TOWARDS A CONTEXTUAL MODEL-DRIVEN DEVELOPMENT APPROACH FOR WEB SERVICES

Zakaria Maamar[1], Karim Baïna[2], Djamal Benslimane[3], Nanjangud C. Narendra[4] and Mehdi Chelbabi[5]

[1]*Zayed University, Dubai, U.A.E.*
[2]*ENSIAS, Mohammed V-Souissi University, Rabat, Morocco*
[3]*Claude Bernard Lyon 1 University, Lyon, France*
[4]*IBM India Research Lab, Bangalore, India*
[5]*CERIST, Algiers, Algeria*

Keywords: Context, Model-Driven, UMLProfile, and Web service.

Abstract: This paper discusses how we develop and apply a contextual model-driven approach to Web services. A Web service is defined using WSDL, posted on an UDDI registry, and invoked through a SOAP request. To deploy adaptable Web services, we consider the environment in which these Web services operate. This environment's features exist in a structure, which we refer to as context. By adopting a contextual model-driven approach, we aim at developing contextual specifications of Web services. To this end ContextUML, an extension of UML through UMLProfile, permits developing these contextual specifications.

1 INTRODUCTION

For the World Wide Web Consortium (W3C), a Web service *is a software application identified by a URI, whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based applications*. Web services constitute an attractive way for running B2B scenarios and addressing the challenges of these scenarios like distribution, heterogeneity, and coordination. Over the last few years tremendous efforts have been put in developing standards for Web services in terms of just to cite a few description, discovery, composition, and security (Ma, 2005). Particularly Web services composition is object of intense investigations. Composition emphasizes the complex nature of user demands and the inability of a single Web service to satisfy these demands by itself. Hence a collection of interacting Web services - which forms high-level business processes known as composite Web services - is deemed appropriate to satisfy user demands.

Relying on standards for Web services and aiming to ease their development, several design approaches are proposed to those who are put on the front line of satisfying businesses' promise of delivering Web services-based solutions. Some approaches adopt models (Bézivin et al., 2004) while others adopt software agents (Huhns, 2002). In this paper, the focus is on models. A model has a vocabulary and is a kind of representation that first, permits abstracting the features of an application domain and second, supports interaction with end-users during validation. In a model-driven approach once the design of a model is completed, it is automatically converted into program code (Grϕnmo et al., 2004). With regard to Web services, the mapping of models onto runnable specifications should be geared towards the requirements of the development stage of a Web services application. These requirements are about Web services definition, composition, or even interaction.

Although the pervasive adoption of Web service (Kulkarni et al., 2005), Web services still lack the widespread acceptance level of traditional integration middleware like CORBA and DCOM, primarily due to two reasons. The first reason is the trigger-response pattern that frames the interactions of Web services with peers (Maamar et al., 2006a). The second reason is that Web services are still seen by some as distributed objects that react upon request (Birman, 2004). To tackle both reasons, a Web service needs to be context-aware so it can detect and respond to changes in its environment. In (Maamar et al., 2006a), we discussed why Web services are to eb enhanced with capabilities that permit them to be more flexible in managing the situations in which they participate. Therefore, Web services can independently decide when and how to process requests, why and how to delay their participation in some compositions, why

Maamar Z., Bāına K., Benslimane D., C. Narendra N. and Chelbabi M. (2006).
TOWARDS A CONTEXTUAL MODEL-DRIVEN DEVELOPMENT APPROACH FOR WEB SERVICES.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - ISAS*, pages 78-85
DOI: 10.5220/0002442000780085
Copyright © SciTePress

and how to reject requests for security reasons, etc.

The integration of context into a model-driven approach for Web services development is not straightforward. Many problems arise including how do we represent and structure context in a model, how do we specify a Web service binding to context, how do we map a contextual model onto a contextual program code, how do we provide an automatic tool for this conversion, and how do we show the dynamic nature of context? A contextual model complies with the argument that ”there is no unique way to look at data and there is no unique way to represent data” (Arara and Benslimane, 2004) and, opens up as well research opportunities in the field of multi-representation. A contextual model would permit abstracting a situation based on the expected use (e.g., maintenance, monitoring) of this abstraction. During modeling, several elements are strengthened based on the context that is under consideration. These elements include the amount of information to integrate into a model, the representation and organization of this information, the granularity level of this information, and the rationale of this information to the considered context.

