Representing Web Services with UML: A Case Study

Esperanza Marcos, Valeria de Castro, and Belén Vela

Kybele Research Group
Rey Juan Carlos University
Madrid (Spain)
{e.marcos, vcastro, b.vela}@escet.urjc.es

Abstract. Nowadays services are one of the most important issues in the scope of the Web Information Systems (WIS). Although, there is a great amount of Web services, still it do not exist methods or modelling techniques that can guarantee quality in services and service-oriented applications development. MIDAS is a model-driven methodology for the development of WISs and is based on UML, XML and object-relational technology. Web services represent a new dimension in WIS development, in which the systems are constructed by means of transparent integration of services available in the Web. WSDL is the language proposed by the W3C for Web service description. In this paper, an UML extension for Web services modelling defined in WSDL is described through a case study.

1 Introduction

In the last decade the Web has become one of the main channels to share and to spread information. Services are one of the most important issues in the scope of Web Information Systems (WIS). One of the central ideas of this approach is that future applications will be conceived like a collection of services available through the Web. So, for example, companies and organizations could encapsulate their business processes and publish them like services in the Web, or request other available services and integrate them to their applications, to provide new solutions.
Several technologies, such as JAVA or .NET allow implementing this kind of applications. However, there is not any solid methodological basis for service-oriented system and Web services development. For this reason, new methods or modelling techniques are needed to guarantee the quality in service-oriented WIS and Web services development. In the last years a large amount of modelling techniques and methodologies for the development of WIS [5,8,11] and service-oriented WIS [15] have appeared. MIDAS [13,4] is a model-driven methodology for the development of WIS, that proposes to use standards in the development process It is based on UML [2], XML [3] and object-relational technology [7]. MIDAS selects, adapts and integrates, if possible, the best techniques and notations of existing methodologies and also defines some new ones if necessary. Thus, for example, the UML extension for object-relational database design [12,14] and the UML extension to represent XML Schemas [16] have been defined.
Fig. 1 shows the MIDAS architecture, which has a system core that represents the domain and business models. Over this central core we define a ring which includes

M.E. Orlowska et al. (Eds.): IC-SOC 2003, LNCS 2910, pp. 17–27, 2003.

both the structural and the behavior dimension of the system. The core and this ring represent the technology and platform independent modeling. The external ring focuses on the different platforms and supported technologies. We will focus on the behavioral dimension of MIDAS, which is the marked part in figure and we propose to model the systems behavior with Web Services.

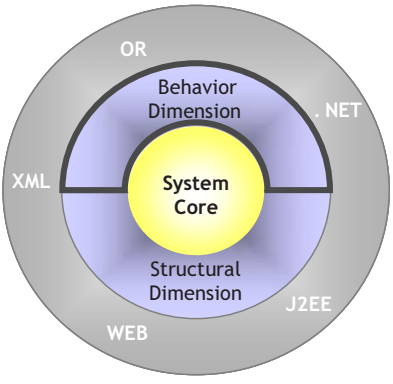

Fig. 1. MIDAS Architecture

The Web Services technologies provide a neutral language that accelerates integration of applications inside and outside the enterprise [10]. A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that describes a collection of operations that will be accessible through the Web by means of standardized XML messaging. Web Services Description Language (WSDL) is the language to describe Web Services proposed by the W3C [17]. In this paper an UML extension for Web Services modelling based on WSDL is proposed. This extension has two purposes: to make easy both, the Web Services documentation at a high level of abstraction and the automatic generation of Web services description in WSDL from UML diagram.

Some other works related with Web services modelling and automatic WSDL code generation have appeared during the last years [1,20]. However these proposals have some limitations with respect to our goals. The extension proposal in [1] is not complete, since it does not allow operations and parameters modelling, neither relations 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 UML diagram, it will be necessary to define modelling guidelines that allow representing all the needed issues for Web services description maintaining the main benefit of modelling that is the reality abstraction. XMLSPY5 [20] case tool 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 organised as follows: section 2 WSDL metamodel is described. In section 3 the UML extension for Web services modelling through a case study is proposed; finally, section 4 sums up the main conclusions and further research topics.

