Journal of Software Engineering and Applications, 2012, 5, 711-721
http://dx.doi.org/10.4236/jsea.2012.59084 Published Online September 2012 (http://www.SciRP.org/journal/jsea)

Towards an UML Profile for Web Service Composition Based on Behavioral Descriptions

Ayoub Sabraoui[1], Ahmed Ettalbi[1], Mohammed El Koutbi[1], Abdeslam En-Nouaary[2]

[1]Mobile Intelligent Systems Team (MIS), Ecole Nationale Supérieure d'Informatique et d'Nalyse des Systèmes (ENSIAS), Rabat, Morocco; [2]Electrical and Computer Engineering Department, Concordia University, Montreal, Canada.
Email: a.sabraoui@uca.ma, {ettalbi,elkoutbi}@ensias.ma, ennouaar@ece.concordia.ca

Received July 8th, 2012; revised August 11th, 2012; accepted August 23rd, 2012

ABSTRACT

Web service composition is one of the challenging issues that have been investigated over the past decade. It consists of combining and reusing existing Web services to best suit new user requirements. This paper proposes an UML profile to compose Web services based on their behavioral aspects. To do so, the web service WSDL files are first transformed to UML models; then the profile is used to integrate them; finally the MDA approach is adopted to transform the applied profile into a BPEL process. As such, our method has the advantages of being independent of the Web service composition language and the UML modeling tool. Finally, a case study is developed in order to show the benefits of our method.

Keywords: Web Service Composition; UML Profile; MDA; BPEL

1. Introduction

Nowadays, distributed applications are increasingly being developed in the context of Service Oriented Architecture (SOA), where the basic unit of computation is called a service. The Web service technology remains the standard means for building enterprise applications because it provides mechanisms that facilitates the interoperability between different software applications, and their executions regardless of the underlying platforms and/or frameworks [1]. According to W3C [2], a Web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network. The technology basically articulates around the following three components:

- Simple Object Access Protocol (SOAP) provides the definition of an XML document, which can be used for exchanging structured and typed information between service peers in a decentralized distributed environment [3].
- Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to concrete network protocols and message formats to define an endpoint [4].
- Universal Description, Discovery and Integration (UDDI) focuses on the definition of a set of services supporting the description and discovery of the Web services available for clients, and the technical interfaces, which may be used to access those services [5].

As such, the Web service technology proposes a high level of abstraction, which replaces the current models of applications by a more modular and flexible architecture, thus allowing their composition and their integration. All these features help the enterprises overcome the difficulties related to the cost and flexibility of the offered solutions for the electronic exchange of information. Moreover, Web services have functional, non-functional, behavioural, and semantic characteristics. The functionality of Web services is described using interfaces with input and output parameters. The quality of services like performance is described by the non-functional specification usually given as cost, response time, availability, security, reliability, and reputation. The behaviour states, how to interact with the Web services, in terms of sequences of input/output interactions, for instance. Web service semantics describe the meaning of the services generally through the usage of ontology. The description of Web services exposes the main aspects that enable them to be published, found, and used by other Web services. They are also the key elements in composing the Web services into new ones [6].

The composition of Web services (WSC for short) is one of the key features advanced by the technology. It consists of combining existing services to provide a

richer new composite service to meet some user requirements. Such composition requires methods and languages for basic Web services integration, such as: XML, XLANG, BPML, WSFL, WSCL, WSCI, BPEL4WS and WS-CDL. The composition techniques can be classified into two categories, namely: static service composition and dynamic service composition. Static composition allows the requestor to create an abstract model that should be respected during the execution of the composed Web service. While the dynamic composition enables selecting the atomic Web services automatically and combines them to create an unlimited number of new Web services.

Another way to classify WSC is with regard to the degree of automation (manual, semi-automatic and automatic). Manual composition is completely performed by a human, semi-automatic composition is carried out with human assistance, and automatic composition takes place without any human involvement [6].

There is yet a third classification, which is based on how the composition is specified, either by orchestration or by choreography [7]. Orchestration consists of combining available services by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single sub-activities, as illustrated in **Figure 1**.

Choreography, however, defines complex tasks via the definition of the conversation that should be undertaken by each participant, as shown in **Figure 2**.

This paper proposes a new method for composing Web services by defining a "WSComposition profile" which is an extension of the UML2.2 sequence diagrams metamodel. The proposed method applies the MDA [8] to generate the code of the composite Web service in Business Process Execution Language (BPEL) [9]. Therefore, this profile allows us to represent the behavioral characteristics of Web services, and provides an easy way to design and compose Web services based on their behavioral aspect. Basically, our approach consists of three main steps. In the first step, existing simple Web services are discovered and located in the Web services registry. The developer imports the WSDL files of the candidate services and translates them into UML diagrams (class diagram and use case diagram). In the second step, the modeling of the composite Web services is allowed by our WSComposition profile. In the third step, the MDA approach is adopted to automatically transform the applied profile into a BPEL process. The MDA transformation rules are expressed in Atlas Transformation Language (ATL) [10].

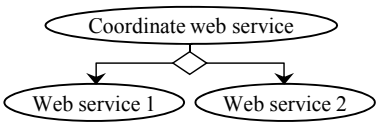

Figure 1. Web services orchestration.

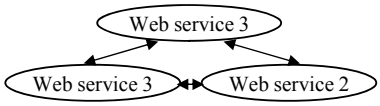

Figure 2. Web services choreography.