This paper discusses the value-added of context to a model-driven approach and to suggest extensions (if needed) to such an approach so the obtained implementable specifications of Web services are contextualized, too. The remainder of this paper is set out as follows. Section 2 overviews prior work on model-driven approaches, and discusses the challenges associated with the multi-representation of Web services. Section 3 uses a scenario to discuss the need for Web services composition and to shed the light on the challenges that this composition faces when a model-driven approach is adopted. Section 4 introduces our ongoing work on a contextual model-driven approach that we propose in terms of definition and mapping steps. Section 5 contains our conclusions.

2 BACKGROUND

2.1 Prior Work

Model-driven approaches have been subject to various investigations as the literature shows (Mellor et al., 2003). Usually the complete design model of an application comprises multiple models, each model offers a different viewpoint on the application. For instance, UML offers models ranging from use-cases and class diagrams to deployment diagrams, which are used depending on the objectives to reach during the design of an application.

In a model-driven approach, the emphasis is on the design of very detailed models and platform independent specifications. For Backx, a model-driven approach addresses the migration problems between computing platforms, which are error-prone and time consuming (Backx, 2004). By having models that are language and platform independent, same models can be applied in different scenarios. To boost embracing a model-driven approach by designers, the following models are produced (Backx, 2004): the Computation Independent Model (CIM) for business requirements (or business model), the Platform-Independent Model (PIM) for system functionalities, and the Platform Specific Model (PSM) for platform specific details. Initially the work starts with preparing a CIM to be mapped onto a detailed PIM. A PIM is the starting point of the derivation of the PSM according to the characteristics of the target platform using specific mapping rules. The last step consists of relying on a model-driven approach-enabled tool to generate the complete code for the system from the PSM.

In (Baïna et al., 2004) the authors suggested a framework to perform a model-driven development of Web services, generating a set of extensible service implementation templates and a executable service specifications. In this framework, a protocol specification of a Web service identifies the message exchange sequences that the Web service supports (constraints on the invocation order of the Web service operations). A continuation of the work of Bana et al. is an UML-based modeling language for model-driven and context-aware Web services (Sheng and Benatallah, 2005). More details are given in Section 4.2.

Zarras et al. adopted a model-driven approach for the development of a methodology for the dependability analysis of composite Web services (Zarras et al., 2004). In (Bordbar and Staikopoulos, 2004) the authors noted that developing Web services by applying model-driven approaches has received considerable attention. However, most of the existing research focuses on the transformation of models that express the static structure of the system. Therefore, Bordbar and Staikopoulos have decided to study transformations that concern the dynamic structure of the system (Bordbar and Staikopoulos, 2004). This structure is modelled using behavioral models, expressing how the various components collaborate in order to manage a task and to fulfill the system functionalities. Bordbar and Staikopoulos have developed transformation mechanisms from UML activity diagrams to BPEL specifications.

2.2 Multi-representation

Recent works on Web services define new standards to include more semantic and contextual information when describing, publishing, discovering, and composing Web services. Many researchers recognize that semantics is context-dependent: there is no single way to look at data and functionalities of web services

Table 1: Proposed contextual definition of a Web service.

Context C_1	Context C_2	Context C_3

Interface WS_A{
int op_0(string a);
string op_1(string a, int b);
string op_2 (int a)}
	Interface WS_A{
	int op_0(string a);
	string op_1(string a, string b)}
		Interface WS_A{
		string op_1(string a, int b)}

and there is no unique way to represent them.

Web services are described in WSDL language to capture a set of functionalities that they offer and to precise the description of input and output data that they receive and deliver. The description of Web services is necessary defined according to a given context or a given abstraction/conceptualization of the real world. A context can be viewed as various criteria such as the abstraction paradigm, the granularity scale, interest of user communities, etc. So, a same domain application (Travel agency, libraries,) can have more than one set of Web services, where each set describes the available functionalities in a particular context. We call each one of these set of web services, mono-representation web services. Our motivation is to see how we can describe Web services according to several contexts at the same time. We shall call such Web services description a multi-representation Web services (MuWS).