2 The WSDL Metamodel

WSDL [17,18] is a markup language proposed by the W3C for describing Web services. A WSDL document is an XML document which specifies the operations that a service can perform. One of the advantages of WSDL is that it enables separating the abstract functionality description offered by a service from description of concrete details, such as "how" and "where" that functionality is offered [6,9].
WSDL describes Web services through the *messages* that are exchanged between the service provider and requestor. An exchange of messages between the service provider and requestor are described as an *operation*. A collection of operations is called a *port type*, which define the service interface in abstract way. The *binding* between a *port type* and concrete network protocol and message format define the service interface in concrete way.
Fig. 2 shows the WSDL metamodel represented by an UML class diagram. The shadowed components represent the concrete issues of service description and the rest represent abstract issues of service description.

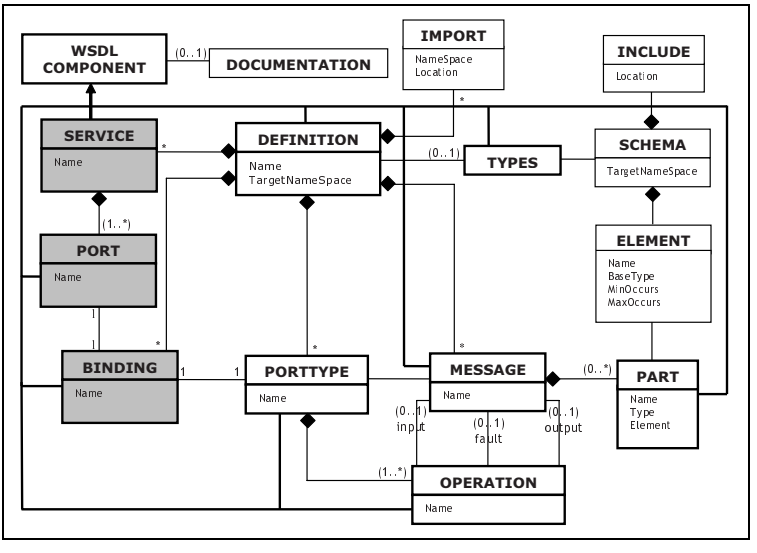

Fig. 2. WSDL metamodel represented in UML

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 associates 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 [19] and contains a SCHEMA component in which, namespaces and data types are defined. WSDL allows to include XML Schemas documents defined previously, for that uses a INCLUDE component 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 realizes, 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 *fault message*. A MESSAGE contains a *Name* attribute and zero or more PART components. The PART component describes one portion of particular message that 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 associated through a *type* or *element* attribute.
A BINDING component describes the binding of a PORTTYPE component and associated operations to concrete message format and communication protocol, such as SOAP, HTTP or MIME [18]. 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 paper 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 *location* attribute) to interact with the service interface.

3 Extended UML for Web Services Modelling

In order to represent each one of proposed elements by WSDL and described in the previous section, it will be necessary to extend the UML using its own extension mechanisms [2]. 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 [16].
The proposed extension for Web services modelling will be described through a case study. Firstly, we explain the criteria that have been used for the definition of UML extension. Next, in section 3.2 we describe in detail the service and next we formalize the extension explaining the use of proposed stereotypes.

3.1 Design Guidelines for the Definition of UML Extension

To choose the stereotypes necessary to represent in UML all the components of WSDL and their relations the following criteria are used:

- DEFINITION components have been considered as stereotyped classes 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 stereotyped compositions with <<*TypeSchema*>> and represent the relation between a DEFINITION component and the data type definitions.
- MESSAGE, PART, PORT TYPE, OPERATION, BINDING, PORT and SERVICE components have been considered stereotyped classes because they represent important components and explicitly defined in WSDL.

- MESSAGE components will be related to the DEFINITION component, by means of a composition and must be associated, at least, one PART component.
- PART components will be related to the MESSAGE component that it used, by means of a composition. In addition the will have associate the data types that will be used in the message. Each PART component must be associate to only one MESSAGE component.
- PORTTYPE component will be related to the DEFINITION component, by means of a composition and will have associated, at least, one OPERATION component.
- OPERATION components will be related by means of an aggregation to the PORTTYPE component that defines its. In addition the MESSAGE component that its use, will have associated.
- BINDING components will be related to the DEFINITION component, by means of a composition and must be associated to one PORT TYPE component.
- PORT components will be related to only one SERVICE component and must be associated to only one BINDING component.
- SERVICE components will be related to the DEFINITION component, by means of a composition and must be associate, at least, one PORT component.

