AWS-WSDL: A WSDL Extension to Support Adaptable Web Service

Chiraz EL Hog
MIRACL, ISIMS, Cité El Ons
Route de Tunis Km 10
Sakiet Ezzi, Stfax,
Tunisia
elhog.chiraz@gmail.com

Raoudha Ben Djemaa
MIRACL, ISIMS, Cité El Ons
Route de Tunis Km 10
Sakiet Ezzi, Sfax,
Tunisia
Raoudha.Bendjemaa
@isimsf.rnu.tn

Ikram Amous
MIRACL, ISIMS, Cité El Ons
Route de Tunis Km 10
Sakiet Ezzi 3021, Sfax,
Tunisia
Ikram.Amous@isecs.rnu.tn

ABSTRACT
A Web Service is a software component allowing to expose services via Internet. It insures interactions between heterogeneous applications and systems and is platform independent. The explosion of the Internet users number has led to an important diversity of their profiles. Nevertheless, existing Web Services offer the same result regardless the user profile. Therefore, we are interested in defining a complete adaptation solution that deals with the different steps of the service life cycle. In a previous work we were interested in the design step of the service life cycle and proposed an UML (Unified Modeling Language) extension named AWS-UML (Adaptable Web Service Unifying Modeling Language) that describe the different allowed users profiles at the metamodel level according to the MDA (Model Driven Architecture) principles and standards. As we modified the design of the Web Service and so the resulting implementation code, we need also to extend the service description in order to support the additional adaptation informations. Thus we introduce in the current paper an extension of the standard WSDL (Web Service Description Language) used to describe the service offered functionalities. Our proposal is named AWS-WSDL (Adaptable Web Service Web Service Description Language).

Categories and Subject Descriptors
H.3.5 [Information Storage and Retrieval]: Online Information Services—Web-based services; D.2.2 [Software Engineering]: Design Tools and Techniques—Object-oriented design methods

General Terms
Design

Keywords
AWS-UML, Web Service, AWS-WSDL, adaptation

1. INTRODUCTION
The Internet is evolving from a solely data repository to a collection of complex and heterogeneous services. Web services technology have enabled service provisioning through the Internet. They have become the technology of choice for Internet based applications that guarantees loosely coupled clients and servers. They simplify the development of business applications and improve their interoperability due to their platform-independent nature and to the use of XML-based standards. Mainly, Web Services are structured around three major standards: SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description Discovery and Integration).

The increasing interest on Web Service technology, the growing number of published Web Services and the diversity of users profiles have raised new issues in service use. For instance, a Web Service should be able to deliver an adequate service that fulfills each specific user’s needs and take into consideration his context. Besides, users’ categories are becoming increasingly heterogeneous due to their different rights, individual characteristics, preferences (desired content, layout,...) and interests. A Web Service can also be accessed from different locations, through a diversity of devices (laptops, WebTV, PDA, WAP phone, etc) and network characteristics (Wi-Fi, bandwidth ...). Therefore, adaptable Web Services have emerged as new direction of research, in order to provide the appropriate services to users instead of passively performing user requests and so to improve the customer satisfaction. The adaptation in our case means to adjust the Web Service behavior according to users conditions and situations. It is a major requirement that must be taken into account earlier in the Web Service life cycle essentially on the design and description steps.

UML (Unifying Modeling Language) is considered as the basic visualized modeling language for software systems but its predefined elements are insufficient to design adaptable web services. Thus, we have extended it in [7] by defining new elements: labels, graphics and stereotypes as well as new constraints. Meanwhile, the extension of the modeling language have had a major impact on the description of the Web Service. The standard WSDL, as it is today, is not able to support the description of the extension elements introduced on the design step, So it must be extended. In the
current paper we proposed a possible extension of WSDL and called it AWS-WSDL (Adaptive Web Service Web Service Description Language).

To do so, we proceeded by briefly reviewing our proposed modeling language, AWS-UML in Section 2. In Section 3, we introduced the AWS-WSDL as an extension of WSDL. We described then in section 4 the impact of the adaptation process on the Web Service architecture. Finally, before concluding and giving some directions for future works, we discussed related works.

2. AWS-UML: UML EXTENSION TO MODEL ADAPTABLE WEB SERVICE

Most modeling methods take an object-oriented approach using UML notation. However the use of standard elements of UML, as they are defined by default, is not sufficient to adequately express the aspects of adaptation in Web Services. We proposed a solution to model AWS-UML [7], which complete with UML semantics and constraints for integrating elements and concepts reflecting the user profile. UML is used as a foundation for building domain specific languages by specifying stereotypes. It is used as meta-modeling language, where UML diagrams formalizes the abstract syntax of our modeling language. In this section we will briefly review AWS-UML Use Case diagram and the Class diagram.

