Representing WSDL with Extended UML

V. de Castro, E. Marcos and B. Vela *

Abstract

Web services are emerging to provide a systematic and extensible framework for interactions of applications, built on existing Web protocols and based on open XML standards. In spite of the existence of several middleware platforms that support Web service development, the lack of a solid methodological base for the development of Web services, as well as service-oriented applications, give rise to the need of methods or modeling techniques that guarantee the quality in the development of this kind of applications. MIDAS is a model-driven methodological framework for the agile development of Web Information Systems (WIS). MIDAS is based on Model Driven Architecture (MDA) and defines Platform Independent Models (PIMs) and Platform Specific Models (PSMs) according to the aspects of content, hypertext and behavior; it proposes some mapping rules between the different models. MIDAS proposes to model the behavior of a WIS following the service oriented approach. In this work we focus on the proposed PSM for WIS behavior modelling, presenting the UML extension for the representation of Web Service Description Language (WSDL). The proposed extension provides a UML notation which on one hand, allows obtaining graphic representation of a Web service and, on the other hand facilitates the automatic generation of WSDL code from a UML diagram.

Keywords: *UML extension, WSDL, Web Service Modeling, Web Information Systems.*

Resumen

Los servicios Web proveen un marco sistemático y extensible para la interacción de aplicaciones a través de la Web, basado en XML y construido sobre protocolos Web existentes. Aunque existen varias tecnologías que facilitan el desarrollo, la carencia de una metodolgía sólida para el desarrollo tanto de servicios Web, como de aplicaciones orientadas a servicios, plantea la necesidad de nuevos métodos o técnicas de modelado que puedan garantizar la calidad en el desarrollo de este tipo de aplicaciones. MIDAS es un marco metodológico orientado a modelos para el desarrollo de Sistemas de Información Web. MIDAS está basado en MDA (*Model Driven Architecture) y propone modelos PIM (Platform Independent Models) y PSM (Platform Specific Models) acorde a tres aspectos: Contenido, Hipertexto y Comportamiento; además define guías para la generación de modelos y para la transformación de los mismos. En MIDAS proponemos modelar el comportamiento del SIW desde una perspectiva orientada a servicios y en este trabajo presentamos uno de los PSM propuestos para ello, llamado WSDL Model. Dicho modelo es una extensión de UML para el modelado de servicios Web, basada en el estándar Web Service Description Language (WSDL). La extensión propuesta aporta una notación en UML que, por un lado, permite obtener una representación gráfica de un servicio Web y por otro, facilitará la generación automática de código WSDL a partir de un diagrama UML.*

Palabras clave*: extensión de UML, WSDL, modelado de servicios Web, sistemas de información basados en la web.*

*Kybele Research Gruop - Rey Juan Carlos University - Madrid (Spain)
{emarcos,vcastro,vbela}@escet.urjc.es

1 Introduction

In the last decade, many businesses and organizations have used different approaches to interact with others taking the advantages and infrastructure of the Web. Web services are emerging to provide a systematic and extensible framework for application-to-application interactions, built on existing Web protocols. Web services are based on open XML standards and define a standardized mechanism to describe, locate and communicate with online applications [6]. Now, the services are one of the most important issues in the scope of Web Information Systems (WIS) development. There are several middleware platforms, such as JAVA or .NET that allow implementing Web services and facilitate the service-oriented applications development. However, the lack of a solid methodological base for the development of Web services, as well as service-oriented applications give rise to the need of methods or modeling techniques that can guarantee the quality in the development of this kind of applications.

In the last years a large amount of modeling techniques and methodologies for the development of WIS [5, 7, 9] and service-oriented WIS [18] have appeared. MIDAS [4, 13] is a model-driven methodological framework for agile development of WIS, that proposes to use standards in the development process. It is based on XML [3] and proposes to model the whole system in UML [15].

MIDAS defines Platform Independent Models (PIMs) and Platform Specific Models (PSMs) according to three aspects [11]: *content*, *hypertext* and *behavior*, and also proposes some mapping rules between the different models. As MIDAS proposes to use UML notation to model the whole system, both the PIMs and PSMs will be represented in this notation. In this work we focus on the WIS behavior modeling and we present one of the PSMs proposed by MIDAS for this purpose.

MIDAS proposes to model the behavior of a WIS following the service oriented approach, in [11] we have presented an approach to obtain a navigation model which integrates the structured information and the *services*, providing a unified view of the structural and behavioral aspects of a WIS. We define a *service* as a specific functionality that the WIS offers to the user. A service is a conceptual definition which will exist just at PIM level. A service could be represented, at PSM level, by one Web service or by the composition of several Web services. So, a service can be *basic* for example to validate an email address; or a *composite* as to locate the best prices on ticket airlines that probably involve many basic services. The *basic* and *composite services* concepts are introduced in [16]. So much *basic services* as *composite services* utilize "service description" to offer their functionality so that they can be located and utilized through the Web.

