

WSDL Automatic Generation from UML Models in a MDA Framework

Juan M. Vara, Valeria De Castro and Esperanza Marcos

Escuela Superior de Ciencias Experimentales y Tecnología,
Departamento de Lenguajes y Sistemas Informáticos,
Universidad Rey Juan Carlos,
28933 Madrid, Spain

juanmanuel.vara@urjc.es, valeria.decastro@urjc.es, esperanza.marcos@urjc.e

Abstract: Nowadays the growing significance and the utility of modeling in the development of any kind of software is clearly illustrated by the thriving Model Driven Architecture (MDA). MDA proposes the definition and use of models, with different levels of abstraction, all along the software development process. Together with the popularity of MDA, we must consider the increasingly importance of Web services as the perfect framework to integrate systems and services. Since Web Services are software components, it seems proper to apply modeling techniques in the Web Services development process. From this point of view MDA serves as the ideal framework to include the modeling process in Web Services development. The application of MDA techniques for Web Services development needs of new tools supporting the modeling of Web Services. In this work we present the MIDAS-CASE subsystem for Web Services. MIDAS-CASE is a MDA tool for Web Information Systems development. Using an extension of the Unified Modeling Language (UML) for the Web Services Description Language (WSDL), the MIDAS-CASE subsystem proposed supports the modeling of Web Services in extended UML and the automatic generation of the respective WSDL description for the Web Service modeled.

Keywords: Model Driven Architecture, Web Services modeling, Automatic code generation.

I. Introduction

The Web has become the tool par excellence to share information between different organizations. In this context, Web Services have emerged as the perfect framework for application-to-application interactions, making these applications available as Web Services. To standardize the use of Web Services, the World Wide Web Consortium (W3C) proposed the Web Service Description Language (WSDL) standard [25], [26] an XML based language that describes Web Service functionality. Essentially, a WSDL file is a language independent XML-based version of an Interface Definition Language (IDL) file that describes the operations offered by a Web Service, as well as the parameters that these operations accepts and returns.

So, WSDL is the standard that supports the description of Web Services: what they do, how they should be used and where they are localized.

Together with the boom of Web Services, we must consider the growing significance and utility of modeling in the development of any kind of software, clearly illustrated by the thriving Model Driven Architecture (MDA) [11], [16]. MDA is a model-driven framework for software

development that proposes to model the business logic with Platform Independent Models (PIMs), to later transform them on Platform Specific Models (PSMs), using transformation guides between the different models. The main advantage of MDA is the ability to transform one PIM into several PSMs, one for each platform or technology in which the final system will be deployed, and the automatic code generation that implements the system for those platforms from the corresponding PSMs.

Since Web Services are software components, the development of Web Services is suitable to exploit the advantages of MDA. To apply the MDA principles on Web Services development, a modeling process must be introduced. According to MDA principles, this modeling activity should result in code automatic generation. If we want to abstract from the platform in which the Web Service will be finally deployed, the code that should be generated is the WSDL document containing the Web Service description in a standard format. This way, anyone could make use of the Web Service, attending to its WSDL description.

Obtaining the code that implements the Web Service for specific platforms is trivial having the WSDL description of the Web Service. In fact, several tools exist that include this capability [1], [7]. On the other hand there are no tools supporting the complete modeling of Web Services with the in UML.

In this work, we propose an UML extension for WSDL [14] and we introduce the MIDAS-CASE subsystem for Web Services [22]. The Web Services subsystem uses the UML language extension for WSDL proposed to support the modeling of Web Services in extended UML and the automatic generation of the respective Web Service description in the WSDL standard.

Both, the UML extension for WSDL and the CASE tool that includes the subsystem introduced are part of MIDAS [4], a model-driven methodological framework for the agile development of Web Information Systems (WIS) that proposes to use standards all along the development process.

The rest of this article is structured as follows: section II is an overview of the existing tools for Web Service development and its relation with MDA principles; section III gives a brief overview of MIDAS framework; in section IV we present the WSDL metamodel that guides the UML extension for WSDL proposed; section V describes the UML extension for WSDL and its application by means of a case-study is showed in section VI; the MIDAS-CASE subsystem

for Web Services is presented in section VII. Finally, in section VIII we conclude underlying the main contributions and future work.

II. Previous Works

In this section we give a brief overview of the existing tools for Web Services development, considering their capabilities in relation with the MDA principles [11], [16]:

A. Web Services development kits

The first group is comprised by tools or applications focused on generating the Web Service implementation: the Web Services development kits (web Methods Glue [24], Systinet [19]) support an inverse development process, considering the traditional ones. The Web Service description is built starting from its implementation; likewise there are several tools (Borland Web Services Solution [3], IBM WebSphere Studio Application [7]) capable of generate automatically the WSDL description starting from the Java Beans implementation or C++ source code of the Web Service, or even SQL queries or stored procedures. They can also perform the opposite process: to generate the Web Service implementation from its WSDL description. This set of resources does not include any modeling capability, so, we can rule out the tools and applications included in this group, according to the objectives fixed in this work: introduce the modeling activity in the Web Service Development.

B. Tools importing UML models

The second group includes tools, such as Codagen Architect [5] or the IMS General Web Services UML to WSDL Binding Auto-generation Guidelines [8] that supports the generation of the Web Service description or its implementation, starting from UML models imported from other tools. So, the tools included in this group can be used as part of the development process, but they do not support the complete process since they need other tools to perform the modeling activity.

C. Tools supporting Web Service modeling

The last set of tools reviewed comprises those supporting Web Service modeling, but either they use their own notations (BEA WebLogic Workshop [2], XML Spy [1]) or they only support UML modeling at conceptual level (Oracle JDeveloper 10g [18]). Obviously, we are not interested in the former subgroup since they do not use standard notation to model the Web Service. The tools on the latter subgroup generate the Web Service implementation or the WSDL description starting from UML class diagrams. These diagrams, due to their abstraction level, will not be capable of collecting the complete semantics that allows the correct and complete definition of the Web Service. This last approach ignores one of the most important MDA principles: the code is automatically generated from the PIM model, skipping the PSM model.

As a conclusion, we can say that the aim of all the tools or applications reviewed is to obtain the code that deploys the Web Service in a specific technology or platform. Working this way, the whole development process depends on the technology in which the Web Service will be finally

deployed and this assertion is clearly against the MDA principles, that claims that the initial modelling activity should be totally independent from the final platform/s in which the system will be deployed.

