C., Ben Djemaa, R., Amous, I.:"Profile Annotation for Adaptable web Service Description"; Proc. 27th ACM Symposium on Applied Computing, (March 2012), 1935-1940.

[EL Hog et al. 2013a] ElHog, C., Ben Djemaa, R., Amous, I.: "ArgoAWS-UML: Modeling Tool for Adaptive web Services"; IEEE 37th Annual Computer Software and Applications Conference, COMPSAC Workshops, (July 2013), 169-174.

[EL Hog et al. 2013b] El Hog, C., Ben Djemaa, R., Amous, I.: "Adaptable web Service Registry for Publishing Profile Annotation Description"; IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 1010th International Conference on Autonomic and Trusted ComputingInt, (Dec 2013), 533–538.

[Hafiddi et al. 2012] Hafiddi, H., Baidouri, H., Nassar, M. , Kriouile, A.: "Context-Awareness for Service Oriented Systems"; CoRR, abs/1211.3229, (2012) .

[Laborie et al. 2013] Laborie, S., Soukkarieh, B., Sdes, F.: "On Using Generic Profiles and Views for Dynamic web Services Adaptation"; Journal of Modern Internet of things, 2, 3 (August 2013), 18-23.

[Lin et al. 2012] Lin, C., Kavi, K., Adepu, S.: "A Description Language for QoS Properties and a Framework for Service Composition Using QoS Properties"; The Seventh International Conference on Software Engineering Advances, (November 2012), 90-97.

[Nassar 2005] Nassar, M. : "Analyse/conception par points de vue : le profil VUML"; Thesis, France (2005).

[Parimala and Saini 2011] Parimala, N., Saini, A.: "web Service with Criteria: Extending WSDL"; In Sixth Digital Information Management (ICDIM), Sixth International Conference on Digital Information Management, (September 2011), 205-210.

[Sellami et al. 2010] Sellami, M., Bouchaala, O., Gaaloul, W., Tata, S.: "WSRD: A web Services Registry Description"; 10th. Annual International Conference on New Technologies of Distributed Systems (Notere), (May 2010) 89-96.

[Soukkarieh et al. 2008] Soukkarieh, B., Sedes, F.: "Towards an Adaptive web Information System Based on web Services"; Fourth International Conference on Autonomic and Autonomous Systems, (March 2008), 272-277.