A *Web service* is a software component, independent from platform and implementation, that can be: described using a service description language, published to a registry of services, discovered through a standard mechanism, invoked through a declared API (Applications Programming Interface) and composed with other services [8].

Web Services Description Language (WSDL) [20] is the language to describe Web services proposed by the World Wide Web Consortium (W3C). WSDL describes three fundamental properties of the Web service: what a service does (the operation that the service provides), how a service is acceded (details of the data formats and protocols necessary to access the service's operation) and where a service is located (details of protocols-specific network address, such as URL). In this work a UML extension for Web service representation based on WSDL is proposed. This extension has been carried out with a double purpose: on one hand, to give a UML notation that allows representing a Web service and, on the other hand, to facilitates the automatic generation of WSDL description of a Web service from an UML diagram.

Some other related works with Web service modeling and automatic WSDL code generation have appeared during the last years [2, 23]. However these proposals have some limitations with respect to our goals. The extension proposal in [2] is not complete, since it does not allow operations and parameters modeling, neither relation between these components and others like input or output messages. Since one of our goals is to make easy the automatic generation of Web services description in WSDL from a UML diagram, it will be necessary to define modeling guidelines that allow representing all the needed issues for Web services description maintaining the main benefit of modeling that is the reality abstraction. The XMLSPY5 case tool [23] allows automatic generating of WSDL documents, but starting from its own graphical notation instead of from an UML diagram.

The rest of the paper is structured as follows: section 2 is an overview of the MIDAS methodology that represents the framework of this work. In section 3 the WSDL metamodel is described. In section 4 the UML extension for WSDL is proposed, in section 5 we present a case study using the proposed notation. Finally, in section 6, we conclude underlining the main contributions and the future works.

2 MIDAS Framework

MIDAS is a model-driven methodological framework for agile development of WIS, it is based on the OMG's Model-Driven Architecture (MDA) [14], that proposes: a) to specify the whole system with Computation Independent Models (CIM), Platform Independent Models (PIM) and Platform Specific Models (PSM); b) to generate the mapping rules between the models. MIDAS suggests using the Unified Modeling Language [15] (UML) as unique notation to model both PIMs and PSMs. It also proposes using some practices coming from agile methodologies, as eXtreme Programming [1].

As it is stated in [10], at the modeling level it is important to identify the different aspects of the system in order to model them independently. To identify the aspects of a WIS, as MIDAS follows a service oriented approach, we have taken as a reference the middleware architectures of the Web service development platforms, as .NET or J2EE. So, we propose a n-tier model architecture. Until now, our approach considers three aspects corresponding with the three tiers most commonly accepted: *graphic user interface*, *persistence* and *business logic* tiers. For the sake of uniformity with the most commonly used Web Engineering terminology, these aspects will be called *hypertext*, *content* and *behavior* respectively [17]. The advantage of this n-tier model architecture is that it is easily scalable; so, to incorporate new aspects, as security or management, we have just to introduce a new tier.

Summing up, MIDAS propose to model the WIS according to two orthogonal dimensions (see figure 1): a) the MDA approach of the OMG: that is, the dependence degree of the platform; b) the n-tier model architecture previously defined: that is, the aspects to be considered in a WIS. So, it defines PIMs and PSM models for each of the aspects above identified, hypertext, content and behavior, as well as the mapping rules between them.

As stated above, MIDAS proposes to model the behavior of the WIS following the service oriented approach. In this work we propose an UML extension that we call *WSDL model*, for the Web service modeling at PSM level. The MIDAS framework has been partially presented in [4]; the UML extension for object-relational, XML schema can be found in [12, 19]; in [13, 11] can be found the development process of the content aspect.

For the WSDL model definition we will describe the WSDL metamodel and then we will present the UML extension proposed and a case study in which it has been applied.