WSDL Language is not capable of defining a multi-representation Web services. For instance, there isn't any possibility to define a unique WSDL description of a web service with different sets of functionalities at the same time. So, our motivation of this technical opinion is to deal with the problem of multi-representation in the context of Web services description. Our proposal can be illustrated through the example given in Table 1. Let us consider three real world representations, identified by the context names C_1, C_2 and C_3 corresponding to administrator, employee, and customer contexts. Some comments made on Table 1 show the necessity of extending WSDL for the needs of Web services multi-representation:

- Some operations are only visible in some contexts. For instance, operation op_0 only appears in contexts C_1 and C_2. It is not available in context C_3.
- Some input and output parameter definitions differ from one context to another. For instance, operation op_1 appears in the three contexts with two different parameter definitions.
- Some operation's functionalities and results can differ when context is switched to a different one. For instance, the result of operation op_1 in context C_1 can be more or less general than the result of the same operation in other contexts.

From the point of view of Web services multi-representation, many works can be considered. The authors in (Spaccapietra et al., 2000) looked into the multi-representation problem in spatial databases so different geometries, resolutions, and scales are associated with the same spatial object. In Spaccapietra et al.'s work, MADS was suggested as an extension of the Entity-Relationship model. A stamping mechanism of data elements (concepts, attributes, and instances) and relationships is suggested to enable manipulations of data elements from several representations. In object-oriented modeling, several object-role languages are proposed to represent an object with a single structure to be dynamically enhanced with specific information (roles) related to its different facets (Gottlob et al., 1996; Pernici, 1990). These languages allow, for instance, objects to move from one type to another, to acquire new instances as needed, and even to be dynamically regrouped to form new classes while keeping their membership in their original classes.

3 RUNNING SCENARIO

Consider the case of Melissa, a tourist who is visiting Dubai. Upon arriving, she browses some Web sites of tourist spots in and around Dubai. The top-ranked Web site offers different Web services that can be composed according to different ready-to-use patterns (information on the use of Web services is not released to Melissa). Melissa logs on at the Web site and chooses sightseeing and shopping Web services. Melissa's plans are to visit outdoor places in the morning and go shopping in the afternoon. The first part of the plan is subject to weather forecasts; outdoor activities are cancelled in case of hot weather. Initially Melissa is asked to select outdoor places, and to indicate the pickup/drop-off times for sightseeing and shopping.

With regard to the first activity of Melissa's plan, sightseeing Web service consults with weather Web service about the forecasts for the coming five days. If there is no warning of hot weather, the scheduling of the places to visit begins by ensuring that these places are open for public on these days, and transportation and guide are arranged. The logistics of

Melissa's rides is assigned to transportation Web service, which identifies for instance the vehicle type and the possibility that Melissa shares rides with other tourists heading towards the same places. In case of hot-weather warning, sightseeing Web service might suggest other places (e.g., museums) where indoor activities can take place. Similar description applies to shopping, which consists of checking out the running promotions in the malls that Melissa has selected. Transportation Web service coordinates shopping's beginning time with sightseeing's finishing time.

Fig. 1 shows a rough representation of the Web services chronology that implements Melissa scenario. This chronology yields insight into the multiple challenges that a contextual model-driven approach faces including: how is context related to Web services; what are the types of information that feature context; what are the different models to produce, and how do we guarantee the consistency of mapping contextual models onto contextual specifications of Web services?

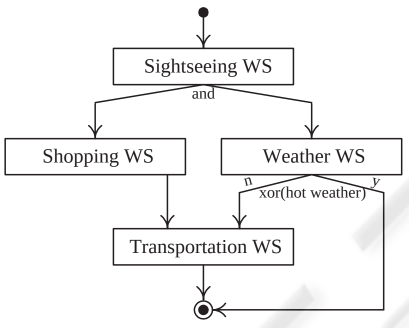


Figure 1: Chronology of Web services in Melissa scenario.

4 TOWARDS A CONTEXTUAL MODEL-DRIVEN APPROACH

Any contextual model-driven approach has to take advantage of the OMG's findings. OMG initiated the development of the model-driven architecture (Miller and Mukerji, 2001). This architecture is a framework for system development that proposes specifying a complete system with platform independent models to be converted into multiple platforms according to the system's technical characteristics.