Probably the best of the reviewed tools for Web service (MDA based) development is Iopsis iNstight [9], a JAVA components suite that extends the Sun One Studio IDE [20] to support the Web Service modelling with the UML language. Once again, considering the principles of MDA, this tool fails in the lack of conceptual modelling support. In fact, a stereotyped UML model is used as starting point for Web Service development. Since it is a stereotyped model, it is already considering a specific platform: the Web Services platform. So, according to MDA, as well as the model is defined for a concrete scope, this model is a PSM. Therefore, the initial model on the MDA approach, the PIM model, is not incorporated in Iopsis Insight.

It is our opinion that, to apply MDA to Web Service development and to obtain a complete and correct development process, it would be preferable the definition of an initial PIM. Next, we would apply the subsequent transformations on the PIM to obtain the UML stereotyped model (or PSM) that will include only concepts on the Web Service scope. Finally, starting from this PSM, the code that implements the Web Service or its WSDL description will be automatically generated.

In this sense, our proposal, the MIDAS-CASE subsystem for Web Services goes beyond the existing proposals by supporting the complete software development process for Web Services, according to the MDA principles.

The capabilities derived from integrating the subsystem introduced in this work in a complete MDA tool will supply the framework in which this complete development process, from conceptual modelling to code generation (the WSDL description) will be carried out.

III. MIDAS Framework

MIDAS is a methodological framework for the agile development of WIS, which proposes a Model Driven Architecture based on the MDA approach. MIDAS proposes to model the WIS according to two orthogonal dimensions (see Figure 1). On the one hand, taking into account the platform dependence degree (based on the MDA approach), two group of activities are considered: to specify the whole system by Computation Independent Models (CIMs), Platform Independent Models (PIMs) and Platform Specific Models (PSMs); and to specify the mapping rules between these models. On the other hand, MIDAS considers the modeling of the WIS according to three basic aspects [8]: hypertext, content and behavior. Besides, MIDAS suggests using the Unified Model Language (UML) as unique notation to model both PIMs and PSMs.

The MIDAS framework has been partially presented in [4], [14], [15], [23]. In this work we focus on the PSM models involved in the behavior aspect modeling of a WIS. Specifically, our proposal solves the problem of modeling Web Services using standard notation (extended UML) by means of the UML extension proposed and the tool that we are building to support the whole methodology.

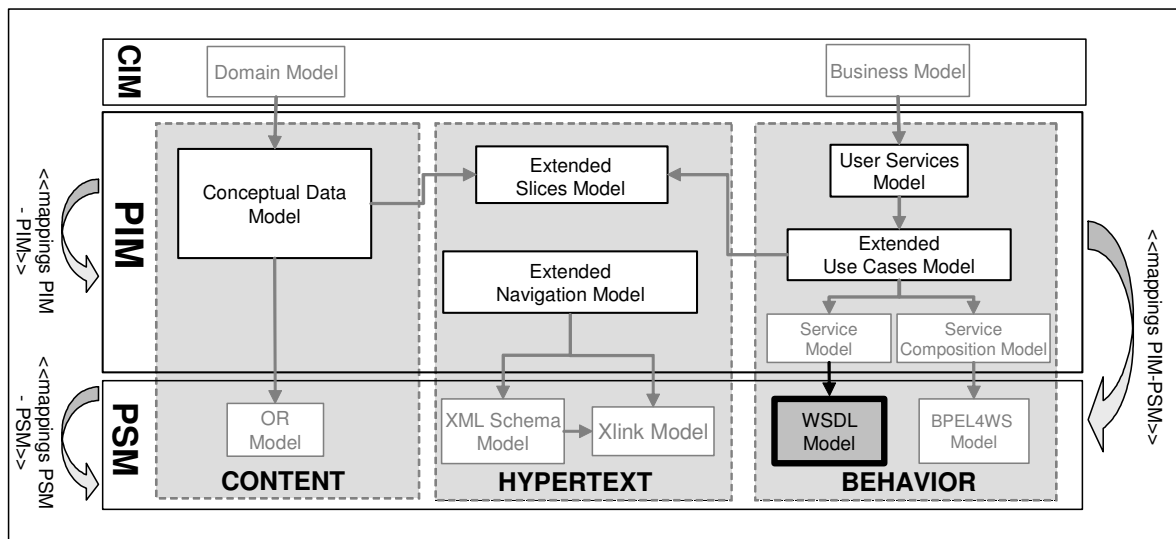


Figure 1. The Model Driven Architecture of MIDAS. We have pointed out the model in which this work is focused: the WSDL model, included in the modeling of the behavioral aspect.

IV. WSDL Metamodel

The Web Services Description Language (WSDL) [25], [26] is the language proposed by the World Wide Consortium (W3C) for Web Services description. A WSDL document is an XML document which specifies the operations that a service can perform. One of the advantages of WSDL is that permits it to separate the abstract description of the functionality offered by a Service from the description of the concrete details, such as the message format or the communication protocol that could be SOAP, HTTP or MIME.

WSDL describes the Web Services through the messages that are exchanged between the Service provider and the Service requester. These exchanged messages are described as an operation. A collection of operations is called a port type, which define the Service interface in an abstract way. The binding between a port type and a concrete network protocol and a message format defines the service interface in a concrete way. A Service defines one or more port. A port indicates the concrete interface localization.

Figure 2 shows the WSDL metamodel represented by an UML class diagram. The definitions included on the WSDL Standard can be separated into two different groups of components, according with the abstraction level of the respective concept represented by each class. The shadowed components represent the concrete issues of the Service description while the rest represent the abstract issues of this description.

A WSDL document contains a version number and a root DEFINITION component. It has a Name and a TargetNameSpace attributes and zero or more namespaces. The namespaces are used to avoid naming conflicts when several services or applications are integrated. A DEFINITION component contains: a TYPES component and zero or more MESSAGE, PORTTYPE, BINDING and SERVICE components. All WSDL components can be associated with a DOCUMENTATION component.

A TYPES component is used for data type definitions

which will be used in messages. For this purpose WSDL is based on XML Schema [27] and contains a SCHEMA component in which, namespaces and data types are defined. WSDL allows including XML Schemas documents previously defined, using a INCLUDE component indicating the document location. In the same way the IMPORT component is used to reuse WSDL documents, in this case the document name and the location are needed.

The PORTTYPE component is the most important component in WSDL, since it describes the operations that the service realizes, that is, the interface. The OPERATION component groups the set of messages that will be interchanged between the service provider and the requester. Each operation can be associated with one, two or three messages, that is, one input message, one output message or both, and optionally a fault message. A MESSAGE contains a Name attribute and zero or more PART components. The PART component describes one portion of a particular message that a Web service sends or receives. The type associated to a PART can be a XSD base type (int, float, string, etc.) or a type defined in the TYPES section. In this case, the data type can be defined by means of a type or element attribute.