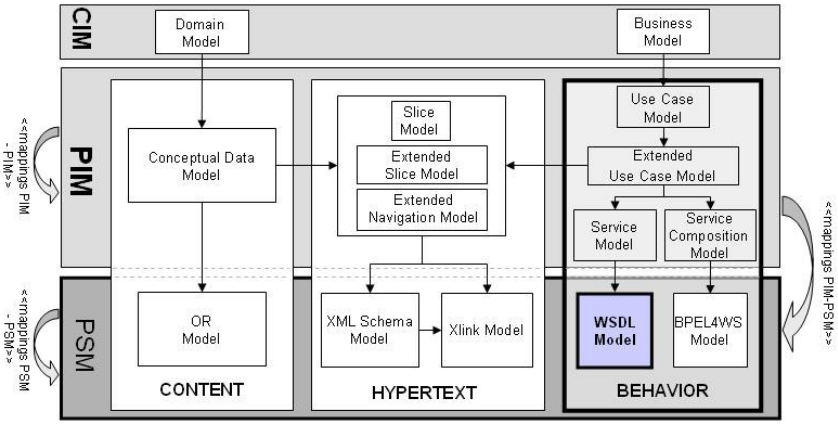

Figure 1: Midas Framework

3 WSDL Metamodel

WSDL [20, 21] is the language proposed by the W3C for Web services description. A WSDL document is a XML document which specifies the operations that a service can perform. One of the advantages of WSDL is that it allows separating the abstract functionality description offered by a service from the concrete details description, such as message format and communication protocol that can be SOAP, HTTP or MIME.

WSDL describes the Web services through the *messages* that are exchanged between the service provider and requester. The messages exchange between the service provider and requester 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 both network protocol and message format, define 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 shadowed components represent the concrete issues of the service description and the rest represent abstract issues of service description.

A WSDL document contains a version number and a root DEFINITION component. It has a *Name* and *TargetNameSpace* attribute 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 [22] 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 for it which indicates the document location. In the same way the IMPORT component is used to reuse WSDL documents, the document name and location are needed.

The PORTTYPE component is the most important component in WSDL, since it describes the operations that the service offers, that is, the interface. The OPERATION component groups the set of messages that will be interchanged between service provider and 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*

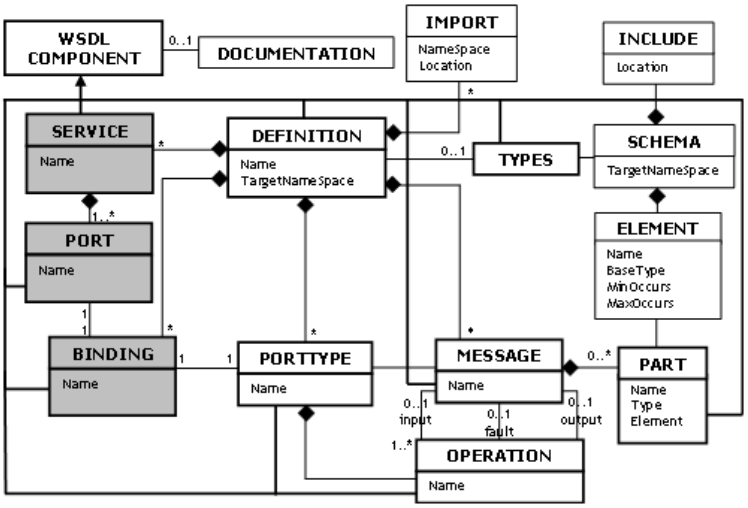

Figure 2: WSDL metamodel represented in UML

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 base type XSD (int, float, string, etc.) or a type defined in the TYPES section. In this last 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 specifically defined message format and communication protocol, such as SOAP, HTTP or MIME [21]. 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. It is related with the BINDING component that describes how and where (by the *location* attribute) to interact with the service interface.

4 Representing WSDL with UML

As we have already said, we propose to represent WSDL with UML. So, we have to extend it so that it allows us to represent each one of component proposed by WSDL and described in the previous section. 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 [5] 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 [15].

Before showing the proposed UML extension in section 4.2, we will explain the design guidelines that have been defined for the representation of WSDL in extended UML.

4.1 Design Guidelines for the UML Extension

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

- DEFINITION component has been considered as stereotyped class because they are explicitly defined in WSDL and constitute the root component that groups all the used elements.
- TYPES and SCHEMA components have been considered compositions stereotyped with $\ll TypeSchema\gg$ 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 explicitly defined in WSDL.
- 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 $\ll Part\_Type\gg$ if the PART component uses the ELEMENT as a *type*, and by means of an association stereotyped with $\ll Part\_Elements\gg$ if the PART component uses the ELEMENT as an *element*.
- The relation between the OPERATION component and the MESSAGE component will be represented by means of an association stereotyped with $\ll Imput\gg$, $\ll Output\gg$ o $\ll Fault\gg$, depending on the type of message that it associates, that is an *input message*, an *output message* or a *fault message.*
- MESSAGE, PORTTYPE, BINDING, SERVICE e 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 data type definitions that will be used for message sending. For this reason we use the UML extensions to represent XML Schemas proposed in [19].