3.2 Web Services Modelling: A Case Study

We present the UML extension for web services modelling. For this, we have taken as a case study a flight information service of an airport called *"FlightService"*. Fig. 3 shows the *"FlightService"* Web service description in WSDL.

Fig. 3. WSDL description of a "FlightService" Web service

The Web service defines two operations *"GetFlightInfo"* and *"CheckIn"*. The operation *"GetFlightInfo"* has two messages, an input and an output message. The input message *"GetFlightInfoInput"* contains two parts: *"AirlineName"* and *"FlightNum"*, which use a base type XSD, string and int, respectively. The output message "*GetFlightInfoOutput"* has only one part *"FlightInfo"*, this part uses the element *"TypeFlightInfo"* as a data type, which is associated through a *type* attribute. The operation *"CheckIn"* contains only one input message *"CheckInInput"*. This message has only one part *"Body"*, which use the element "Ticket" as a data type and is associated through an element attribute.
The port type *"AirportServPortType"* groups the operations that will be performed by the service. The link between this port type and the SOAP protocol is described by the *"AirportServBinding"* element.
The service has only one port *"FlightServicePort"*, which defines through an URL the Web service location.

The UML extension to represent a Web service will be described considering the design guidelines previously established in section 3.1. Next, we will explain the used stereotypes for each component, its constraints and its tagged values.

DEFINITION component will be represented by means of stereotyped class **<<DEFINITION>>**. The *Name* attribute will be the name of the class and *TargetNameSpace* attribute will be represented as a class attribute. The used namespaces will be included as tagged values. Tagged values will be associated to element that defines its as a note, see Fig. 4.

Fig. 4. Representation of the DEFINITION component

Data types that will be used for messages sending will be represented by means of stereotyped classes **<<ELEMENT>>**, which will be related by means of **<<TypesSchema>>** composition, to **<<DEFINITION>>** class. The namespaces used in data type definitions as also *TargetNameSpace* attribute of SCHEMA component will be represented as tagged values. In the example data types *"TypeFlightInfo"* and *"Ticket"* are showed but without representing its complete structure. Will be indicated an order number which appear in the document as a tagged value of a **<<ELEMENT>>** class to maintain the correspondence between the WSDL document and the UML model, see Fig. 5.

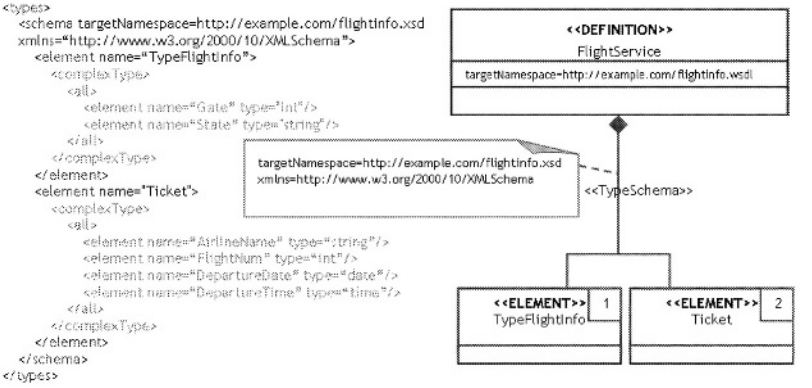

Fig. 5. Representation of the ELEMENT, TYPES and SCHEMA components

MESSAGE component will be represented by means of a **<<MESSAGE>>** stereotyped class. The *Name* attribute will be the name of the class and will have an order number as a tagged value. All **<<MESSAGE>>** classes will be related to a **<<DEFINITION>>** class by means of a composition. The PART component of each message will be represented by means of a **<<PART>>** stereotyped class that will have an order number as a tagged value with the message order number as prefix. In the example, the "*GetFlightInfoInput"* message has two parts which use a base type that are represented as a class attribute. The "*GetFlightInfoOutput"* message contains one part which has associated the "*TypeFlightInfo"* data type through a *type* attribute that has been previously defined. Therefore, the existing association between "*FlightInfo"* part and "*TypeFlightInfo"* element is stereotyped with **<<part_type>>**. The "*CheckInInput"* message contains one part which has associated the "*Ticket"* data type through a *element* attribute that has been previously defined. Therefore, the existing association between "*Body"* part and "*Ticket"* element is stereotyped with **<<part_element>>**. The **<<MESSAGE>>** class will have associated, at least, one **<<PART>>** class and each **<<PART>>** class will be associate to only one **<<MESSAGE>>** class, see Fig. 6.