A BINDING component describes the binding of a PORTTYPE component and the associated operations to a concrete message format and communication protocol, such as SOAP, HTTP or MIME [26]. WSDL defines different components to describe each one of these protocols. However a detailed discussion on message format and communication protocol is beyond the scope of the present work and will be boarded in future works.

A SERVICE component contains a Name attribute and describes the set of PORTs that a service provides. A PORT component contains a Name attribute.

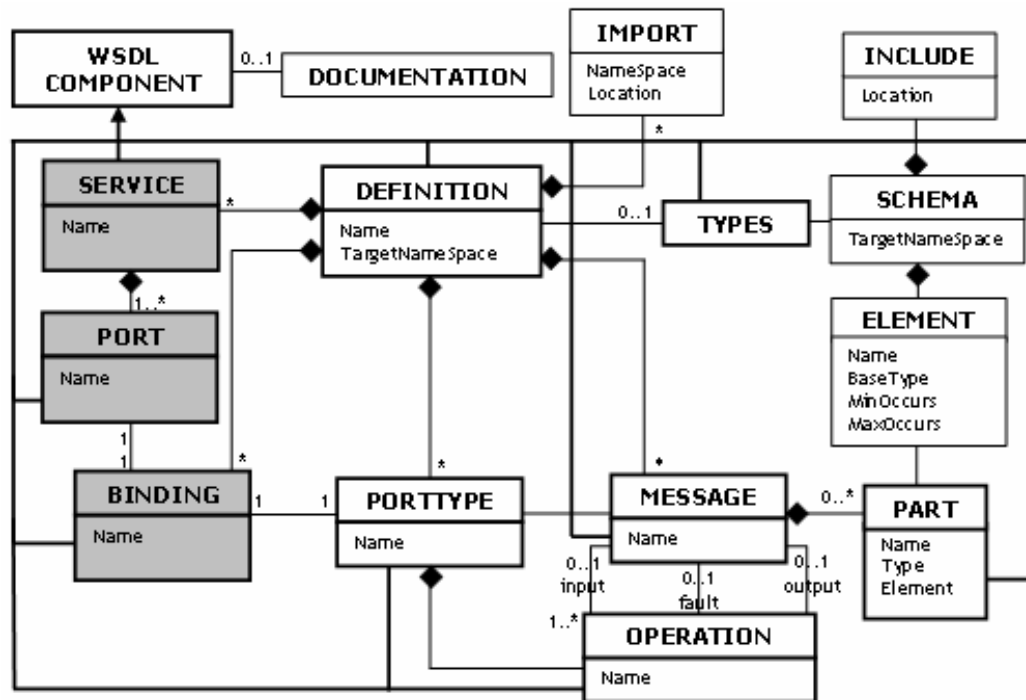


Figure 2. WSDL metamodel represented in UML

It is related with the BINDING component that describes how and where (by the location attribute) to interact with the service interface.

In short, we can say that:

- At conceptual level, the operations offered by a Web service group together forming INTERFACES. Every time an OPERATION is used, the requester interchanges a set of MESSAGEs with the service provider. At the same time each message can contain several PARTs or parameters, whose type can be a XSD base type [27] (int, float, string, etc.), or any one of the types defined in the types section. In the last case, the data type can be defined by means of a TYPE or a ELEMENT attribute, from one of the SCHEMAs referenced in the WSDL document.
- A Web service could be defined with the components already mentioned, despite of the platform or the language used for implementing it. The BINDING component allows to bind the conceptual Web service definition with the implementation in a specific platform. Each implementation will offer access points (END POINT) to the whole SERVICE.

V. Representing WSDL with UML

As previously mentioned, MIDAS proposes to use standard notation all along the development process. Since UML does not support the modeling of the concepts that conforms the previous metamodel, we have to extend it using its own extension mechanisms, to make possible the representation of the components included in the preceding metamodel.

UML has been designed to be extended in a controllable way. This mechanism enables us to create new types of building blocks by means of stereotypes, tagged values and constraints. According to [6] a UML extension should contain: a brief description of the extension; the list and

description of the stereotypes, tagged values and constraints; and a set of well-formedness rules that are used to determine whether a model is semantically consistent. For each stereotype we have to specify the common properties and semantics that go beyond the basic element being stereotyped by defining a set of tagged values and constraints for the stereotype [17].

Before showing the proposed UML extension, we will explain the design guidelines that have been defined for the representation of WSDL in extended UML.

A. Design guidelines for the UML extension

To choose the necessary stereotypes to represent in UML all of the WSDL metamodel components and the relationships between them, the following design guidelines are defined:

- DEFINITION component has been considered as a stereotyped class because it is explicitly defined in WSDL and it constitutes the root component that groups all the used elements.
- TYPES and SCHEMA components have been considered compositions stereotyped with <<TypeSchema>> and represent the relation between a DEFINITION component and the data type definitions.
- MESSAGE, PART, PORT TYPE, OPERATION, BINDING, PORT, SERVICE and IMPORT components have been considered stereotyped classes because they represent important components and they are explicitly defined in WSDL.
- Each MESSAGE component will be related to the PART component that it uses by means of a composition.
- The relationship between a PART component and an ELEMENT component will be represented by means of: an association stereotyped with <<Part_Type>> if the PART component uses the ELEMENT as a type; an association stereotyped with <<Part_Element>> if the PART component uses the ELEMENT as an element.

- The relation between an OPERATION component and a MESSAGE component will be represented by means of an association stereotyped with <<Input>>, <<Output>> or <<Fault>>, depending on the type of the message: an input message, an output message or a fault message.
- MESSAGE, PORTTYPE, BINDING, SERVICE and IMPORT components will be related to the DEFINITION component by means of a composition.
- PORTTYPE component will be related to the OPERATION component that it uses by means of an aggregation.
- BINDING component will be associated to the PORT TYPE component that it describes.
- SERVICE component will be related to the PORT components that it provides by means of a composition.
- The PORT component will be associated to the BINDING component that it uses.
- As we have already said, WSDL uses XML Schema for the definition of the data types that will be used for message sending. For this reason we use the UML extensions to represent XML Schemas proposed in [23].

B. The UML extension

The UML extension collected in Table 1 defines a set of stereotypes, tagged values and constraints that enable us to represent WSDL in graphical notation using UML. The UML extension is defined for the WSDL components included in the W3C proposal. Each WSDL component should be able to be represented in graphical notation with this UML extension.