4.2 The UML extension

This UML extension defines a set of stereotypes, tagged values and constraints that enable us to represent WSDL in graphical notation in UML. The UML extension is defined for the specific WSDL components proposed by the W3C. Each WSDL component should be able to be represented in graphical notation with this UML extension.

- **Class DEFINITION**
  Metamodel class: Class
  Description: A class represents a ≪***DEFINITION***≫ component of the WSDL metamodel.
  Attribute: TargetNameSpace 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.

- **Class ELEMENT**
  Metamodel class: Class
  Description: An ≪***ELEMENT***≫ class represents an element of the XML Schema.
  Constraints: 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.

- **Composition Type_Schema**
  Metamodel class: composition
  Description: 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.
  Constraints: 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.

- **Class MESSAGE**
  Metamodel class: Class
  Description: A ≪***MESSAGE***≫ class represents a MESSAGE component of the WSDL metamodel.
  Constraints: It must be related to the ≪*DEFINITION*≫ class by means of a composition and must be associated to at least one ≪*PART*≫ class.

- **Class PART**
  Metamodel class: Class
  Description: A ≪***PART***≫ class represents a PART component of the WSDL metamodel.
  Constraints: It must be related to one ≪*MESSAGE*≫ class by means of a composition.
  Attribute: Type 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.

- **Association Part_Type and Part_Element**
  Metamodel class: Association
  Description: 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.
  Constraints: 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.

- **Class PORTTYPE**
  Metamodel class: Class
  Description: **A ≪*PORTTYPE*≫** class represents a PORTTYPE component of the WSDL metamodel.

  Constraints: It must be related to the ≪*DEFINITION*≫ class by means of a composition and must be associated at less one ≪*OPERATION*≫ class.

- **Class OPERATION**
  Metamodel class: Class
  Description: An **≪*OPERATION*≫** class represents an OPERATION component of the WSDL metamodel.
  Constraints: It must be related to the ≪*PORTTYPE*≫ class by means of an aggregation.

- **Association Input, Output and Fault**
  Metamodel class: Association
  Description: an **≪*Input*≫**, **≪*Output*≫** or **≪*Fault*≫** association represents a relationship between an OPERATION component and MESSAGE component of the WSDL metamodel.
  Constraints: 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.

- **Class BINDING**
  Metamodel class: Class
  Description: A **≪*BINDING*≫** class represents a BINDING component of the WSDL metamodel.
  Constraints: It must be related to the ≪*DEFINITION*≫ class by means of a composition and it must be associated to only one ≪*PORTTYPE*≫ class.

- **Class SERVICE**
  Metamodel class: Class
  Description: A **≪*SERVICE*≫** class represents a SERVICE component of the WSDL metamodel.
  Constraints: It must be related to the ≪*DEFINITION*≫ class by means of a composition and it must be composed by at least one ≪*PORT*≫ class.

- **Class PORT**
  Metamodel class: Class

Figure 3: WSDL description of a *"ValidateEmail"* Web Service

Description: **A** ≪***PORT***≫ class represents a PORT component of the WSDL metamodel.
Attribute: Location is an URL (Uniform Resource Locator). It is mandatory and identifies the access point to the service.
Constraints: It must be related to the ≪*DEFINITION*≫ class by means of a composition and must be associated to only one ≪*BINDING*≫ class.

- **Class IMPORT**
  Metamodel class: Class
  Description: An ≪***IMPORT***≫ class represents an IMPORT component of the WSDL metamodel.
  Attribute: Namespace 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. Location is an URI. It is optional and indicates the location of some information for the namespace.
  Constraints: It must be related to the ≪*DEFINITION*≫ class by means of a composition.

5 A Case Study

In this section we show through a case study the UML extension proposed for the representation of WSDL. The case study consists of a service for validating an email address called *"ValidateEmail"*. Figure 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 *"ValidateEmailAddress"* as a data type. The output message *"ValidateEmailAddressSoapOut*" defines also one part *"ParametersOut"*