4.1 Definitions

A model graphically depicts a system's structure and behavior from a certain point of view and at a certain level of abstraction (Sendall and Kozaczynski, 2003). ”A certain point of view” and ”at a certain level of abstraction” constitute what we call here context of the model.

During modeling, several aspects need to be geared towards the type of context (e.g., time, location, user, resource), which is under consideration. These elements include the amount of information that needs to be included in the model, the representation and organization of this information, the granularity level of this information, and the relevancy (or rationale) of this information for this context. Relying on a contextual model-driven approach, an application will have several models, where each model sheds the light on the elements that are relevant to a specific context (Section 2.2).

When a contextual model-driven approach is used, it is expected that the specifications to obtain out of this approach will have a dedicated structure that identifies context and makes it apparent in these specifications through, for example, stereotypes, specialized tags, or constraints. Besides this dedicated structure, mechanisms to manage context are deemed appropriate for inclusion in a contextual model-driven approach. Context is sensed, acquired, refined, and distributed over the interested parties. In (Maamar et al., 2006b), we discussed how a Web service connects to context. A Web service requires awareness mechanisms, so it can take advantage of the information that context caters. These mechanisms collect contextual raw data from sensors for the purpose of detecting changes in the environment. A change needs to be evaluated by the Web service's assessment module, so the deployment module of the Web service takes actions over the environment.

4.2 ContextUML - UML Extension

The basic understanding of Web services revolves around three steps: definition, announcement/discovery, and binding. A fourth step, known as composition, is added but does not directly contribute to this understanding. The first three steps feature the primary steps upon which a service-oriented architecture is built: a service provider provides a service, a service requestor is the client that requires a service to be performed, and a service agency provides registration and discovery facilities. Since the announcement step, which targets an UDDI registry, is expected to happen independently of the surrounding environment of a Web service, we excluded this step from our discussions on context integration into a model-driven approach. We also exclude from this paper how Web services bindings are contextualized. We only present the extensions that need to be put forward so the definition of Web services is contextualized, i.e., contextual WSDL.

A model-driven approach has a formalism that is used for representing the models to produce. In this

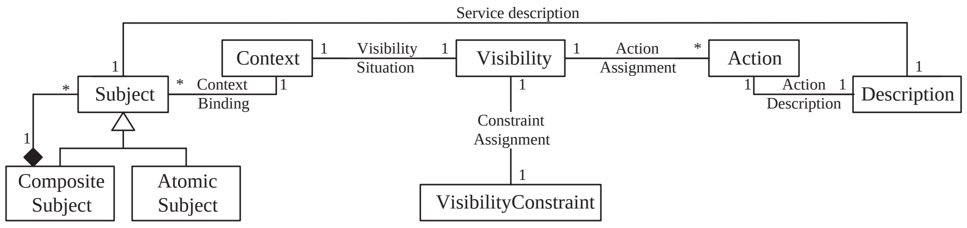


Figure 2: ContextUML metamodel.

paper, we adopt UML for three reasons: UML is the de-facto standard for modeling systems, UML can be extended, through **UMLProfile** (Amsden et al., 2003), in order to meet the modeling specificities and requirements of a particular application domain, and several projects have successfully shown the mapping between UML models and Web services specifications (Section 2.1). When it comes to extending UML, we refer to the work of Lodderstedt et al., who suggested SecureUML. It is a modeling language that defines a dedicated vocabulary for annotating UML-based models with information related to security access-control (Lodderstedt et al., 2002). Moreover, Bonnet has positively argued about the role of UMLProfile in modeling Web services specifications of type WSDL and BPEL (Bonnect, 2004). In this paper, we propose **ContextUML** as a modeling language that defines a dedicated vocabulary for annotating UML-based models with information related to context. Although Sheng and Benatallah propose also ContextUML (Sheng and Benatallah, 2005), it is different from our ContextUML at the following levels.