Component	UML Metamodel Component	Description	Constraints, Attributes & Tagged Values
DEFINITION	CLASS	A <<DEFINITION>> class represents a DEFINITION component of the WSDL metamodel.	ATTRIBUTE: <i>TargetNameSpace</i> is an URI (Uniform Resource Identifier). It is mandatory and identifies the namespace which it will belong all of the component names.
			TAGGED VALUES: Other namespaces that will be used in the Web service description.
ELEMENT	CLASS	An <<ELEMENT>> class represents an element of the XML Schema.	It must be related to the <<DEFINITION>> class by means of a composition stereotyped with <<Type_Schema>>.
			TAGGED VALUES: The name of the element, the base type and the minimum and maximum number of occurrences.
Type_Schema	COMPOSITION	A <<Type_Schema>> composition represents a relationship between the DEFINITION component of the WSDL metamodel and the data types defined by means of element of the XML Schema.	It can only be used to join a <<DEFINITION>> class with the <<ELEMENT>> class that uses it.
			TAGGED VALUES: The namespace defined for SCHEMA component.
MESSAGE	CLASS	A <<MESSAGE>> class represents a MESSAGE component of the WSDL metamodel.	It must be related to the <<DEFINITION>> class by means of a composition and must be associated to at least one <<PART>> class.
PART	CLASS	A <<PART>> class represents a PART component of the WSDL metamodel.	It must be related to one <<MESSAGE>> class by means of a composition.
			ATTRIBUTE: <i>Type</i> is a base type XSD. It is optionally and must be defined when the PART component uses a base type but not when the PART component uses an element of the XML Schema.
Part_Type and Part_Element	ASSOCIATION	A <<Part_Type>> or <<Part_Element>> association represents a relationship between a PART component of the WSDL metamodel and an element of the XML Schema.	A <<Part_Type>> association can only be used to join a <<PART>> class with an <<ELEMENT>> class when a PART component uses the element as a type. A <<Part_Element>> association can only be used to join a <<PART>> class with an <<ELEMENT>> class when a PART component uses the element as a element.
PORTTYPE	CLASS	A <<PORTTYPE>> class represents a PORTTYPE component of the WSDL metamodel.	It must be related to the <<DEFINITION>> class by means of a composition and must be associated at less one <<OPERATION>> class.

Component	UML Metamodel Component	Description	Constraints, Attributes & Tagged Values
OPERATION	CLASS	An <<OPERATION>> class represents an OPERATION component of the WSDL metamodel	It must be related to the <<PORTTYPE>> class by means of an aggregation.
Input, Output and Fault	ASSOCIATION	An <<Input>>, <<Output>> or <<Fault>> association represents a relationship between an OPERATION component and MESSAGE component of the WSDL metamodel.	A <<Input>> association can only be used to join an <<OPERATION>> class with a <<MESSAGE>> class when the message is an input message. A <<Output>> association can only be used to join an <<OPERATION>> class with a <<MESSAGE>> class when the message is an output message. A <<Fault>> association can only be used to join an <<OPERATION>> class with a <<MESSAGE>> class when the message is a fault message.
BINDING	CLASS	A <<BINDING>> class represents a BINDING component of the WSDL metamodel.	It must be related to the <<DEFINITION>> class by means of a composition and it must be associated to only one <<PORTTYPE>> class.
SERVICE	CLASS	A <<SERVICE>> class represents a SERVICE component of the WSDL metamodel.	It must be related to the <<DEFINITION>> class by means of a composition and it must be composed by at least one <<PORT>> class.
PORT	CLASS	A <<PORT>> class represents a PORT component of the WSDL metamodel.	It must be related to the <<DEFINITION>> class by means of a composition and must be associated to only one <<BINDING>> class.
			ATTRIBUTE: <i>Location</i> is an URL (Uniform Resource Locator). It is mandatory and identifies the access point to the service.
			ATTRIBUTES: <i>Namespace</i> is an URI (Uniform Resource Identifier). It is mandatory and indicates that the containing WSDL document can contain references to the WSDL definitions in that namespace. <i>Location</i> is an URI (Uniform Resource Identifier). It is optional and indicates the location of some information for the namespace.
IMPORT	CLASS	An <<IMPORT>> class represents an IMPORT component of the WSDL metamodel.	It must be related to the <<DEFINITION>> class by means of a composition.
			ATTRIBUTES: <i>Namespace</i> is an URI (Uniform Resource Identifier). It is mandatory and indicates that the containing WSDL document can contain references to the WSDL definitions in that namespace. <i>Location</i> is an URI (Uniform Resource Identifier). It is optional and indicates the location of some information for the namespace

Table 1. UML extension for WSDL

VI. A Case Study

In this section we show, through a case study, the use of the UML extension proposed for the representation of WSDL. The case study consists of a service for validating an email address called "ValidateEmail". Fig 3 shows the "ValidateEmail" Web Service description in WSDL.

The Web Service defines one operation, "ValidateEmailAddress" which has two messages, an input and an output message. The input message, "ValidateEmailAddressSoapIn" defines one part, "ParametersIn", which uses the element "ValidateEmail Address" as a data type. The output message "ValidateEmail AddressSoapOut" also defines one part, "ParametersOut" which uses the element "ValidateEmailResponse" as a data

type. Both, the "ValidateEmailAddress" and the "ValidateEmail Response" elements have been defined in the section types. The porttype "EmailServicePortType" groups the operations that will be performed by the service, in this case there is only one operation, "ValidateEmailAddress". The link between this port type and the SOAP protocol is described by the "EmailServiceBinding" element. The service has only one port "EmailServiceSoap", which defines through an URL the Web service location.

Figure 4 shows the UML representation of the "ValidateEmail" Web Service using the defined UML extension. The *Name* attribute of the DEFINITION component will be the name of the class and the *TargetNameSpace* attribute will be represented as a class attribute. The used namespaces will be represented as tagged values. For the sake of clarity a tagged value will be

represented as a note associated to the element that uses it.

The relationship between the **<<DEFINITION>>** class and the data types defined, “ValidateEmailAddress” and “ValidateEmailResponse” is represented by the **<<TypesSchema>>** composition. The *TargetNameSpace* attributes of the SCHEMA component will be represented as

tagged values.