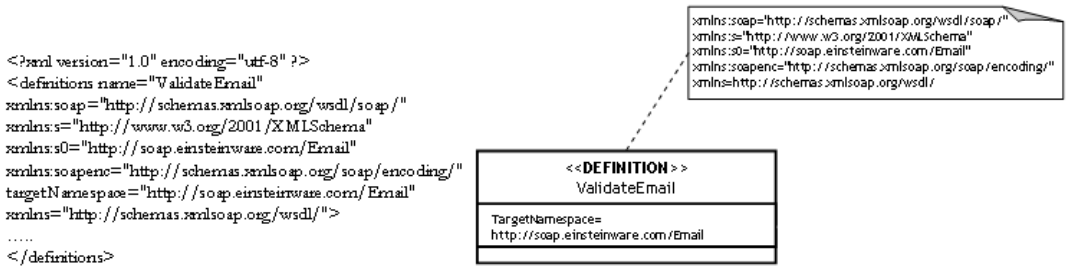

Figure 4: Representation of the DEFINITION component

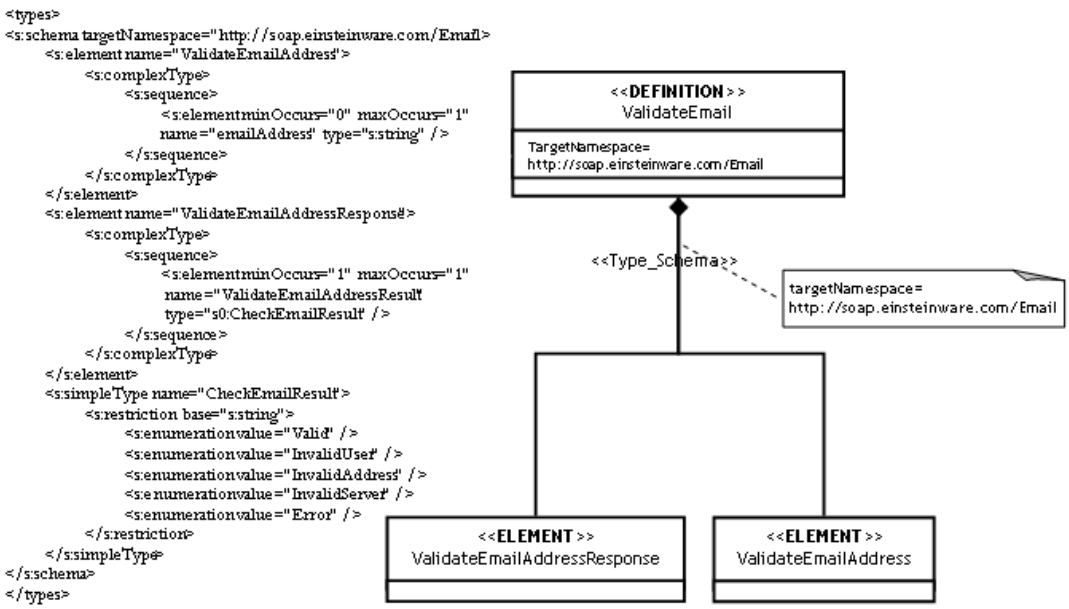

Figure 5: Representation of the ELEMENT, TYPES and SCHEMA components

which uses the element *"ValidateEmailResponse"* as a data type. Both, *"ValidateEmailAddress"* and *"ValidateEmailResponse"* elements have been defined in the section types. The porttype *"EmailServicePortType"* groups the operations that will be performed by the service that in this case is only one. 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 DEFINITION component. The *Name* attribute 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 clarity reasons a tagged value will be represented as a note associated to the element that uses it.

Figure 5 shows the ≪***TypesSchema***≫ composition that represents a relationship between the ≪***DEFINITION***≫ class and the data types defined, *"ValidateEmailAddress"* and *"ValidateEmailResponse"*. The *TargetNameSpace* attribute of SCHEMA component will be represented as tagged values.