Among the composition languages, we chose BPEL (or BPEL4WS) because it is an XML-based language and describes the business process interactions based on Web services, within and between enterprises. A BPEL process specifies the exact order in which participating Web services should be invoked. This can be done sequentially or in parallel. With BPEL, we can express conditional behavior; for example, a Web service invocation can depend on the value of a previous invocation. We can also construct loops, declare variables, copy and assign values, define fault handlers, and so on. By combining all these constructs, we can define complex business processes in an algorithmic manner [11].

Compared to existing WSC approaches, our method has the advantages of being independent of both the WSC language and the UML modeling tool; the user can implement our WSComposition profile in any UML design tool, choose any language of composition and any execution engine to compose services.

Our method offers also a semi-automatic way to compose Web services by assisting the user in the design stage. It allows the user to maintain some control over the process with no need for programming knowledge. In other words, the user can define the modeling and design process in a graphical way through the WSComposition profile in order to compose Web services.

The remainder of this paper is structured as follows: Section 2 gives an overview of the fundamental aspects on which our proposal is based. Section 3 describes the WSComposition profile. Sections 4 and 5 present the key ideas of our approach for WSC. Section 6 gives a case study for the application of our method. Section 7 is devoted to related work. Finally, Section 8 concludes the paper and presents future work.

2. Our UML Profile for Web Services Composition

Our aim of designing an UML profile for WSC, is to provide a standard means for expressing the semantics of WSC using UML2.2 notation and thus to support expressing these semantics with UML tools. In addition, the profile allows a mechanism to model a WSC independently of the underlying platforms. Basically, our UML profile relies on two main levels of abstraction: At the higher level we have the metamodel of UML2.2 se-

quence diagrams that defines the basic entities of the model, whereas at the second level, we have the WS-Composition UML profile which adapts the previous metamodel for WSC.

2.1. The Metamodel of UML2.2 Sequence Diagrams

Here, we present the sequence diagrams metamodel which constitutes a type of the interaction package of UML2.2 metamodel. To define our profile we use the Lifeline, Message and Combined Fragment classes as defined in OMG UML2.2 [12]: A lifeline is a Named Element that represents an individual participant in the Interaction. While Parts and Structural Features may have multiplicity greater than 1, Lifelines represent only one interacting entity. Message, however, is a Named Element that defines one specific kind of communication between lifelines of an interaction. The message specifies not only the kind of communication, but also the sender and the receiver. Sender and receiver are normally two occurrence specifications (points at the ends of messages). Finally, combined fragment is an interaction fragment which defines a combination (expression) of interaction fragments. A combined fragment is defined by an interaction operator and corresponding interaction operands. Through the use of combined fragments the user will be able to describe a number of traces in a compact and concise manner.

2.2. WSComposition UML Profile

This section presents the WSComposition UML Profile which is a group of extensions of UML2.2 sequence diagrams metamodel. We present some elements that we extend and specialize to define our Profile, as shown in **Figure 3**. On the one hand, we extend the Lifeline class by two stereotypes: "Actor" and "WebService". The latter stereotype is specialized by the "CordinateWS" one, which represents the business process.

On the other hand and in order to manage the messages exchanged between client and services or those invoked from other services, we extend the Message class in three stereotypes: Receive, Reply and Invoke.

"Receive" element allows business process to wait for a matching message to arrive. This stereotype has a filled arrow head.

"Reply" element allows the business process to send a message in reply to a message that was received by "receive" stereotype. It has a dashed line with open arrow head.

"Invoke" element allows the business process to invoke a one-way or request-response operation on a port-type offered by a partner. In the request-response case, the invoke activity completes when the response is received. This stereotype has an open arrow head.

Finally and for managing the workflow of the business process, we extend the Combined Fragment class in the "Control Flow" stereotype, which contains an enumeration designating the different kinds of control flow elements, like sequential, parallel, alternative and iterative activities.

Beside the aforementioned steretypes, our UML profile uses the enumerated type Activity Element Kind defines a set of available activity types. They are presented as follows:

Sequential activity: defines a list of activities to be performed sequentially in lexical order.

Parallel activity: specifies one or more activities to be performed concurrently.

Alternative activity: is used to select one activity for execution from a set of choices.

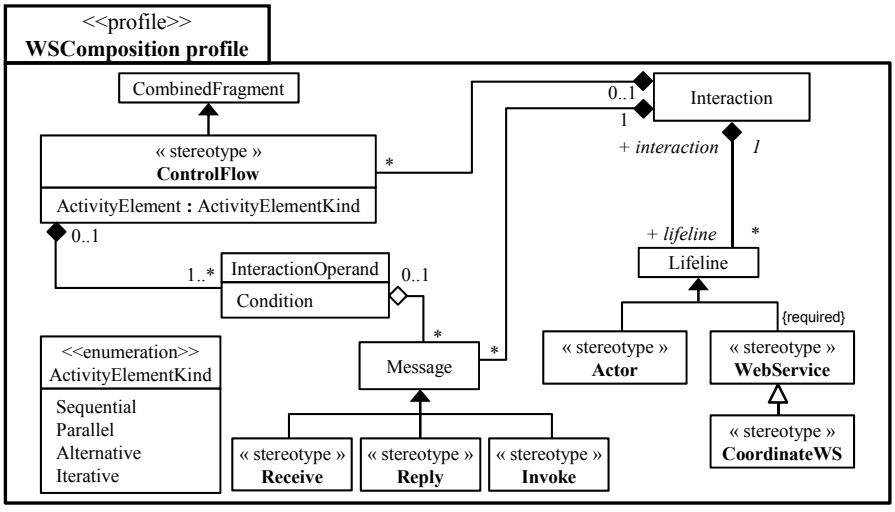

Figure 3. Excerpt of the WSComposition UML profile.