When referring to the messages components and the part components they use we can show that the “ValidateEmailAddressSoapIn” message contains one part, which data type is specified by the association with the “ValidateEmailAddress” element as a data type.

```

<?xml version="1.0" encoding="utf-8" ?>
<definitions name="ValidateEmail"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:s0="http://soap.einsteinware.com/Email"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  targetNamespace="http://soap.einsteinware.com/Email"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace=http://soap.einsteinware.com/Email/EmailDataSet.xsd local
  "http://soap.einsteinware.com/email/emailservices.asmx?schema=EmailDataSet"

  <types>
  <s:schema targetNamespace="http://soap.einsteinware.com/Email">
    <s:element name="ValidateEmailAddress">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
            name="emailAddress" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ValidateEmailAddressResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1"
            name="ValidateEmailAddressResult"
            type="s0:CheckEmailResult" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:simpleType name="CheckEmailResult">
      <s:restriction base="s:string">
        <s:enumeration value="Valid" />
        <s:enumeration value="InvalidUser" />
        <s:enumeration value="InvalidAddress" />
        <s:enumeration value="InvalidServer" />
        <s:enumeration value="Error" />
      </s:restriction>
    </s:simpleType>
  </s:schema>
</types>

  <message name="ValidateEmailAddressSoapIn">
    <part name="parametersIn" element="s0:ValidateEmailAddress" />
  </message>

  <message name="ValidateEmailAddressSoapOut">
    <part name="parametersOut" element="s0:ValidateEmailAddressResponse" />
  </message>

  <portType name="EmailServicesPortType">
    <operation name="ValidateEmailAddress">
      <input message="s0:ValidateEmailAddressSoapIn" />
      <output message="s0:ValidateEmailAddressSoapOut" />
    </operation>
  </portType>

  <binding name="EmailServicesBinding" type="s0:EmailServicesPortType">
    <soap:binding transport=http://schemas.xmlsoap.org/soap/http style="document" />
    <operation name="ValidateEmailAddress">
      <soap:operation soapAction=
        "http://soap.einsteinware.com/Email/ValidateEmailAddress"
        style="document" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>

  <service name="EmailServices">
    <port name="EmailServicesSoap" binding="s0:EmailServicesBinding">
      <soap:address location=
        "http://soap.einsteinware.com/email/emailservices.asmx" />
    </port>
  </service>
</definitions>
  
```

Figure 3. WSDL description of a “ValidateEmail” Web Service

In the WSDL document, this association is collected by means of the element attribute of the respective part element. Therefore, the existing association between “ParametersIn” part and “ValidateEmailAddress” element is stereotyped

with **<<Part_Element>>**. In the same way, the existing association between “ParametersOut” part and “ValidateEmailResponse” element is stereotyped with **<<Part_Element>>**.

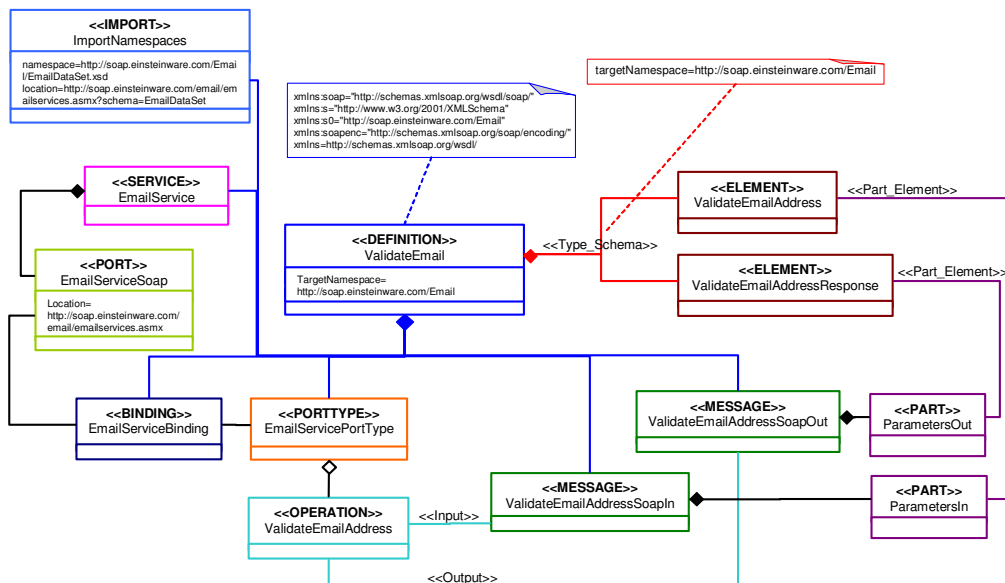


Figure 4. “ValidateEmail” Web service represented in UML extended

Now, let focus on the PORTTYPE and the OPERATION components. The “*EmailServicePortType*” uses one operation, “*ValidateEmailAddress*”. Therefore an aggregation is represented between them.

This operation defines two messages, the input message “*ValidateEmailAddressSoapIn*”, related with the operation “*ValidateEmailAddress*” by means of an <<Input>> association and the “*ValidateEmailAddressSoapOut*” output message, related to the “*ValidateEmailAddress*” operation by means of an <<Output>> association.

The extended UML model also represents the BINDING component, without representing the connection with the SOAP protocol. The “*EmailServiceBinding*” component describes the binding to the porttype; therefore an association between the “*EmailServiceBinding*” binding and the “*EmailServicePortType*” porttype is represented.

Finally, the representation of the SERVICE and PORT components is also collected in the extended UML model. The “*EmailService*” service contains one port, “*EmailServiceSoap*”,

therefore the association between them is represented by means of a composition. The *Location* attribute indicates the service URL and is represented like a class attribute.

VII. WSDL Automatic Generation

In this section we present the MIDAS-CASE Web Services subsystem that makes use of the UML extension described in the last section to support the modeling of Web Services in extended UML and the automatic generation of the WSDL description of the Web Service modeled.

The MIDAS-CASE Web Services subsystem architecture, showed in Figure 5 is a n-tier architecture with three tiers: the graphic user interface, the application logic and the persistence layer.

The architecture definition is guided by the common idea followed by the vast majority of existing CASE tools, and more specifically by all the subsystems that compose MIDAS-CASE: the user defines a (extended) UML model and that model is stored in some XML format to support its later reconstruction.