2.1 Extended Use Case diagram of AWS-UML

A Use Case diagram is used to describe the dynamic view of the system. It is defined as "a description of a set of sequence of actions that a system performs to yield an observable result of value to an actor" [4] The UML Use Case diagram metamodel, defines one class to model actors and one for Use Cases. This definition does not fit well our need to design Adaptable Web Service. Therefore we propose enriching this metamodel by specifying three kinds of actors that could interact with our Web Service and three kinds of Use Cases that describe functionalities according to the variety of actors.

An actor specifies a role played by a user or any other system that interacts with the subject [4]. It is a behaviored classifier which specifies a role played by an external entity, a human user of the designed system, some other system or hardware using services of the subject. The only criterion is that they must be external to the part of the system being partitioned into Use Cases. However, Web Services can be accessed through different ways and by a variety of roles and profiles: the service supplier, the service human client and application client. To model this distinction, we propose three categories of actors:

- Application Consumer: This actor is used to model a software that interacts with the Web Service. It could be:
  - A Composite Web Service: This actor is used in the case of services composition. It concerns requests of users that cannot be satisfied by any available Web Service, whereas a composite service obtained by combining a set of available Web Services might be used. In that case, a Web Service can play the role of a client to another one.
  - A Web Application: This actor is used to model a web application that uses a Web Service to accomplish its functionalities. Through a servlet, a Web application can connect to a Web Service using it’s URI 1 and access account.
- Human Consumer: This actor is used to model a human requester using a Web Service by a web URL. He is an Internet user how interacts with service from the web (weather service, currency convertor...). He can also, express his preferences and interests to customize service results.
- Provider: This actor is used to model the Web Service provider. He is a person or an organization that supplies services over service registry. He could create, update, deploy or undeploy the Web Service.

The Web Service architecture is based on the interaction between three components: service provider, service registry (broker) for storing service descriptions, and service requester (the client). These interactions are based on publish, find, and bind operations. According to these operations and by the increasing needs of Web Service consumer, their higher mobility and various interests, we distinguish three classes of Use Cases as follows:

- Service Interaction Use Case: Used to model interaction with the Web Service. It includes request sent to a web service, response received from a Web Service, subscription, specify preferences and interests.
- Service Publication Use Case: Used to add or remove Web Service on or from the service registry. It includes create, modify, describe, deploy (publish Web Service on the service registry) and undeploy (retrieve service from the service registry).
- Service Search Use Case: Used to search and select an adequate Web Service.

The UML metamodel does not allow to model actors and Use Cases presented above. So, we present the metamodel of the Use Case diagram of AWS-UML depicted in Figure 1. AWS-UML extension is presented with grey boxes.

2.2 Extended Class Diagram of AWS-UML

Class diagrams are the mainstay of object-oriented analysis and design. They show the classes of the Web Service, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes.

A Web Service may interact with several actors with a variety of roles. Each role has different needs on this service. In order to ensure the adaptability of results, a service must take into account the profile in interaction with it. The problem is how to model these various actors needs when designing class diagram of a Web Service. In order to tackle this problem, we will use the concept of VUML (View based Unified Modeling Language) introduced by Nassar ([10]) and classes modeling user’s profiles when creating class diagram.

1 Uniform Resource Identifier
The concept of VUML revolves around two key concepts: Base and View.

- Base: Is a core entity which includes specifications that are common to all types of actor.
- Views: Are used as a means of assuring functional separation of concern and managing access rights.

In our case of study, 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 context, we should discover all context elements that influence the result. These elements are enclosed on actor’s profiles. The base class permits the representation of the functionalities of the services required by all kinds of users. In contrast, the view class permits the representation of the functionalities required by a specific kind of user. These functionalities are accessible only if the specific user is in interaction with the service. We distinguish one Base and three kind of Views related to users category:

- Application Consumer View: Is an abstract view class to model functionalities allowed to an Application. It depends on the Application Profile. This abstract view contains two views: a Composite Web service View and a Web Application View.
- Provider View: Is a view class to model functionalities allowed to a Web Service provider.
- Human Consumer View: Is a view class to model functionalities allowed to a human consumer. It depends on the Consumer Profile.

Modeling elements of the conceptual class diagram are: the class and the association. Figure 2 shows the metamodel of this class diagram. AWS-UML extensions are presented with grey boxes.