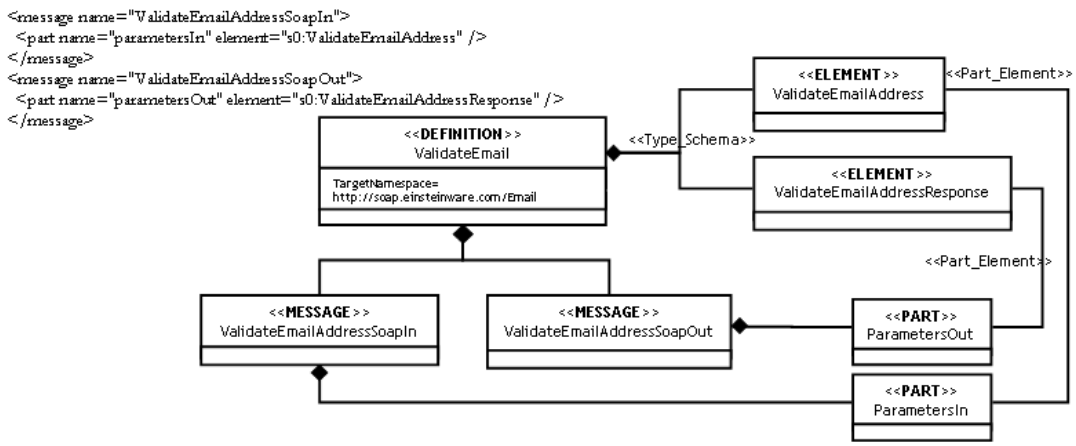

Figure 6: Representation of the MESSAGE and PART components

Figure 6 shows the representation of the MESSAGE and PART components. The *"ValidateEmailAddressSoapIn"* message contains one part which has associated the *"ValidateEmailAddress"* element as a data type, by means of the element attribute. Therefore, the existing association between *"ParametersIn"* part and *"ValidateEmailAddress"* element is stereotyped with $\ll$***Part_Element***$\gg$. In the same way, the existing association between *"ParametersOut"* part and *"ValidateEmailResponse"* element is stereotyped with $\ll$***Part_Element***$\gg$.

Figure 7 shows the representation of the PORTTYPE and OPERATION components. The *"EmailServicePortType"* uses one operation *"ValidateEmailAddress"*, therefore an aggregation is represented between them. The operation defines two messages, the input message *"ValidateEmailAddressSoapIn"* is related with the operation *"ValidateEmailAddress"* by mean a $\ll$***Input***$\gg$ association and the output message *"ValidateEmailAddressSoapOut"* is related with the operation *"ValidateEmailAddress"* by mean a $\ll$***Output***$\gg$ association.

Figure 8 shows the representation of the BINDING component, without representing connection with SOAP protocol. The *"EmailServiceBinding"* describe the binding to the porttype; therefore an association between the *"EmailServiceBinding"* binding and the *"EmailServicePortType"* porttype is represented.

Finally, figure 9 show the representation of the SERVICE and PORT components. The *"EmailService"* service contains one port, *"EmailServiceSoap"* therefore the composition is represented between them. The *Location* attribute indicates the service URL and is represented as a class attribute.

Figure 10 shows the representation of "ValidateEmail" Web service using defined UML extension.

6 Conclusion

In the last decade, many businesses and organizations have used different approaches to interact with others taking the advantages and infrastructure of the Web. The Web services are emerging to provide a systematic and extensible framework for application-to-application interactions. The services are one of the most important issues in the scope of WIS development but, the lack of a solid methodological base for the development of Web services, as well as service-oriented applications give rise to the need of methods or modeling techniques that can guarantee the quality in the development of this kind of applications.