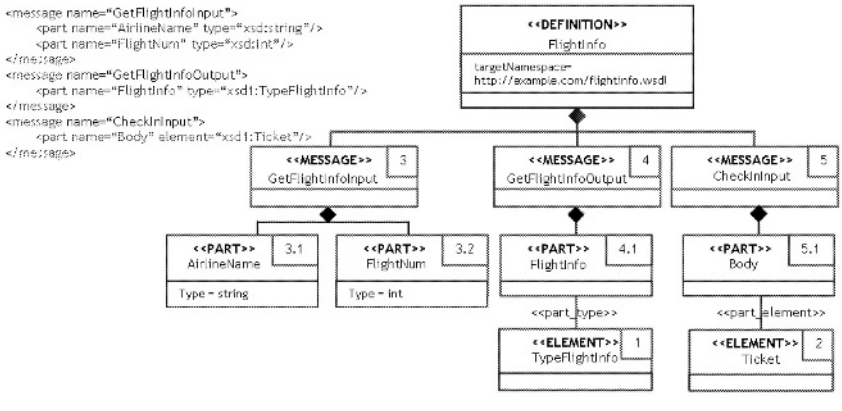

Fig. 6. Representation of the MESSAGE and PART components

PORTTYPE component will be represented by means of a **<<PORTTYPE>>** stereotyped class. The *Name* attribute will be the name of the class and will have an order number as a tagged value. The **<<PORTTYPE>>** classes will be related to a **<<DEFINITION>>** class by means of a composition. To represent OPERATION component, a **<<OPERATION>>** stereotyped classes will be used, that will have an order number as a tagged value with the PORTTYPE order number as prefix. A **<<PORTTYPE>>** class will have to be associated, at least, one **<<OPERATION>>** class by means of a composition. Each operation will be associated with the messages that it use and association stereotypes will be **<<input>>**, **<<output>>** or **<<fault>>** depending on the way which messages are used, see Fig. 7.

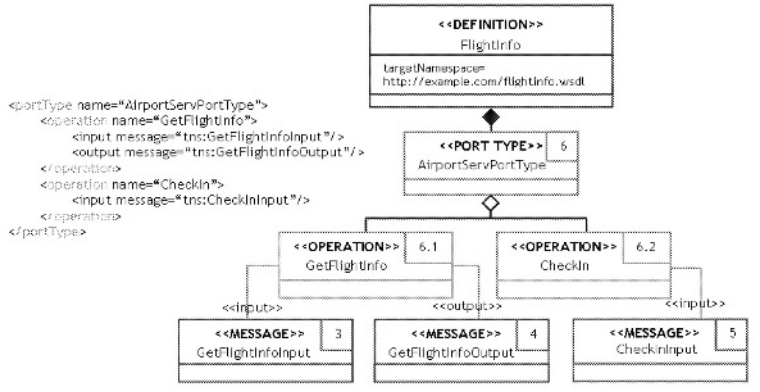

Fig. 7. Representation of the PORTTYPE and OPERATION components

BINDING component will be represented by means of a **<<BINDING>>** stereotyped class. The *Name* attribute will be the name of the class and will have order number as tagged value. The **<<BINDING>>** classes by means of a composition to a **<<DEFINITION>>** class and it will be related by means of an association to a **<<PORTTYPE>>** class that it describes. In the example BINDING component is showed without representing connection with SOAP protocol, see Fig. 8.