Iterative activity: is used to define that the child activity is executed while the condition evaluates to true.

3. Our New Method for WSC

This section presents the key ideas of our method to compose Web services. In our approach, interactions with a services are modeled as scenarios, thus the composition of web services is none other than the composition of those scenarios. Knowing that the scenario describes the behavior of a system, our approach is based on the behavioral characteristics of web services to compose them. For this reason, we propose two graphical ways:

3.1. Composing Scenarios Using Operators

In [13], we defined four operators, namely: the sequential, the alternative, the iterative and the concurrent operators. The idea consists of composing scenarios that models partial behaviors using these operators, in order to obtain one scenario that represents a global behavior of the system. **Figure 4** shows the graphical application that we developed to compose scenarios [13].

In this example, we compose three scenarios S1, S2 and S3 as following:

par**{S1;or{S2;S3;Price>1000}}**

This means S2 and S3 are composed alternatively and the result is executed concurrently with S1.

3.2. Composing Scenarios Using the WSComposition UML Profile

Our WSComposition profile **Figure 3** extends the UML 2.2 sequence diagrams metamodel and customizes it for specific requirement of web services composition. WSComposition profile presents a multiple operators to model the order of execution of scenarios (interactions with services). This is the method adopted in this paper and which we explain in more details in the next sections.

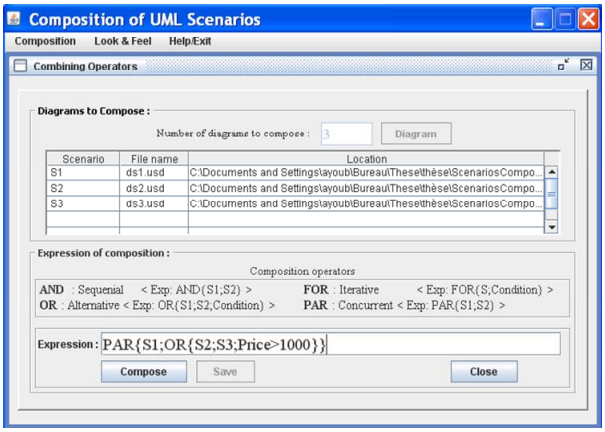

Figure 4. Graphical interface for scenarios composition.

The resulting composite scenario from the application of each of the two ways represents the composite Web service which satisfies the client needs.

The interaction between the user and any requested service can be clearly represented by sequence diagrams (scenarios) in UML. Consequently, the services composition amounts composing these scenarios to meet the client needs. Thus, the idea is to model the WSC by WSComposition profile which is an extension of the UML2.2 sequence diagram metamodel and to adopt the MDA approach to automate the generation of BPEL process execution code corresponding to the resulting composite Web service. Our method is independent of the WSC language and the UML modeling tool. Thus, the user can choose the UML design tool, the language of composition and the execution engine to make his/her composition. In the current version, our method supports the BPEL language and the EclipseUML tool for UML design. **Figure 5** presents the various steps of our composition method.

1) ***Searching for candidate Web services*****:** the Web services are searched and located in the Web service registry. The developer imports the service description represented in WSDL and translates it into UML diagrams: classes diagram, use case diagram and sequence diagrams.

2) ***Composition of Web services*****:** in this step the designer use the WSComposition profile to manage the workflow of the business process. The interaction between these services and their execution order are modeled by the interaction operators defined in this profile.

3) ***Adopting the resulting scenario to the WSComposition UML Profile*****:** the resulting scenario created in the previous step is in XMI format. This file does not comply with the WSComposition profile that we will use in the next step to transform models (WSComposition2BPEL) according to MDA approach.

Therefore, the purpose of this step is to transform this XMI file to another one according to our profile. The advantage of this first transformation is to allow the use

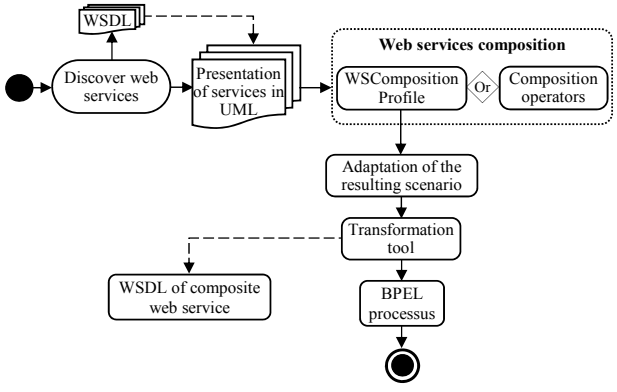

Figure 5. Steps of our WSC method.

of other UML modeling tools than EclipseUML; and than make the application independent of the used tool. This transformation is done by the JDOM API [14].