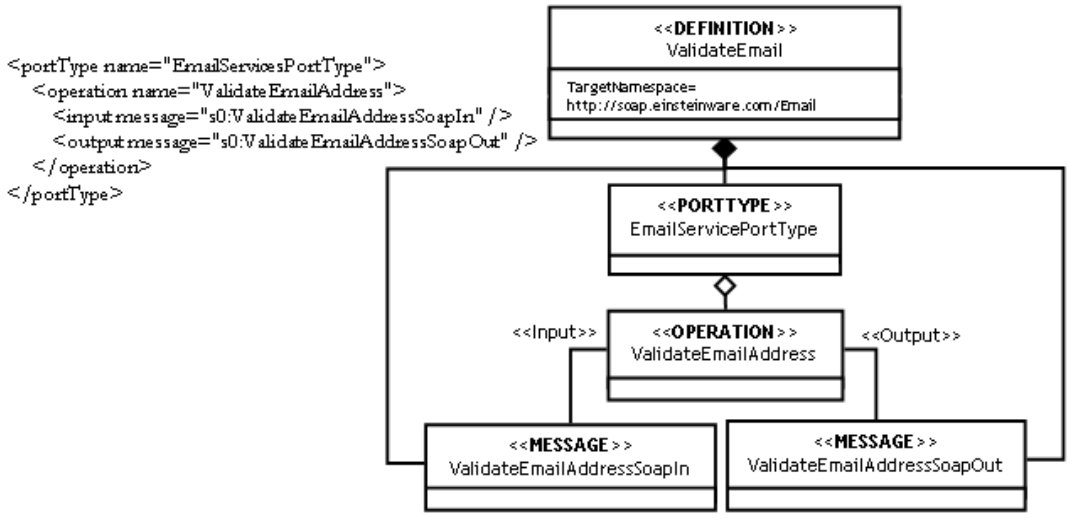

Figure 7: Representation of the PORTYPE and OPERATION components

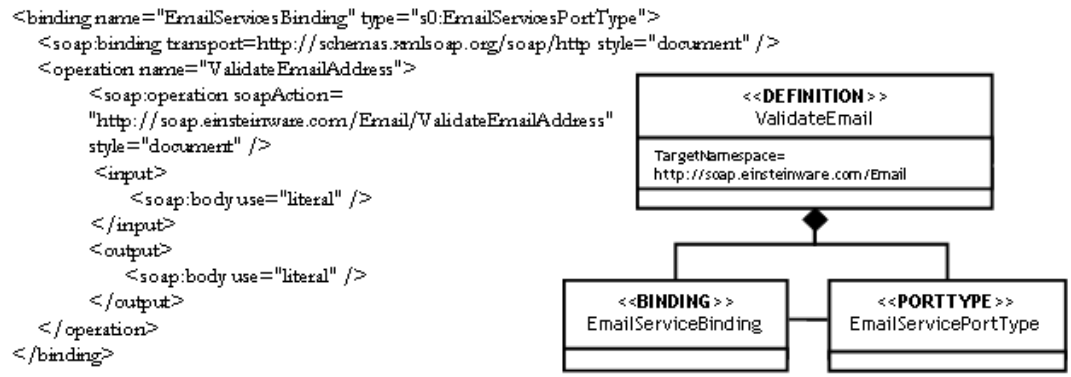

Figure 8: Representation of the BINDING component

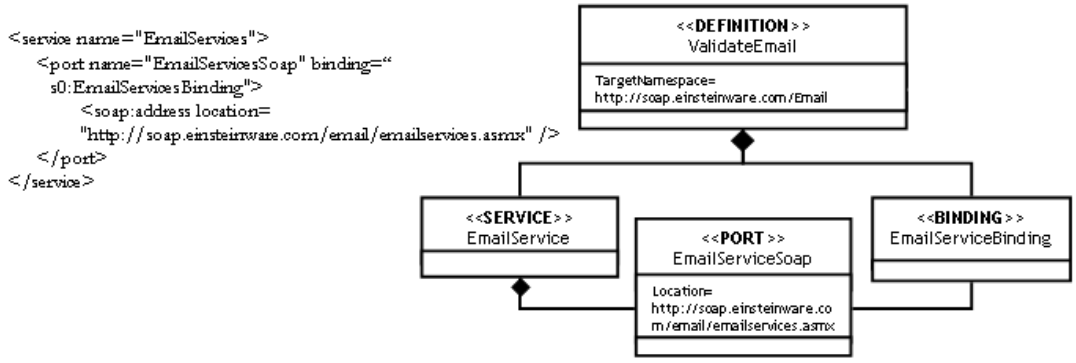

Figure 9: Representation of the SERVICE and PORT components

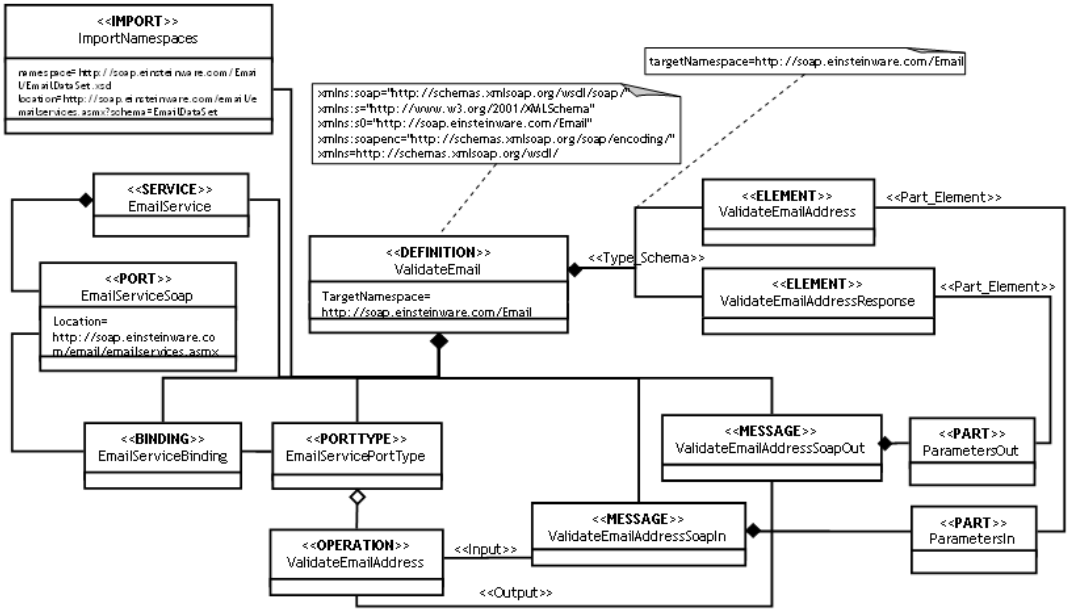

Figure 10: UML representation of web service "ValidateEmail"

In this paper we have presented a UML extension to represent WSDL. This work is integrated in MIDAS, a model-driven methodological framework for the agile development of WIS based on MDA. It defines PIMs and PSMs according to the aspects of content, hypertext and behavior, and proposes to use the UML as unique notation to model both PIMs and PSMs.