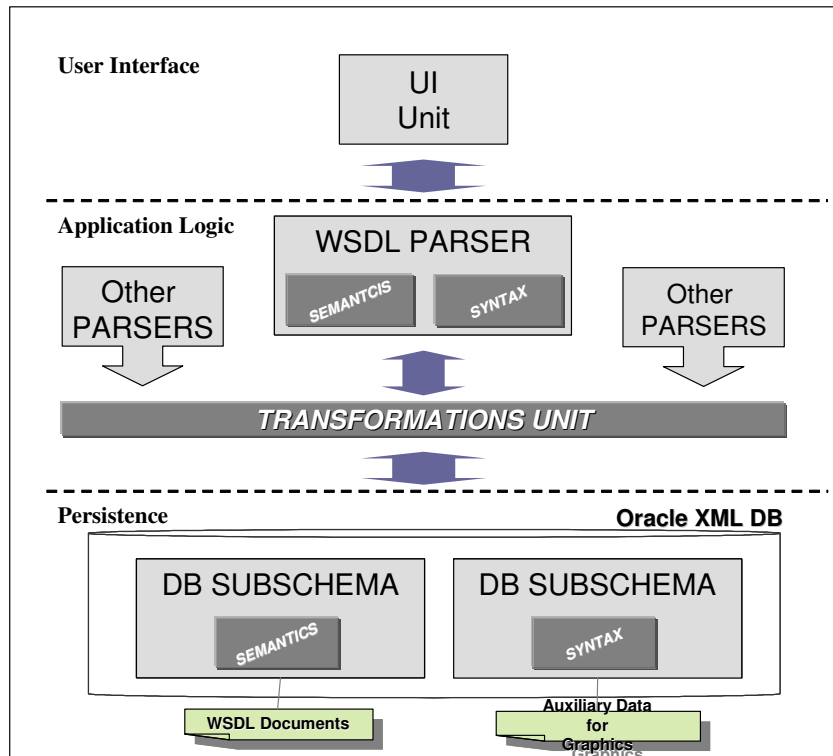


Figure 5. The MIDAS-CASE Web Services subsystem architecture

To specify this XML format, we have defined an XML Schema that describes the meta-model of the UML extension for WSDL presented in section V. The XML document in which the model is stored is an instance of the XML Schema defined. This way, if the XML document is not valid according to the XML Schema, we can conclude that the model is not valid, since the model does not carry out the metamodel.

Now, we summarize the characteristics of each layer in the MIDAS-CASE Web Services architecture, showing its functionality with the case study already presented: the Web Service for validate Email addresses. Moreover, we will specify some implementation details.

A. User Interface Layer

In the user interface layer we found the UI (User Interface) unit, which includes the software components that support the management of the graphic elements that can be included in the extended UML model, corresponding to the concepts included in the respective meta-model [14].

Figure 6 shows the MIDAS-CASE Web Services subsystem interface. The extended UML model for the Web Service used as case study is displayed in the working area.

B. Application Logic Layer

The application logic layer comprises two levels: in the

upper level we find the parser unit. This unit generates two XML documents to store separately the semantics and the syntax of the model that is being defined.

By dividing the semantics of the model from its syntax we can separate the significant data, the different classes (corresponding to instances of the WSDL metamodel) and their associations, from the extra data, such as the screen

positions of the different boxes that represent each class, as well as their sizes or colors. The structure of these two XML documents is defined by two XML Schemas. As mentioned previously, the definition of the XML Schema that determines the structure of the XML documents that contain the semantics data, forces the respective XML document to be a valid WSDL document.

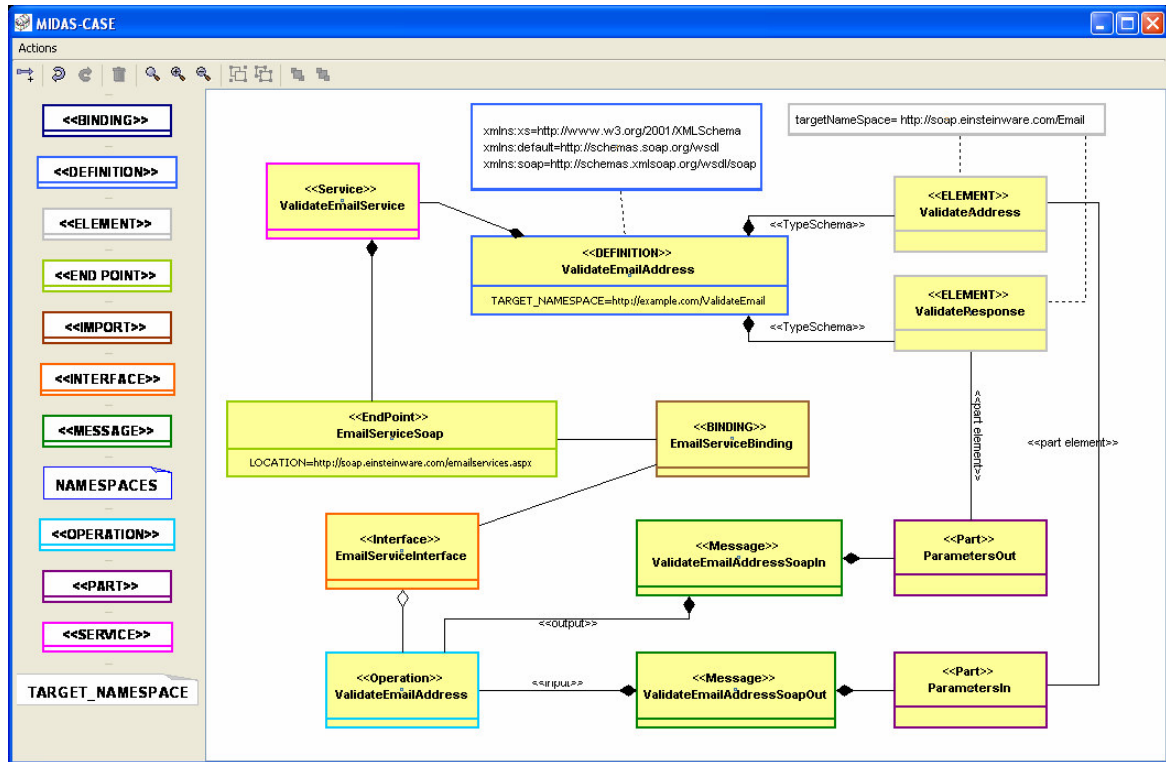


Figure 6. Interface of MIDAS-CASE subsystem for Web Services

The lower level is a common module that supports automatic model transformation. In the scope of the Web Services subsystem of MIDAS-CASE, this module will allow us to obtain the WSDL model for a Web Service starting from a pure conceptual model.

Continuing with the case study, Figure 7 focuses on part of the Web Service used: the messages used by the operation offered by the Web Service as well as the parameters (parts) used by each one of these messages.