1. We aim at extending the existing Web services' specifications like WSDL at the meta-level;
2. We aim at developing generic contextual Web services' specifications regardless of the type of context whether time, location, etc;
3. We aim at developing a contextual methodological approach where the life cycle of a Web service from definition, announcement, and binding is contextualized.

Prior to integrating context into Web services specifications, we make clear the levels of use of ContextUML. We consider two levels: *triggering* and *self-management*.

- A contextual specification of a Web service at the *triggering* level corresponds to the parts of a Web service, which are made visible (i.e., accessible) according to a specific context. By parts, we mean the elements that populate for example a Web service's WSDL file. Interesting to note that the visibility option at this level opens up the opportunity of looking into the multi-representation issue of a Web service (Section 2.2).
- A contextual specification of a Web service at the *self-management* level corresponds to the mechanisms that permit a Web service to oversee the different compositions wherein it participates. These mechanisms are thoroughly explained in our previous work (Maamar et al., 2005).

In this paper, ContextUML only concerns the triggering level since the self-management level is intrinsic to a Web service and does not have to be revealed to external parties. The categorization of ContextUML's levels of use into triggering and self-management is backed by the work done in (Grϕnmo and Solheim, 2004). The authors consider that there are two main aspects that can be modelled in the area of Web services: the Web service and the workflow. Web service modeling identifies the functionalities to be exposed with their interfaces and operations, while workflow modeling identifies the control and data flows from one component Web service to the next component (this includes compensation in case of execution failures of the Web service). The first type of modeling matches quite well the triggering level of context use, and the second type of modeling matches quite well the self-management level of context use.

4.3 Abstract Syntax of ContextUML

ContextUML or the UMLProfile for contextual Web services is the outcome of extending UML with constructs dedicated to context. The objective of ContextUML is to boost the formulation of context requirements at an earlier stage of the life cycle development of Web services. ContextUML permits first, to explicitly integrate context into the specification metamodels of Web services (e.g., WSDL), and second, to facilitate the mapping of these contextual metamodels onto concrete languages to be enhanced with context constructs (Section 4.4).

Fig. 2 presents the metamodel diagram of the abstract syntax of ContextUML. The left part of the diagram presents a subject that is aware of its surrounding environment through context. **Subject** is

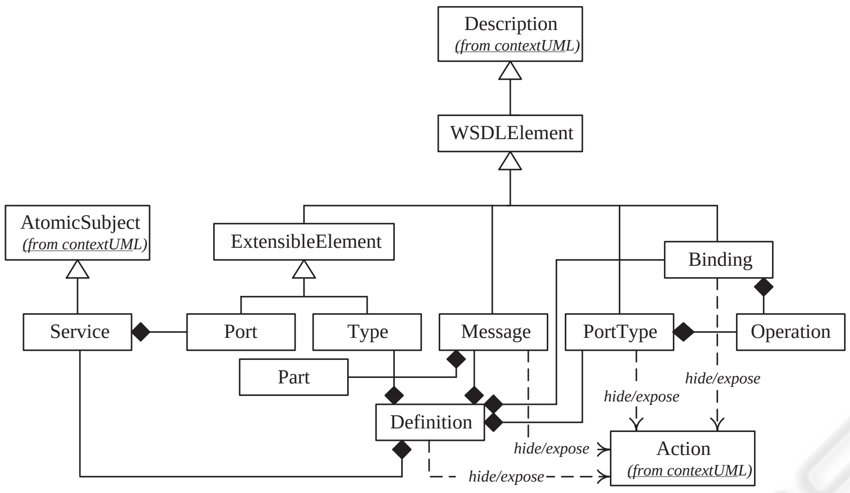


Figure 3: Integration of ContextUML metamodel into WSDL metamodel.

an element of type service, which changes its behavior (e.g., operation interface, conversation protocol) according to the changes reported in the context. Independently of the specialization of Subject into **CompositeSubject** (i.e., composite Web service) and **AtomicSubject** (i.e., Web service), Subject binds to **Context**. Context tracks and assesses the changes in the environment of a subject. These changes are multiple ranging from time and location to access privileges and resource scheduling.