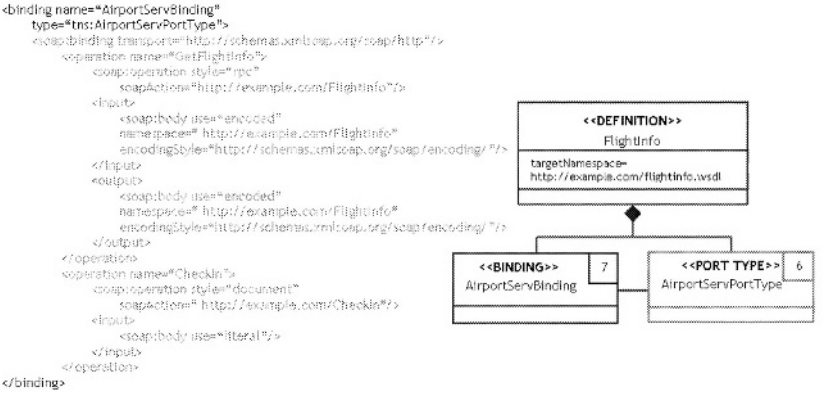

Fig. 8. Representation of the BINDING component

In order to represent SERVICE component a **<<SERVICE>>** stereotyped class will be used. The *Name* attribute will be the name of the class and will have an order number as a tagged value. The **<<SERVICE>>** classes will be related to a **<<DEFINITION>>** class by means of a composition and must be composite, at least of one **<<PORT>>** class which represent PORT component. This class will be related by means of an association to a **<<BINDING>>** class. The **<<PORT>>** class will have an order number as a tagged value with the SERVICE order number as prefix. The *Location* attribute indicates the service URL and will be represented like a class attribute. The **<<PORT>>** class will be related with only one **<<BINDING>>** class, see Fig. 9.

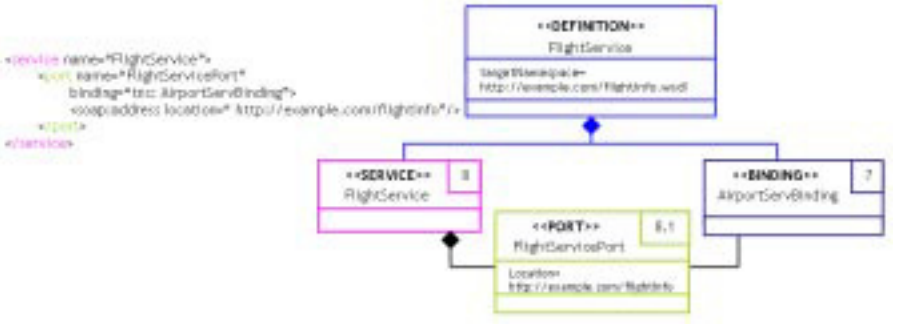

Fig. 9. Representation of the SERVICE and PORT components

Fig. 10 shows the UML representation of "FlightService" Web service taken as case study, using the defined extensions.

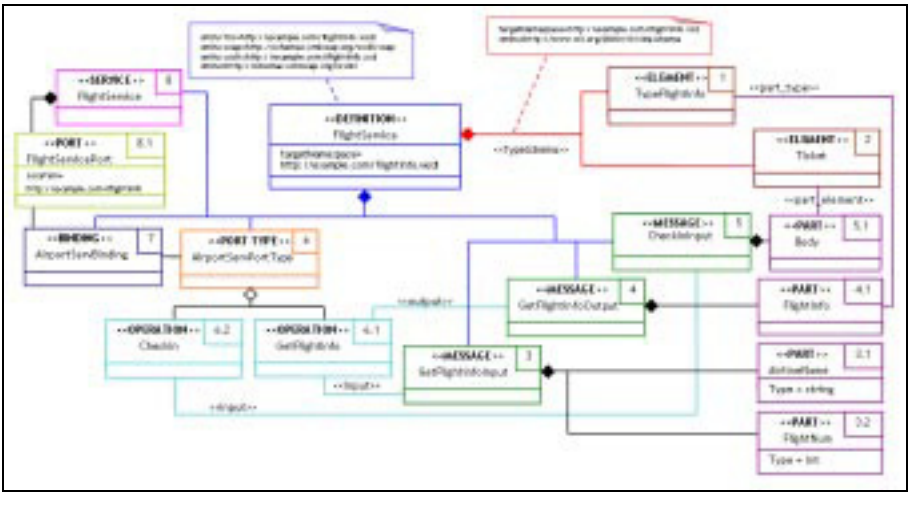

Fig. 10. UML representation of "FlightService" Web service

4 Conclusions and Further Research Topics

In this paper we have presented, by mean of a case study, an UML extension to model Web services defined in WSDL. This proposal is integrated in MIDAS, a methodological framework for WIS development, which proposes UML for the whole system modelling.
Firstly we have described the WSDL metamodel using UML. Next, a Web service which offers airport flight information and its UML graphical representation has proposed using the defined extension.
Actually 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). Also we are studying the implementation of the proposed extension as Add-In for Rational Rose, with the aim to allow the automatic generation of the service WSDL description of, from a UML diagram. In addition we are working in the incorporation of integration techniques in MIDAS that will allow us to compose several Web Services.