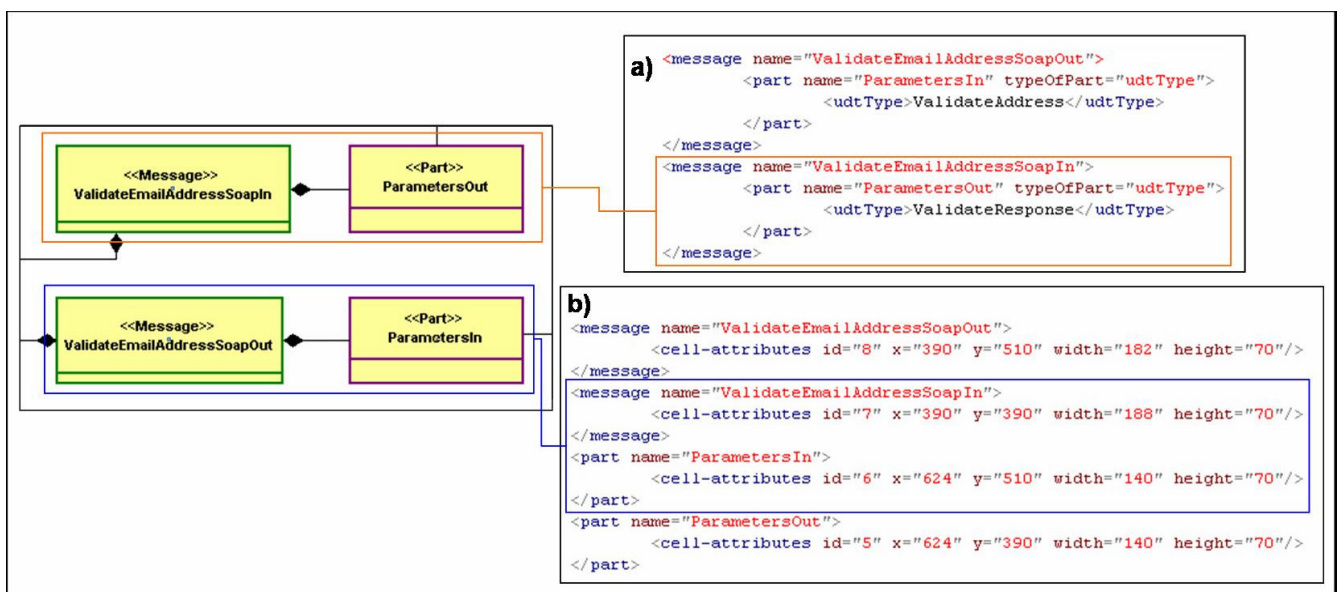


Figure.7. Part of the Web Service “ValidateEmailAddress” and the XML documents containing its syntax (a) and semantics (b)

On the one hand, Figure 7(a) shows the fragment of the XML document containing the syntax for this part of the model. As can be showed, the XML fragment contains only dimension and shape data referring to the graphic components included on this part of the model. So, there is an XML element to store the information of the graphic element that represents the input message, another one for the output message, etc.

On the other hand Figure 7(b) shows the XML fragment of the document containing the semantics of the same part of the model. The semantics of a Web Service is compiled in a WSDL document. So, the data in the XML fragment is the WSDL information that describes the messages used by the operation provided on the Web Service and the parameters used by these messages.

C. Persistence Layer

We have used Oracle XML DB [21] to build the models repository and supply the models defined with MIDAS-CASE with persistence features. The XML DB main feature is that it automatically generates database schemas from XML Schemas. The MIDAS-CASE subsystem for Web Services exploits this feature to obtain two different database schemas: one for storing the semantics of the models (the WSDL documents), defined from the XML Schema that describes the meta-model of the UML extension for WSDL, and the other one for storing the syntax of the same models, defined by the XML Schema that describes the structure of the XML documents that collects the syntax of the models.

Additionally, XML DB supplies automatic validation of models. When we store a new model in the repository, XML DB validates the XML document that contains the semantics model, against the respective XML Schema.

This way, MIDAS-CASE acts also as a model validator: if the document which stores the semantics of the modelled Web Service (that is to say, the WSDL document that describes the Web Service) does not fulfill the respective XML Schema (the one defining the WSDL standard), the model is not valid and the user is advised on this matter.

D. Technical Details

The whole MIDAS-CASE subsystem for Web Services has been implemented with JAVA.

To develop the interface layer we have used an open source graph component available for JAVA named: JGraph. This Swing component includes a powerful API that simplifies the tedious task of drawing diagrams.

For the application layer interface, we have coded some JAVA packages that use the SAX (Simple API for XML) API for JAVA to parse and unparsed the XML documents that describe the different models.

Finally, the persistence layer, apart from the Oracle XML DB database is coded in a JAVA package that allows inserting and querying XML data on the database. To code this package we have used the Java DOM API for XMLType.

VIII. Conclusions

In this work we have presented the MIDAS-CASE subsystem for Web Services along with the UML extension

for representing WSDL that it uses. This tool supports the modeling of Web Services with extended UML and the automatic generation of the WSDL description for the Web Service modeled.

For the definition of the UML extension, firstly we have described each of the components of the WSDL metamodel using UML and then we have presented the design guidelines for the extension definition. To validate the proposed extension we have developed different cases study. In this work we have showed a Web Service for validating an email address represented with the defined UML extension.

To introduce the MIDAS-CASE subsystem for Web Services, we started by giving an overview of the whole architecture that have guided its development and later we have summarize the characteristics of each layer using the case study already used to present the UML extension for WSDL.

The subsystem introduced, together with the whole MDA tool in which it is integrated, supports a complete development process for Web Services, according to the principles of MDA.

Besides, the Web Services subsystem allows us to validate some of the proposals of MIDAS, the methodological framework in which MIDAS-CASE is being developed.

The Web Services subsystem shows how the UML extension for WSDL proposed in MIDAS can be effectively used for the UML modeling of Web Services. Moreover, starting from the Web Service UML extended model, the WSDL description is automatically generated, showing that the extension proposed is correct and complete and it supports the correct and complete definition of Web Services models.

Currently we are planning to obtain the necessary extensions for the complete description of the Web Service, including the connection to specific protocols (SOAP, HTTP and MIME) as well as the integration of techniques for Web Services composition in MIDAS, such as BPEL4WS.

As an immediate improvement, we are collaborating with the GIS group of Murcia University in the formal validation of the models.

Likewise we are working in the MIDAS-CASE subsystem for model transformation, which will include the mapping rules proposed in MIDAS. This way, we will be able to develop a Web Service starting from a conceptual model, such a pure UML class diagram.

Finally, we will have an MDA tool that support the complete development process proposed in MDA for Web Services: PIM definition, direct transformation from the PIM to the PSM for Web Services platform and automatic code generation of the WSDL description from the PSM model. This description shall be immediately transformed into the code that deploys the Web Service for different technologies.

Acknowledgment

This research is carried out in the framework of the project GOLD (TIN2005-00010), financed by the Ministry of Science and Technology of Spain.