Still in Fig. 2, the right part of the diagram highlights the visibility mechanisms. **Visibility** is the capacity of a context-aware subject to adapt its description because of a given context. This adaptation happens with the use of **Action** that corresponds to the operations to perform over the description of a subject. The rationale of this performance is to expose/hide some elements of a subject to/from other subjects. An example of process visibility is discussed in (Baïna et al., 2005). Finally, **Description** corresponds to the specification of a subject for example in WSDL. It should be noted that Visibility is framed using **VisibilityConstraint**, which permits achieving the consistency of the description of a subject after adaptation.

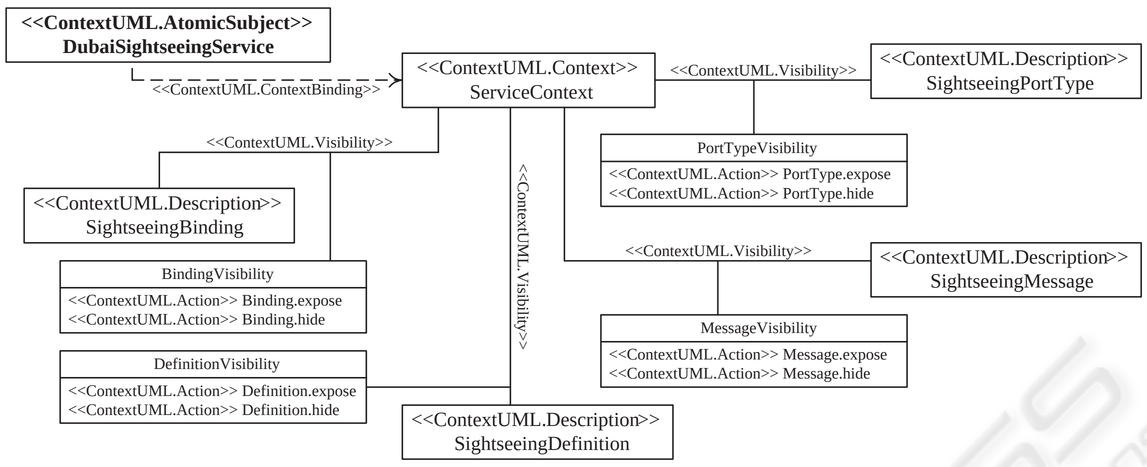
The **VisibilityConstraint** represents the framework through which the *expose* and *hide* operations are implemented. As part of our work we will be developing a Context Constraint Language (CCL), as an enhancement of UML's OCL, for specifying visibility constraints. In particular, we will be enhancing OCL for specifying pre- and post-conditions to run over a description of type WSDL specification.

According to (Srivastava and Koehler, 2003), the functionality of a Web service needs to be described with additional pieces of information including semantic annotation of what it does and functional annotation of how it behaves. The contextual definition of a Web service enables the adaptation of semantic and functional annotations to be adaptable according to the context wherein this Web service evolves, i.e., the different compositions in which the Web service participates. Hence the first step of developing a contextual WSDL is to extend the abstract syntax of WSDL with the vocabulary of ContextUML. In Fig. 3, the extension of WSDL is represented and occurs by importing constructs from the ContextUML metamodel (Fig. 2). Thus, WSDL will now include more constructs such as visibility and context.

After a thorough analysis of WSDL metamodel, we established a list of elements to be subject to contextualization in a WSDL specification: **Port**, **Type**, **Message**, **Binding**, and **PortType**. These elements are all derived from Description of ContextUML metamodel. Using Visibility, *expose/hide* actions are executed over each of these elements. More details on the execution of these actions are given in Section 4.4. Besides Port, Type, Message, Binding, and PortType elements, additional elements are added because the semantics of ContextUML states that the contextualized of a description means the contextualization of it constituents (e.g., **Part** with regard to Message, and **Operation** with regard to PortType and Binding). In Fig. 3, it is also shown that Service is derived from AtomicSubject of the ContextUML metamodel.