In this section we have presented AWS-UML, a UML profile to model either Web Services functionalities and users profile characteristics according to the UML standard extension mechanisms. These extensions facilitate the design and implementation of Adaptable Web Service.

3. EXTENSION OF THE WSDL STANDARD
Web Service Architecture is based on the interaction between three primary components: the service provider who creates a service description to describe the service functionalities using the WSDL language, the discovery registry in which the service provider should publish his service, such as UDDI and the service consumer who can access to the published service description (WSDL file) and execute his requests or use it to develop new applications. Unfortunately, a WSDL document describes only the functional aspects of a Web Service regardless the adaptation aspect. In order to fully fulfill the needs of users with different profiles we must consider both functional and profile characteristics. So we worked on extending the WSDL standard in order to integrate profile characteristics on the description of the Web Service.

3.1 WSDL overview
WSDL (Web Service Description Language) is an XML-based language used to describe the functionalities offered by the service, needed inputs and produced outputs. It is standardized by the W3C. A WSDL document consists of a set of definitions that describe what a service does, how it is accessed and where it is located. In particular, a WSDL provides an abstract description defining the functional aspect of the Web service (methods, types and parameters) as well as a concrete description presenting technical information that make it possible to connect to the service (format of exchanged messages, protocol of supported transport, access address, etc.). It is defined using six major elements delimited by the <definitions> root element [5]:

- Types: Provides data type definitions used to describe the messages exchanged. Data types are expressed as XML Schema elements, they can be simple or complex.

**PortTypes:** Is the abstract interface definition of the Web Service. It constitutes aggregations of operations provided by the service. It contains operation signature definition (operation name, input parameters, outputs) by referring to message definitions in the Message section.

**Message:** Defines the format of the messages that the Web Service uses. These messages relate to the input, output and fault parameters of the service offered operations.

**Bindings:** Specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.

**Port:** Specifies an address for a binding, thus defining a single communication endpoint enabling the access to a portType through an URI address.

**Services:** A collection of port elements, used to aggregate a set of related ports.

Those definitions are specified and validated by the use of the WSDL XML Schema. WSDL is designed to be extensible, as a matter of fact, description elements and their sub-elements can be extended. Therefore, we can easily extend the description of the operations, the portTypes, the bindings and the offered services.

### 3.2 Extension process

In order to extend properly the WSDL standard, we should transform the abstract model defining the WSDL schema and so interfere at the metamodel level. Actually, a metamodel shows basic structure of a special model, and it is the basis for model extension and transformation. Therefore, our extension process is based on the theory of Model Driven Architecture (MDA) which is Object Management Group’s (OMG) standard for software engineering by model-level design and exploitation. MDA is a software design approach allowing the development of plateform independent software systems based on the separation between process logic and technical plateform. This separation allows to undergo the complexity of information systems. MDA approach let us produce high level abstraction models that can be reused with different implementations. Its architecture is based on four levels of abstractions:

1. **Level M0** which defines concrete entities to model, they represent in our case the WSDL instance files;
2. **Level M1** represents the different software applications models that describe the level M0, it englobes the UML models, les PIM (Plateform Independent Model) and the PSM (Plateform Specific Model). It is an implementation of the level M2;
3. **Level M2** which contains the different metamodels used to define the models of the level M1;
4. **Level M3** which contains the MOF (Meta Object Facility) that is self-defined and used to define the models of the level M2. It is an abstract language for describing, designing and managing intermediate model. As the core element of MDA, MOF will be the key for the extension.

Levels M1, M2 et M3 belong to the word of modeling. According to MDA theory, each model is an instance of a special MOF metamodel which is an instance of a particular MOF meta-metamodel at the same time. XMI (XML Metadata Interchange) provides a set of rules for mapping from models and MOF metamodels into XML documents and XML Schemas.

In our case of study, we are dealing on level M2 and we propose an extension on the WSDL metamodel that we call AWS-WSDL metamodel. The extension is carried out by applying a metamodel transformation depicted in figure 3, that maps the elements of WSDL metamodel to AWS-WSDL one. The transformation is inspired from our AWS-UML Profile and using UML stereotypes so that AWS-WSDL models can take into consideration the diverse profile information. The AWS-WSDL metamodel is then serialized to AWS-WSDL XML schema using XMI rules. The obtained schema is used to produce and validate our model, the AWS-WSDL document. This model can be instantiated later to fit the adequate description of the service.