References

- [1] Altova Company. "XMLSPY 5". Retrieved from: http://www.xmlspy.com/features_wsdl.html, 2003.
- [2] BEA Systems, Inc. "Bea Web Logic Workshop". Retrieved from http://www.beasys.es/productos/weblogic/weblogic_workshop.jsp.
- [3] Borland, Inc. "Borland Delphi Studio", "Borland C++ Builder", "Borland Jbuilder", "Borland Web Services Kit for Java".
- [4] P. Caceres, E. Marcos, B. Vela. "A MDA-Based Approach for Web Information System Development". In *Proceedings of the Workshop in Software Model Engineering*. (WiSME@UML2003), 2003.
- [5] Codagen: "Codagen Architect", <http://www.codagen.com/products/architect/default.htm>
- [6] J. Conallen, *Building Web Applications with UML*. Addison Wesley, 2000.
- [7] IBM, "IBM Studio Application Developer", <http://www-306.ibm.com/software/awdtools/studioappdev/>.
- [8] IMS Global Learning Consortium, Inc. "IMS General Web Services UML to WSDL Binding Auto-generation Guidelines". Retrieved from http://www.imsglobal.org/gws/gwsv1p0pd/ims_gws_transfv1p0pd.html
- [9] Iopsis Software: "Iopsis iNsiht", retrieved from <http://www.iopsis.com/products/insight.html>.
- [10] JGraph. "JGraph - The Java Open Source Graph Drawing Component". <http://www.jgraph.com>.
- [11] A. Kleppe, J. Warmer, W. Bast. *MDA Explained, The Model Driven Architecture: Practice and Promise*. Addison Wesley, 2003.
- [12] V. Kulkarni, S. Reddy. "Separation of Concerns in Model-Driven Development". *IEEE Software*, Vol. 20 (5), pp. 64-69, 2003.
- [13] E. Marcos, P. Cáceres, V. De Castro. "An approach for Navigation Model Construction from the Use Cases Model". In *Proceedings of the 16th Conference On Advanced Information Systems Engineering*. CAISE FORUM, pp. 83-92, 2004.
- [14] E. Marcos, V. De Castro, B. Vela. "Representing Web Services with UML: A Case Study." In *Proceedings of the International Conference on Service Oriented Computing (ICSOC)*. pp. 15-27, 2003.
- [15] E. Marcos, B. Vela, J. M. Cavero, "Methodological Approach for Object-Relational Database Design using UML". *Journal on Software and Systems Modeling (SoSyM)*. Vol. 2, pp.59-72, 2003.
- [16] J. Miller, J. Mukerji. "MDA Guide V1.0.1". OMG Document - omg/03-06-01, Object Management Group, 2001.
- [17] OMG. "OMG Unified Modeling Language Specification. Version 1.5". Retrieved from: <http://www.omg.org/technology/documents/formal/uml.htm>, 2003.
- [18] Oracle Corporation. "Oracle Jdeveloper 10g". Retrieved from <http://www.oracle.com/technology/products/jdev>.
- [19] Systinet. "WASP Server for Java, 4.7.1, WASP Server for C++, 4.6, WASP UDDI, 4.6 SP1, Systinet Developer for Sun One Studio 4.6, Systinet Developer for Borland Jbuilder 4.6". <http://www.systinet.com/>.
- [20] Sun Microsystems Inc. "Sun Studio 9". Retrieved from <http://www.sun.com/software/products/studio/>.
- [21] J. M. Vara, C. J. Acuña, E. Marcos, M. Lopez. "Desarrollo de un Sistema de Información web: una experiencia con Oracle XMLDB". *Revista del Círculo de Usuarios Oracle de España (CUORE)*. Vol. 27, *Vivat Acenmia* (9) pp. 3-12, 2004.
- [22] J. M. Vara, V. De Castro, P. Caceres, E. Marcos. "Arquitectura de MIDAS-CASE: una herramienta para el desarrollo de SIW basada en MDA". In *Proceedings of the IV Jornadas Iberoamericanas en Ingeniería del Software e Ingeniería del Conocimiento (JIISIC'04)*. pp.83-92. 2004.
- [23] B. Vela, E. Marcos. "Extending UML to represent XML Schemas". In *Short Paper Proceedings of the 15th Conference On Advanced Information Systems Engineering*. (CAISE FORUM). 2003.
- [24] webMethods. "webMethods Glue". Retrieved from <http://www.webmethods.com/meta/default/folder/000008895>.
- [25] World Wide Web Consortium (W3C) Working Draft 3. "Web Services Description Language (WSDL) Version 1.2". Retrieved from: <http://www.w3.org/TR/wsd112/>, 2003.
- [26] W3C Working Draft 3. "Web Services Description Language (WSDL) Version 1.2: Bindings". Retrieved from: <http://www.w3.org/TR/2003/WD-wsd112-bindings-20030124/>, 2003.
- [27] W3C XML Schema Working Group. "XML Schema Parts 0-2:[Primer, Structures, Datatypes]". W3C Recommendation. Retrieved from: <http://www.w3.org/TR/xmlschema-0/>, <http://www.w3.org/TR/xmlschema-1/> and <http://www.w3.org/TR/xmlschema-2/>, 2001.

Author Biographies

Juan M. Vara has obtained his degree on Computer Science Engineering at the Rey Juan Carlos University. He has done the Doctoral Courses on the Computer Science and Mathematical Modeling Program of the Informatics, Statistics and Telematics department of the Rey Juan Carlos University where he is doing his Ph. Thesis focused on Model-Driven Engineering for the development of Web Information Systems.

At present he works as assistant professor at the department of Informatics Languages and Systems of the Rey Juan Carlos University, where he is a member of the Kybele Research Group. He is co-author of several publications at

national and international events and he has participated on several research projects.

Valeria De Castro has obtained her degree on Information Systems Engineering at the National Technological University of Resistencia, Argentina where she worked during two years as assistant professor. She has obtained a grant on the Spanish Agency for International Cooperation to make a pre-doctoral stay at the Rey Juan Carlos university. She has done her Doctoral Courses on the Computer Science and Mathematical Modeling Program of the Informatics, Statistics and Telematics department of the Rey Juan Carlos University. At present she is making her Ph. Thesis, focused on Software Engineering for the Web and she works as assistant professor at the department of Informatics Languages and Systems of the Rey Juan Carlos University, where she is included at the Kybele Research Group. She is co-author in several publications on national and international congresses and she has participated on several research projects.

Esperanza Marcos obtained his Computer Science Engineer degree as well as her Ph. D. at the Technical University of Madrid. She is also graduated on Computer Science at the Valladolid university. Nowadays she is Associated Professor at the Rey Juan Carlos university where she is the head of the Kybele Research Group, whose research topics are mainly focused on Information Systems Engineering, specially on model-driven methodologies, Web Information Systems, etc. She has given several courses and specialization masters; At present she collaborates on the "Software Engineering Master" at the Technical University of Madrid. She is also author of a vast amount of books, book chapters and national and international articles. Finally, she has also led several research projects and she has participated in others as well.