4.4 Concrete Syntax of ContextUML

Using the running scenario of Section 3, we now illustrate the concrete syntax of Contex-



<<ContextUML.AtomicSubject>> DubaiSightseeingService
<<ContextUML.ContextBinding>>
<<ContextUML.Context>> ServiceContext
<<ContextUML.Visibility>>
<<ContextUML.Description>> SightseeingPortType
PortTypeVisibility
<<ContextUML.Action>> PortType.expose
<<ContextUML.Action>> PortType.hide
<<ContextUML.Visibility>>
<<ContextUML.Description>> SightseeingBinding
BindingVisibility
<<ContextUML.Action>> Binding.expose
<<ContextUML.Action>> Binding.hide
<<ContextUML.Visibility>>
<<ContextUML.Visibility>>
<<ContextUML.Description>> SightseeingMessage
MessageVisibility
<<ContextUML.Action>> Message.expose
<<ContextUML.Action>> Message.hide
DefinitionVisibility
<<ContextUML.Action>> Definition.expose
<<ContextUML.Action>> Definition.hide
<<ContextUML.Description>> SightseeingDefinition

Figure 4: Concrete description of a Web service using contextual WSDL metamodel.

tUML in Fig. 4. Each Web service of the scenario like sightseeing and transportation is represented by an UML class with the stereotype <<**ContextUML.AtomicSubject**>>. In addition the context of a Web service is defined as an UML class with the stereotype <<**ContextUML.Context**>>. The connection between Web service and context is illustrated using a dependency and the stereotype <<**ContextUML.ContextBinding**>>.

The rest of Fig. 4 specifies a visibility situation that involves the description of a Web service. We focus on Port, Type, Message, Binding, and PortType constituents of this description. A visibility along with its relation to action and description (<<**ActionAssignment**>> and <<**ActionDescription**>>, respectively as reported in Fig. 2) is defined with a single UML association that has the stereotype <<**ContextUML.Description**>>. This association connects context to these five constituents subject to contextualization. Each constituent is represented as an UML class with the stereotype <<**ContextUML.Description**>>. For example the message constituent of a WSDL specification of sightseeing Web service is represented as UML class denoted by SightseeingMessage and the stereotype <<**ContextUML.Description**>>. In addition each attribute of the association represents the actions that could be carried out over a description, in that case a WSDL specification. An action is identified by the name of a WSDL constituent and its nature whether expose or hide. It should be noted at this point of time constraints through **VisibilityConstraint** are not yet integrated into Fig. 4.

5 CONCLUSION

In this paper we presented our contextual model-driven approach for Web services with emphasis on their definition. The features of the environment in which Web services operate were modelled using context. Our aim is to develop contextual specifications of Web Services using ContextUML an extension of UML using UMLProfile. Future work will focus on the application of our approach to modeling the control and data flow of the Web services workflows, including exception handling and compensation. We will also be focusing on developing the context constraint language.

ACKNOWLEDGMENTS

The authors wish to thank R. Hauser from IBM Research Lab in Zurich, Switzerland for his feedback. The fourth author wishes to thank his manager, K. Muralidharan, for supporting his work. Other company (i.e., non-IBM), product and service names may be trademarks or service marks of others.

REFERENCES

Amsden, J., Gardner, T., Griffin, C., and Iyengar, S. (2003). Draft UML 1.4 Profile for Automated Business Processes with a Mapping to BPEL 1.0. Technical report, IBM. http://www-128.ibm.com/developerworks/rational/library/4593.html.

Arara, A. and Benslimane, D. (2004). Towards Formal Ontologies Requirements with Multiple Perspectives. In *Proceedings of The 6th International Conference* *on Flexible Query Answering Systems (FQAS'2004)*, Lyon, France.

Backx, N. (2004). Model-Driven Architecture for Web Services Applications. Technical report, University of Twente, The Netherlands, and Universidade Federal do Espírito Santo, Brazil.

Baïna, K., Benali, K., and Godart, C. (2005). DISCOBOLE: A Service Architecture for Interconnecting Workflow Processes. *Computers in Industry - Special issue Concurrent Engineering (CE'03), Elsevier Science Publisher*.

Baïna, K., Benatallah, B., Casati, F., and Toumani, F. (2004). Model-Driven Web Service Development. In *Proceedings of The 16th Conference On Advanced Information Systems Engineering (CAiSE'2004)*, Riga, Latvia.

Bézivin, J., Hammoudi, S., Lopes, D., and Jouault, F. (2004). Applying MDA Approach for Web Service Platform. In *Proceedings of The 8th International Enterprise Distributed Object Computing Conference (EDOC'2004)*, Monterey, CA, USA.