![Figure 3: Metamodel transformation](image)

**3.3 Metamodel of AWS-WSDL**

In order to elaborate a metamodel that can meet the requirements related to the profile adaptation, we introduce new elements into the Web Service description. Figure 4 illustrates a view of our proposed metamodel. White boxes depicts WSDL standard metamodel and Grey ones extend the WSDL metamodel to include adaptation notion. The complete set of classes and associations in figure 4 (white and grey boxes) identifies the AWS-WSDL metamodel. As said above the extension is inspired from the AWS-UML profile, which provides the ability to model the variety of actors in interaction with the Web Service and their different interests.

PortType is the most important part of WSDL, because it is an XML schema that describes the structure of an XML document.

---

3 An XML schema is a W3C recommendation that describes the structure of an XML document

4 Introduced by the OMG as a common interchange format between UML tools for interchanging models and metadata.
Figure 4: Metamodel of the AWS-WSDL

Figure 4: Metamodel of the AWS-WSDL

describes the operation provided by the service, which is also called the Interface. As the main goal of our extension is to offer to the service client the suitable operation according to the class of user it represents and so to its access rights and profile, we will integrate a new element on the PortType called **UserCategory**. This element permits the definition of the profile interacting with the service and so associates allowed operations in the corresponding view according to his access rights and profile characteristics. This new element is modeled by the **UserCategory** class related to the PortType class by a **is-used-by** association and has an attribute **name** that could take three values: **Provider**, **Human Consumer** or **Application Consumer**. On the other hand, the **UserCategory** has a composition with the **AccessRights** which stores the **views** introduced on section 2.2. It has two attributes: **designation** to mention the name of the view and **isOffered** a boolean value that indicates if the selected right is offered to the current user or no. **UserCategory** class has also an association **described-by** with the **Profile** class. The latest is used to model the profile of the user composed by the **Context** captured by adequate sensors, interpreted and then stored on the WSDL file and by the **Preferences** entered by the user itself using a Web Service user interface. The **Context** is composed by three sub element: **Connection** which describes the network characteristics (protocol, type, bandwidth, delay) **Location** which describes the geolocation of the user (GPS) **Device** which describes the device characteristics using a composition with **Hardware** and **Software** modeling class. The **Hardware** models the type of the device, the screenSize, the memory capacity and the processor rate. The **Software** stores the OperatingSystem types and the used browser.

XMI rules are generated from the proposed metamodel. Figure 5 illustrates an excerpt from the XMI file corresponding to a part from the definition of the **UserCategory** class. Using XMI document, we will be able to generate an AWS-WSDL schema and finally the AWS-WSDL document as an instance of the schema. **Profile** and **Preferences** are stored on the WSDL file using the Comprehensive Structured Context Profiles (CSCP) [13]. It is a representation format that is thoroughly structured and comprehensive to allow for all flavors of context information. It is based on RDF 5 (Resource Description Framework) and thus inherits its interchangeability and extensibility. It does not impose any fixed hierarchy. It rather supports the full flexibility of RDF to express natural structures of context information. Attribute names are interpreted context-sensitively according to their position in the profile structure. Thus, unambiguous attribute naming throughout the profile is not required. Figure 6 illustrates AWS-WSDL document.

4. ADAPTATION ON THE WEB SERVICE ARCHITECTURE

Web services architecture is based on three components: service providers, service consumers and a service registry. Service providers can publish the Web services they offer through a service registry and service consumers can access to the offered Web Service list and select the suitable one that matches their needs.

In our case study, we complete the standard architecture with the proposed adaptation notion presented above. On the design step, the supplier has to use the AWS-UML as a modeling tool. It will be able to model consumer and functionalities offered to each category of users. After the

---

5 is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax formats.
deployment, the Web Service description will be generated according to AWS-WSDL and then published in the UDDI. When a consumer search or select a Web Service, it formalizes the request according to AWS-WSDL. Figure 7 depicts these modifications. Our proposed AWS-WSDL can be used either by the service provider to describe the supported adaptation criteria of the Web Service or by the consumer to express its adaptation specific needs.

5. RELATED WORKS

In this section, we take a look at some research works interested in the possibilities of applying the context adaptation on Web Services design or description steps. We provide an overview of some of these works.

For the design, El Asri et al.[3] propose a model driven approach for the modeling of user-aware web services on the basis of the multi-view component concept. The multi-view 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 profile of the user and his right access to the Web Service functionality. Despite that, the user preferences, device ca-
capacity, network characteristic, localization ... are not taken into account.

Sheng et al. present an UML extension, ContextUML [11], for the model-driven development of context-aware Web services. Although, this work propose a metamodel to model service context, it don’t care about modeling diagrams extension.