4) ***Transforming the applied profile into a BPEL process*****:** in this stage, we adopt the MDA approach to transform the applied profile into a BPEL process representing the composite Web service. This transformation requires the source metamodel (WSComposition profile) and the target metamodel (BPEL metamodel), the applied profile, and the transformation rules which can be expressed by one of the following transformation languages: OCL [15], YATL [16,17], ATL [10], etc. In our approach, we choose the ATL language because it is easy to use, close to the standards and there is a rich set of tools that have been built for ATL development under Eclipse.

4. Model Transformation

Mapping specification and defining transformations in MDA is not an easy task. Authors in [18], proposed a taxonomy of model transformation, based on the discussions of a working group on model transformation of the Dagstuhl seminar on Language Engineering for Model-Driven Software Development. This taxonomy can be used, among others, to help developers in deciding which model transformation language or tool is best suited to carry out a particular model transformation activity.

In this section, we present our proposition for the specified transformation in the context of MDA. According to this approach, the source and target metamodels are based on a common metamodel MOF or EMF Ecore [19]. The source metamodel is the WSComposition profile, and the target one is the BPEL metamodel. This mapping is implemented as a tool on EclipseUML IDE and the rules of transformation are defined using the ATL language.

In order to enhance automatic mapping, we define the rules for transforming the WSComposition profile into BPEL process as follow:

Rule**1 (*process and partners*):** the CordinateWS element constitutes the process element in BPEL process. Actor and WebService elements are mapped to partnerLinks elements which present the different parties that interact with the business process.

WSComposition Profile Element	BPEL Process Element	Transformation
"CordinateWS"	<process>	<process name="..." targetNamespace="..."xmlns="..." ... </process>
"Actor" and "WebService"	<partnerlinks>	<partnerLinks> <partnerLink name="…" partnerLinkType="…" partnerRole="..."myRole=".."/> </partnerLinks>

***Rule* 2 (*variable*):** Variable element defines the data used by the exchanged message. It is mapped into variable element of BPEL process.

WSComposition Profile Element	BPEL Process Element	Transformation
"Variable"	<variables> <variable/> ... </variables>	<variables> <variable name="..." element="..." </variable> </variables>

***Rule* 3 (*exchanged messages*):** messages exchanged between client and services or those invoked from other services are mapped into receive, reply, or invoke elements of BPEL process.

WSComposition Profile Element	BPEL Process Element	Transformation
"Receive"	<receive>	<receive partnerLink="…" portType="QName" operation="…" variable="…" … </receive>
"Reply"	<reply>	<reply … </reply>
"Invoke"	<invoke>	<invoke … </invoke>

In addition to these basis elements, we have defined another five rules to transform the control flow into BPEL elements. The details of these rules are given in the following:

***Rule* 4 (*sequential activity*):** Sequential activity defines a list of activities to be performed sequentially in lexical order, this element is mapped into sequence element of BPEL process.

WSComposition Profile Element	BPEL Process Element	Transformation
"Sequential"	<sequence>	<sequence> <activity1/> <activity2/> … </sequence>

***Rule* 5 (*parallel activity*):** Parallel activity is used to specify one or more activities to be performed concurrently, this element is mapped into flow element of BPEL process.

WSComposition Profile Element	BPEL Process Element	Transformation
"Parallel"	<flow>	<flow> <activity1/> <activity2/> … </flow>

***Rule* 6 (*alternative activity*):** Alternative activity is used to select one activity for execution from a set of choices. This element is mapped into if element of BPEL process.

WSComposition Profile Element	BPEL Process Element	Transformation
"Alternative"	<if>	<if condition="…"> activity <elseif condition="…"> activity … </while>

***Rule* 7 (*iterative activity*):** Iterative activity is used to define that the child activity is executed while the condition evaluates to true. The Iterative element is mapped into while element of BPEL process.

WSComposition Profile Element	BPEL Process Element	Transformation
"Iterative"	<while>	<while condition="…"> activity … </while>

These transformation rules permit the mapping between two metamodels, which are the WSComposition profile and the BPEL metamodel. The code in ATL is generated based on this model transformation can help the designer to specify how the metamodels are inter-related. This model transformation can also help us to define a bidirectional mapping, which will be investigated in more details in future papers.

5. Case Study

In this section, we are going to illustrate the application of our method through a case study. Suppose that a travel agency proposes to its customers organized trips adapted to their needs by composing appropriate existing Web services. The proposed tours contain the transport, the hosting and the car rent. When the customer decides to reserve for a tour, he/she can make several choices to get one of them. To this end, the customer gets access to the web site of the travel agency which proposes tickets of plan and train, the rental of cars and the hotel reservation. **Figure 6** presents the business process behavior, which gathers the various tasks to be achieved by the composition.

Each reservation is related to a Web service. Services run in this order with respect to the following conditions:

Flight reservation (WS1), if not train reservation (WS2).

Hotel reservation (WS3).

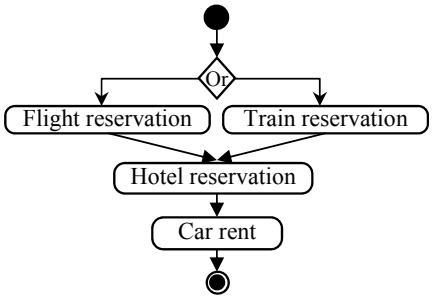

Figure 6. Business process of the tourist tour reservation.

Car rent (WS4).