Birman, K. P. (2004). Like It or Not, Web Services Are Distributed Objects. *Communications of the ACM*, 47(12).

Bonnect, P. (2004). Offshore + MDA + SOA, Vers L'ingénierie des Modèles. Technical report. http://www.orchestranetworks.com/.

Bordbar, B. and Staikopoulos, A. (2004). On Behavioural Model Transformation in Web Services. In *Proceedings of The 5th International Workshop on Conceptual Modeling Approaches for e-Business (eCOMO'2004) held in conjunction with The 23rd International Conference on Conceptual Modeling (ER'2004)*, Shanghai, China.

Gottlob, G., Schrefl, M., and Rck, B. (1996). Extending Object-oriented Systems with Roles. *ACM Transactions on Information systems*, 14(3).

Grϕnmo, R., Skogan, D., Solheim, I., and Oldevik, J. (2004). Model-Driven Web Services Development. In *Proceedings of The 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'2004)*, Taipei, Taiwan.

Grϕnmo, R. and Solheim, I. (2004). Towards Modeling Web Service Composition in UML. In *Proceedings of The 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WS-MAI'2004) held in conjunction with The 6th International Conference on Enterprise Information Systems (ICEIS'2004)*, Porto, Portugal.

Huhns, M. (2002). Agents as Web Services. *IEEE Internet Computing*, 6(4).

Kulkarni, N., Kumar, S., Mani, S., and Padmanabhuni, S. (2005). Web Services: E-Commerce Partner Integration. *IEEE IT Professional*, 7(2).

Lodderstedt, T., Basin, D., and Doser, J. (2002). SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *Proceedings of The 5th International Conference on the Unified Modeling Language - The Language and its Applications (UML'2002)*, Dresden, Germany.

Ma, K. L. (2005). Web Services: What's Real and What's Not. *IEEE IT Professional*, 7(2).

Maamar, Z., Benslimane, D., and Narendra, N. C. (2006a). What Can Context do for Web Services? *Communications of the ACM*. (to appear).

Maamar, Z., Kouadri Mostéfaoui, S., and Yahyaoui, H. (2005). Towards an Agent-based and Context-oriented Approach for Web Services Composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(5).

Maamar, Z., Narendra, N. C., and Sattanathan, S. (2006b). Towards an Ontology-based Approach for Specifying and Securing Web Services. *Journal of Information & Software Technology, Elsevier Science Publisher*. (to appear).

Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Introduction to the Special Issue on Model-Driven Development. *IEEE Software*, 20(5).

Miller, J. and Mukerji, J. (2001). Model Driven Architecture. Technical Report ormsc/2001-07-01, Object Management Group. www.omg.com/mda.

Pernici, B. (1990). Objects with Roles. In *Proceedings of The ACM Conference on Office Information Systems*, Cambridge, Massachusetts, US.

Sendall, S. and Kozaczynski, W. (2003). Model Transformation: The Heart and Soul of Model-Model-Driven Software Development. *IEEE Software*, 20(5).

Sheng, Q. Z. and Benatallah, B. (2005). ContextUML: A UML-based Modeling Language for Model-driven and Context-Aware Web Services. In *Proceedings of The 4th International Conference on Mobile Business (ICMB'2005)*, Sydney, Australia.

Spaccapietra, S., Parent, C., and Vangenot, C. (2000). GIS Databases: From Multiscale to MultiRepresentation. In *Proceedings of The 4th International Symposium Abstraction, Reformulation, and Approximation (SARA'2004)*, Horseshoe Bay, Texas, USA.

Srivastava, B. and Koehler, J. (2003). Web Service Composition - Current Solutions and Open Problems. In *Proceedings of The Workshop on Planning for Web Services held in conjunction with The 13th International Conference on Automated Planning & Scheduling (ICAPS'2003)*, Trento, Italy.

Zarras, A., Vassiliadis, P., and Issarny, V. (2004). Model-Driven Dependability Analysis of Web Services. In *Proceedings of The 6th International Symposium on Distributed Objects and Applications (DOA'2004)*, Larnaca, Cyprus.