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 paper we have showed a Web service for validating an email address represented whit the defined UML extension.

Now we are working in the definition of the necessary extensions for the complete description of the service including the connection to specific protocols (SOAP, HTTP and MIME) and the automatic generation of the service WSDL description from a UML diagram. Also we are working in the integration of techniques for the Web services composition in MIDAS, such as BPEL4WS. As future work we are going to implement the proposed models and mapping rules in a CASE tool that supports MIDAS.

Acknowledgement

This research is carried out in the framework of following projects: EDAD (07T/0056/2003 1) financed by Autonomous Community of Madrid and DAWIS financed in part by the Ministry of Science and Technology of Spain (TIC 2002-04050-C02-01) and the Rey Juan Carlos University (CG-2003-12).

References

[1] S. Ambler, Agile Model Driven Development is Good Enough. *IEEE Software*, 20(5):71-73, 2003.

[2] C. Armstrong. Modeling Web Services with UML. In: *OMG Web Services Workshop 2002*, Retrieved from: http://www.omg.org/news/meetings/workshops/webservices_2002.htm, 2003.

[3] T. Bray et al. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation. Retrieved from: http://www.w3.org/TR/2000/REC-xml-20001006/, 2000.

[4] P. Cáceres et al. A MDA-Based Approach for Web Information System Development. In: *Workshop in Software Model Engineering in conjunction with UML Conference*, San Francisco, 2003.

[5] J. Conallen. Building Web Applications with UML. Addison Wesley, 2000.

[6] F. Curbera et al., Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2):86-93, 2002.

[7] P. Fraternali, Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys*, 31(3):227-263, 1999.

[8] S. Graham et al. Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI. SAMS, 2002.

[9] N. Koch; H. Baumeister and L. Mandel, Extending UML to Model Navigation and Presentation in Web Applications. In: *Modeling Web Applications, Workshop of the UML 2000*, England, 2000.

[10] V. Kulkarni and S. Reddy, Separation of Concerns in Model-Driven Development. *IEEE Software*, 20(5):64-69, 2003.

[11] E. Marcos; P. Cáceres and V. de Castro. An Approach for Navigation Model Construction from the Use Case Model. In: *The 16th Conference On Advanced Information Systems Engineering. CAISE'04 FORUM*, Accepted to be published.

[12] E. Marcos; B. Vela and J. M. Cavero. Extending UML for Object-Relational Database Design. In: *Fourth International Conference on the Unified Modelling Language, UML 2001*, Toronto, LNCS 2185, p. 225-239, 2001.

[13] E. Marcos et al. MIDAS/DB: a Methodological Framework for Web Database Design. In: *DASWIS 2001*, Yokohama, LNCS-2465, p. 227-238, 2002.

[14] OMG. OMG Model Driven Architecture. Miller, J., Mukerji, J. (eds.) 2001. Document number ormsc/2001-07-01. Retrieved from: http://www.omg.com/mda, 2003

[15] OMG. OMG Unified Modeling Language Specification. Version 1.5. Retrieved from: http://www.omg.org/technology/documents/formal/uml.htm, 2003.

[16] M.P. Papazoglou and D. Georgakopoulos, Serviced-Oriented Computing. *Communications of ACM*, 46(10):25-28, 2003.

[17] W. Retschitzegger and W. Schwinger. Towards Modeling of Data Web Applications - A Requirement's Perspective. In: *Proceedings of the America's Conference on Information Systems*, p. 149-155, 2000.

[18] J.J. Rodríguez; O. Díaz and F. Ibánez. Moving Web Services Dependencies at the Front-end. In: *Engineering Information Systems in the Internet Context 2002*, p. 221-237, 2002.

[19] B. Vela and E. Marcos. Extending UML to represent XML Schemas. In: *The 15th Conference On Advanced Information Systems Engineering. CAISE'03 FORUM*, Klagenfurt/Velden, p. 16-20, 2003.

[20] W3C Web Services Description Language (WSDL) Version 1.2. W3C Working Draft 3 March 2003. Retrieved from: http://www.w3.org/TR/wsdl12/, 2003.

[21] W3C Web Services Description Language (WSDL) Version 1.2: Bindings. W3C Working Draft 3 March 2003. Retrieved from: http://www.w3.org/TR/2003/WD-wsdl12-bindings-20030124/, 2003.

[22] 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.

[23] XMLSPY 5. Retrieved from: http://www.xmlspy.com/features_wsdl.html, 2003.