1) *The first step of our approach*: It consists of the manual selection of the candidate Web services involved in this composition. These services are represented as UML diagrams (class diagram, use case diagram and sequence diagrams) based on their WSDL files. This allows the requirements capture [20] and the description of the different interactions.

Figure 7 shows these Web services in terms of class diagrams. Each diagram represents the Web service with the operation signature invoked in our example:

The use case diagram in **Figure 8** describes the interactions between the user and the various services.

The client expresses his choices through the agency travel system. Then, the agency system sends them to the airline company, railways company, car rent company and hotel reservation systems. Then, it receives the different responses from its partners and sends them back to the user. Afterwards, the user makes his choice and completes his reservation.

From the use case diagram above, we can instantiate some scenarios that are useful for our study. The sequence diagram in **Figure 9** illustrates the different interactions to book airline ticket.

These interactions remain valid for the other bookings. For the sake of simplicity, we suppose that the reservation is made as shown in **Figure 10**.

2) *The second step of our approach*: The sequence diagrams from the previous step give us an overview to compose them using WSComposition profile, in order to obtain a single diagram which represents the desired composite Web service.

We return to our example to define the different scenarios of book in the following order and condition:

Booking an airline ticket (S1), if not a train ticket (S2) ➔ "alt" operator.

Booking hotel (S3) and car rental (S4) in parallel ➔ "par" operator.

We merge these scenarios to produce the resulting sequence diagram (Sr) of this composition, which represents the global behavior of the business process.

Then: **Sr = alt(S1,S2)+par(S3,S4)**

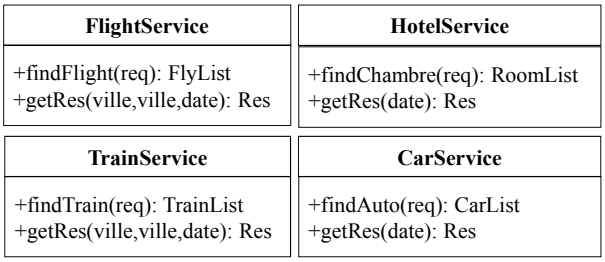

Figure 7. Class diagram of the travel agency.

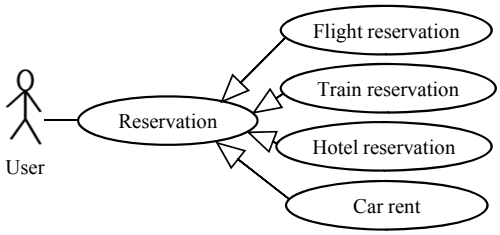

Figure 8. Use case diagram of the travel agency.

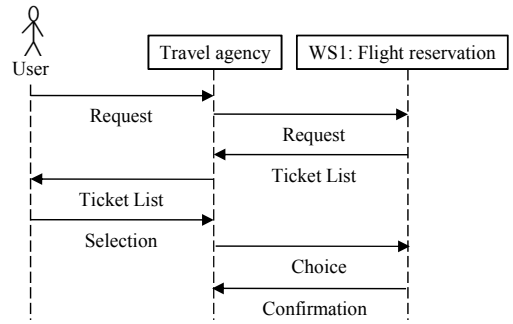

Figure 9. Sequence diagram to book airline ticket.

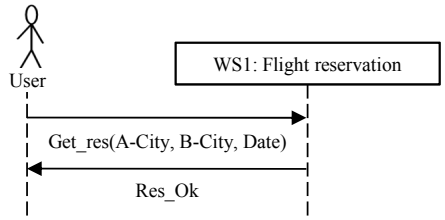

Figure 10. Simplified sequence diagram.

which mean that we compose S1 and S2 alternately, S3 and S4 in parallel and the two results will be composed sequentially.

Figure 11 shows the sequence diagram resulting from this composition.

3) ***The third step of our approach*****:** The UML modeling tool used is the EclipseUML IDE, which permits to create the different diagrams. These diagrams are in XMI format. These files does not comply with WSComposition profile that we will use in the next step to transform models (WSComposition2BPEL) according to MDA approach. Therefore, the purpose of this step is to transform this XMI file to another one according to our profile. The advantage of this first transformation is to allow the use of other UML modeling tools than EclipseUML; and then make the application independent of the used tool.

4) ***The fourth step of our approach*****:** In this step, we adopt the MDA approach for transforming the applied profile into BPEL process, which represents the composite Web service. The transformation definition must be based on the correspondence between the WSComposition profile and BPEL models. The rules of transformation are defined in the ATL language. The transformation process is illustrated in **Figure 12**.

In MDA, a metamodel defines the language and process from which to form a created model, and all metamodels are based on a common metamodel MOF or EMF Ecore.

In this transformation we use the source—WSCompo-

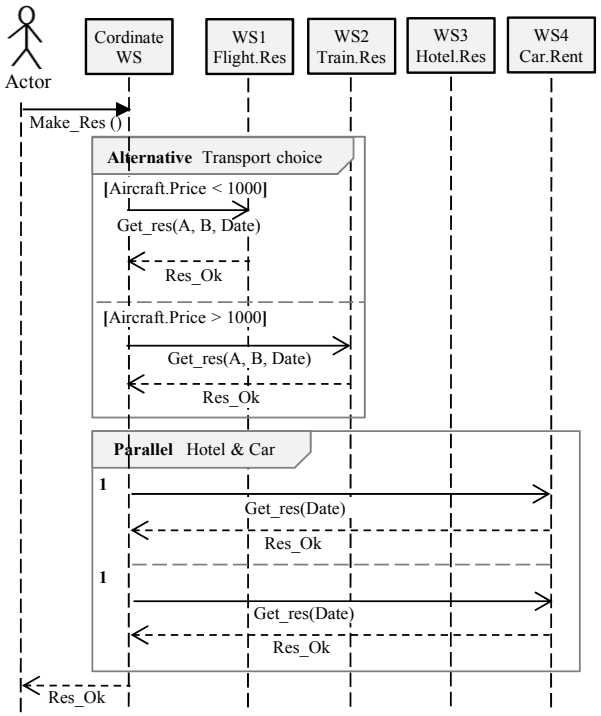

Figure 11. The applied profile resulting from composition.