In [12, 9], authors describe ContextServ, a platform for rapid development of context-aware Web Services. It adopts model-driven development where context-aware Web services are specified using ContextUML [11]. The platform offers a set of automated tools for generating and deploying executable implementations of context-aware Web services. But, this paper did not specify how to generate the Web Service description and the impact of their extension to the WSDL standard.

Concerning the description of the Web Service, most of the extensions proposed are related to specific demand. In [8], the authors discuss adding semantics to Web Service descriptions. SAWSDL 6 is the first step toward standardizing SWS 7. But it must consider a service ontology and the appropriate domain ontologies to describe Web Services. P-WSDL 8 [1] and Q-WSDL 9 [2] are extensions proposed to specify performance and QOS requirements of Web Services. They give answer only for the QOS need.

All of works previously studied deal separately the design and description step of the Web Service life cycle. However, none of the previous works proposes a complete modeling and descriptive solution. They did not consider the impact of a new modeling method on the generation of the Web Service description. Thus, in a first step our proposal would be useful to design and introduce specific profile element on the main AWS-UML diagrams and after that, to generate corresponding adaptable Web Service description according to AWS-WSDL.

6. CONCLUSION

Web services are the most popular and powerful software development technology in today’s software world. But their large use requires a review of their implementation to be able to answer users specific functional and non functional waiting. Therefore, the adaptability became a challenging issue in the world of Web Services and their life cycle.

In this regard, we have presented in this paper, a solution to model and describe adaptable Web Services. As there are no tools supporting the complete modeling of adaptable Web Services in UML and the WSDL description only addresses the functional aspects of a Web Service without containing any information about the context of use. We have introduced the metamodel and notation of our UML extension as an UML profile, AWS-UML, and the impact on the WSDL standard, AWS-WSDL. Extensions proposed bring answers to reach insufficiencies of the UML language and the standard WSDL to provide consumers with adaptable Web Services.

Currently, our proposal is being validated by designing an Automatic Programming tool that supports new AWS-UML extensions and automatically generates AWS-WSDL description. It is based on the open source tool: ArgoUML. To generate the AWS-WSDL document we are using the java API WSDL4J. In the near future, we aim to extend the UDDI registry to be able to support new Web Services descriptions and we plan to introduce the context acquisition process. Then, we intend to define a platform that contain all extensions proposed in order to obtain an adaptable Web Service from the design step to the interaction with the final client.

7. REFERENCES

A. EXAMPLES OF AWS-UML

To illustrate our proposal, we will give an example of an adaptive Web Service designed with AWS-UML. The example is a travel agency Web Service. To elaborate our diagrams we have defined a set of icons. Table 1 illustrates icons used on the following examples.

<table>
<thead>
<tr>
<th>Icons</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>![App Consumer Actor]</td>
<td>Application Consumer Actor</td>
</tr>
<tr>
<td>![Composite Web Service Actor]</td>
<td>Composite Web Service Actor</td>
</tr>
<tr>
<td>![Web Application Actor]</td>
<td>Web Application Actor</td>
</tr>
<tr>
<td>![Interaction Use Case]</td>
<td>Interaction Use Case</td>
</tr>
<tr>
<td>![Search Use Case]</td>
<td>Search Use Case</td>
</tr>
</tbody>
</table>

### A.1 Example of Use Case diagram of AWS-UML

Figure 8 shows the Use Case diagram of an application consumer. This actor can only have association with Service Interaction Use Case and Service Search Use Case. The web application actor and the composite web service actor can invoke the travel agency service specifying usage agreement. The web application can subscribe with the service. The composite web service can search or compose with other Web Service.

1. User preferences specify the display preferences and the content preferences.
2. The context of use contains:
   (a) The localization defined by latitude and longitude,
   (b) The connection characteristics defined by the bandwidth, type and the debit,
   (c) The device characteristics: software (browser, operating system) and hardware (type, desktop, processor, memory, graphicsCard).

The Application View is related to an application profile defined by the context, an access account and an usage agreement between the web service and the application:

1. Access account defined by login, password and date of validity of the account,
2. Usage agreement defined by the list of input parameters, the list of output parameters and the access URL.

These actor profiles will be saved in the WSDL file used when searching or invoking the Web Service.

Figure 9 shows the example.

### A.2 Example of Class diagram of AWS-UML

In this section, we will design a class diagram of a travel agency service using AWS-UML. The class stereotyped Base represents the common behavior. Classes stereotyped View represent specific behavior. Views are related to profiles. The Human Consumer View is related to a consumer profile defined by user preferences and the context of use.