Acknowledgements. This research is carried out in the framework of the 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 (PIGE 02-05).

References

1. Armstrong, C., *Modeling Web Services with UML*. OMG Web Services Workshop 2002. Retrieved from: http://www.omg.org/news/meetings/workshops/webservices_2002.htm, 2003.
2. Booch, G., Rumbaugh, J. and Jacobson, I., *The Unified Modelling Language User Guide*. Addison Wesley, 1999.
3. Bray, T., Paoli, J, Sperberg-McQu4een, C. M. and Maler, E., *Extensible Markup Language (XML) 1.0 (Second Edition),* W3C Recommendation. Retrieved from: http://www.w3.org/TR/2000/REC-xml-20001006/, 2000.
4. Cáceres, P., Marcos, E., Vela, B., *A MDA-Based Approach for Web Information System Development.* Workshop in Software Model Engineering in conjunction with UML Conference. October, 2003. San Francisco, USA. Accepted.
5. Conallen, J., *Building Web Applications with UML*. Addison Wesley, 2000.
6. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana, S., *Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI.* Internet Computing, IEEE, Volume: 6, 2 , Mar/Apr 2002, pp. 86–93.
7. Eisenberg, A. and Melton, J., *SQL:1999, formerly known as SQL3*. ACM SIGMOD Record, Vol. 28, No. 1, pp. 131–138, March, 1999.
8. Fraternali, P., *Tools and approaches for developing data-intensive Web applications: a survey.* ACM Computing Surveys, Vol. 31, nº 3, 1999.
9. Graham, S., Simeonov, S., Boubez, T., Davis, D., Daniels, G., Nakamura, Y. and Neyama, R., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI.* SAMS, 2002.
10. Gottschalk, K., Graham, S., Kreger, H. and Snell, J., *Introduction to Web services architecture.* Retrieved from: http://researchweb.watson.ibm.com/journal/sj/412/gottschalk.html, 2003.

11. Koch, N., Baumeister, H. and Mandel, L., *Extending UML to Model Navigation and Presentation in Web Applications*. In Modeling Web Applications, Workshop of the UML 2000. Ed. Geri Winters and Jason Winters, York, England, October, 2000.
12. Marcos, E., Vela, B., and Cavero, J. M., *Extending UML for Object-Relational Database Design*. Fourth Int. Conference on the Unified Modelling Language, UML 2001, Toronto (Canadá), LNCS 2185, Springer Verlag, pp. 225–239, 2001.
13. Marcos, E., Vela, B., Cáceres, P. and Cavero, J.M., *MIDAS/DB: a Methodological Framework for Web Database Design.* DASWIS 2001. Yokohama (Japan), November, 2001. LNCS-2465. Springer Verlag. ISBN 3-540-44122-0. September, 2002.
14. Marcos, E., Vela, B. and Cavero, J.M., *Methodological Approach for Object-Relational Database Design using UML.* Journal on Software and System Modeling (SoSyM). Springer-Verlag. Ed.: R. France and B. Rumpe. Accepted to be published.
15. Rodríguez, J.J., Díaz, O. and Ibánez, F., *Moving Web Services Dependencies at the Front-end*. Engineering Information Systems in the Internet Context 2002, pp.221–237, 2002.
16. Vela, B. and Marcos, E., *Extending UML to represent XML Schemas.* The 15th Conference On Advanced Information Systems Engineering (CAISE'03). CAISE'03 FORUM. Klagenfurt/Velden (Austria). 16–20 June 2003. Ed: J. Eder, T. Welzer. Short Paper Proceedings. ISBN 86-435-0549-8. 2003
17. W3C *Web Services Description Language (WSDL) Version 1.2*. W3C Working Draft 3 March 2003. Retrieved from: http://www.w3.org/TR/wsdl12/, 2003.
18. 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.
19. 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.
20. XMLSPY 5. Retrieved from: http://www.xmlspy.com/features_wsdl.html, 2003.