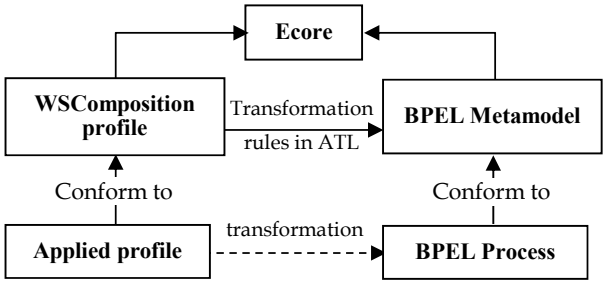

Figure 12. Transformation process according the MDA approach.

sition profile- **Figure 3** and the target—BPEL metamodel-implemented in Ecore **Figure 13**.

The transformation rules from the applied profile to BPEL process are illustrated in **Figure 14**.

The CordinateWS element represents the BPEL process. Actor and the four Web services are transformed into "partnerLink" elements.

The Alternative and Parallel control flow elements of profile are transformed into "if" and "flow" activities of BPEL process respectively. Both activities are included in the main activity process named "sequence".

The Alternative element transformed into "if" activity contains two blocks: the first is transformed into "if" and its condition, the second into "else if" and his condition.

The Parallel element is also composed of two blocks which are transformed into two "sequence" activities included in the "flow" one.

Finally, the different messages exchanged between CordinateWS, Actor and Web services are transformed into "receive", "relpy" and "invoke" activities depending on the appropriate context.

6. Related Work

The problem of combining multiple web services to satisfy a single task, has received much attention recently. Several approaches are proposed to compose Web services. In this section, we discuss some of the existing work that is most relevant to our approach.

Our work is most similar to the approaches presented by [21,22], in which they describe a semi-automatic process for WSC based on the behavioral aspect. They propose a method that uses UML Activity models to design Web service, and MDA to generate executable specifications in different composition languages. They propose a metamodels for the target models, and present case studies to prove the applicability of their methods to compose Web services. Unlike our method which uses UML sequence diagrams to design WSC assuming that is close to the services concept. According to the behavioral aspect, authors in [23], have formally defined the realistic model for behavioral description, and studied the computational complexity of four variations of WSC problems: 1) solving the composition problem of deterministic Web services for a restricted case (when the coordinator Web service has complete information about the states of all Web services) is PSPACE-complete; 2) solving the composition problem of deterministic Web services for a general case (when the coordinator Web

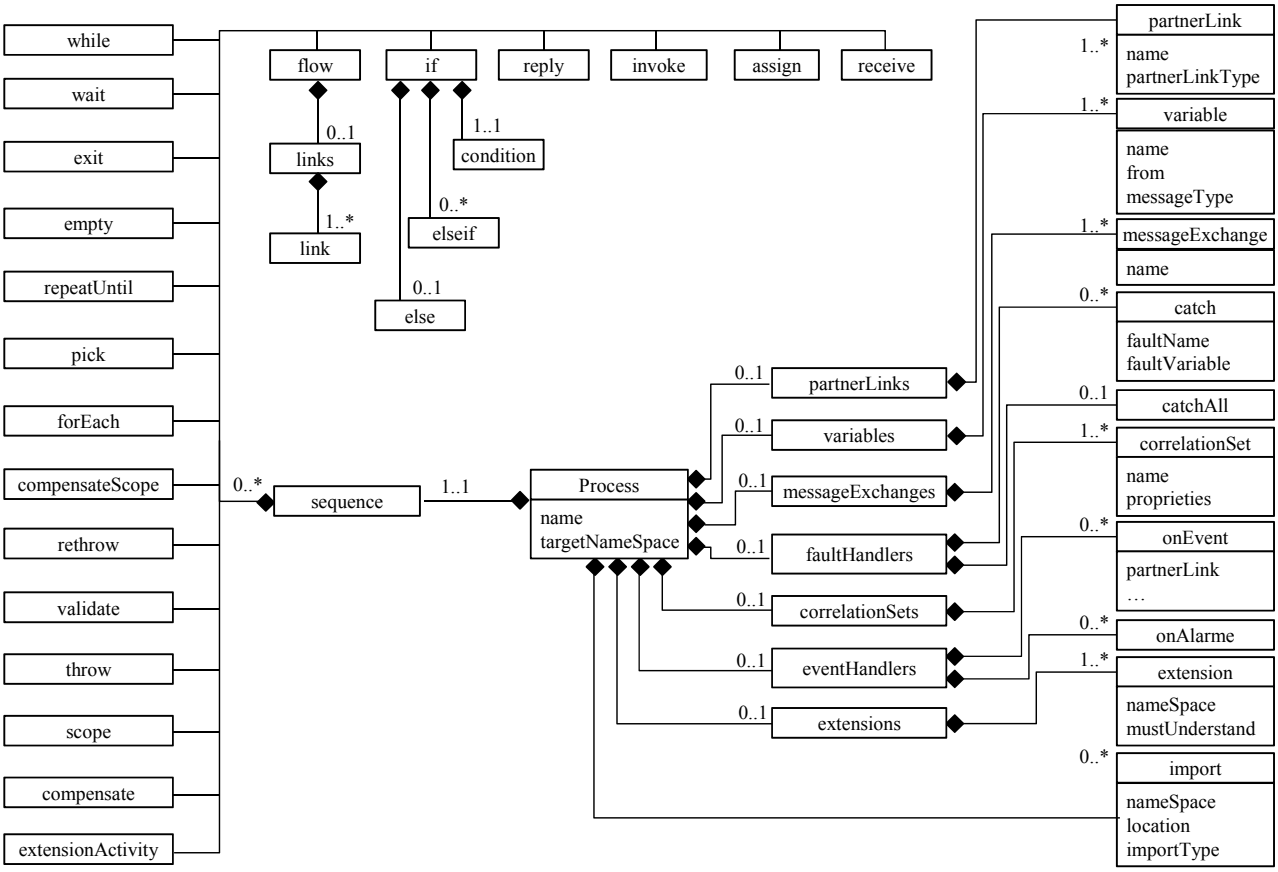

Figure 13. A metamodel of the BPEL 2.0 process.

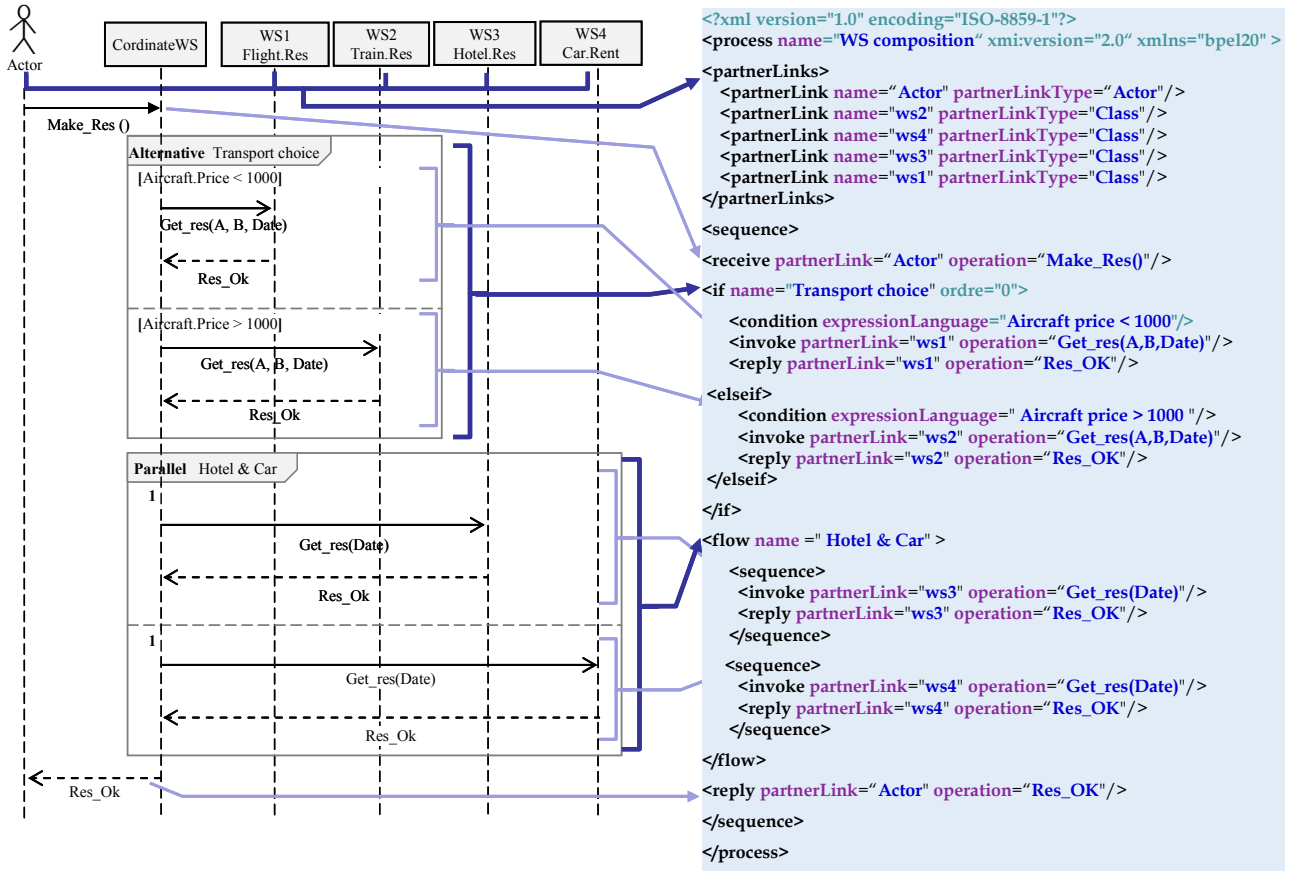

Figure 14. Transformation of the applied profile into BPEL process.

service has incomplete information about the states of Web services) is EXPSPACE-complete; 3) solving the composition problem of non-deterministic Web services on complete information is EXP-complete and 4) solving the composition problem of nondeterministic Web services on incomplete information (which is the most general case) is 2-EXP-complete.

There is another stream of work [24,25] which considers non-functional aspect like the quality-of-service (QoS) and the optimization of the WSC problem. In this paper, QoS parameters are not taken into account in the process for composing Web services. In another study, we can take into consideration the user needs in terms of QoS at the design stage, by extending our WSComposition profile.

The approach in [26] is different from the other existing work and this paper. It proposes a natural language interface to Web services, which can be used even by a novice user who does not know Web service technologies. Given a user's natural language request to a composite service, the method generates an abstract workflow, which describes the constituent tasks and their transitions in a composite service. In the design step, our method gives two graphical ways based on the sequence diagrams which make it simple and understandable by even non experienced users.

The semantic approach [27-29] describes the meaning of the services based on the ontology. It gives a definition of an automated services composition and discusses their limitations. In our proposal, the semantic aspect is not taken into account. Consequently, semantic description presents a good way to increase the automation degree of our method.

Compared to the aforementioned composition methods, which mainly take into consideration either the semantic aspect using ontology, the functional aspect which is described using interfaces with input and output parameters or the non-functional aspect such as QoS, our approach is essentially based on the behavioral aspect using WSComposition profile in a graphical way to compose Web services.

Table 1 shows a brief comparison of some approaches cited in this paper and our approach.

7. Conclusion and Future Work

This paper introduced a new method for WSC. The proposed approach is based on MDA, and consists of using a WSComposition profile to model WSC and to generate

Table 1. Comparison between WSC approaches.

Approaches	Degree of automation	Design type	Aspect	Based modeling	Output format language
Our approach	Semi-automatic	Graphical	Behavioral	Sequence diagram	BPEL
Skogan approach	Semi-automatic	Graphical	Functional	Activity diagram	BPEL and WorkSCo
Bordbar approach	Semi-automatic	Graphical	Behavioral	Activity diagram	BPEL
Ko approach	Automatic	Interactive	Non-functional and semantic	client-friendly manner	OWL-S
Lim approach	Automatic	Natural language	Semantic	Workflow	OWL-S
Yue approach	Automatic	Graphical	Semantic and non-functional	OWL-S	Colored Petri Nets

an executable model (BPEL process) through transformation rules, expressed in ATL. The case study presented here proves the applicability of our method for WSC.

The existing simple Web services are discovered and located in the Web service registry. The developer imports the WSDL files of the candidate services and translates them into UML diagrams (class diagram, use case diagram and sequence diagrams). A WSComposition profile extending the UML2.2 sequence diagrams metamodel is defined; it allows the developer to model the composition of the Web services basing on the behavioral aspects. The MDA approach is adopted to transform the applied profile into a BPEL process. The transformation and rules are expressed in ATL to achieve a mapping from WSComposition profile to BPEL.

Our method has several advantages. It is graphical, which allows the developer to import descriptions of exiting Web services (WSDL) and represent them clearly in class diagrams. Furthermore, our approach is based on UML but it is independent of the UML modeling tools and the WSC language used. Another advantage is that our method gives a semi-automatic way to generate a BPEL process by adopting the MDA approach. These features are the keys that make our method simple and understandable by even non experienced users.

For future improvements, we will study the possibility to extend our WSComposition profile to take into consideration the user needs in terms of QoS. We will look into the possibility of using semantic Web services to automate Web services finder and to offer a better precision on existing services. Our mapping method is unidirectional, we study the possibility to make it bidirectional in order to enhance the automation degree.

REFERENCES

[1] W3C Working Group, "Web Services Architecture," 2004. http://www.w3.org/TR/ws-arch/

[2] W3C Working Draft, "Soap Version 1.2 Part0: Primer," 2001. http://w3.org/TR/2001/WD-soap12-part0-20011217

[3] W3C Note, "Web Services Description Language (WSDL) 1.1," 2001. http://www.w3.org/TR/wsdl

[4] UDDI Spec Technical Committee Draft, "UDDI Version 3.0.2," 2004. http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

[5] R. Karunamurthy, "Web Service Composition: Architecture, Frameworks, and Techniques," Ph.D. Thesis, Concordia University, Montreal, 2009.

[6] C. Pahl and Y. Zhu, "A Semantical Framework for the Orchestration and Choreography of Web Services," *Electronic Notes in Theoretical Computer Science*, Vol. 151, No. 2, 2006, pp. 3-18. doi:10.1016/j.entcs.2005.07.033

[7] Object Management Group, "MDA Guide Version 1.0.1," 2003.

[8] OASIS Standard, "Web Services Business Process Execution Language Version 2.0," 2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[9] ATLAS group LINA & INRIA, "ATL: Atlas Transformation Language," Version 0.2, 2005. http://www.uio.no/studier/emner/matnat/ifi/INF5120/v05/undervisningsmateriale/

[10] M. Juric, "Business Process Execution Language for Web Services," Packt Publishing Ltd., Birmingham, 2004.

[11] Object Management Group, "OMG Unified Modeling Language (OMG UML) Superstructure, Version 2.2," 2009. http://www.omg.org/spec/UML/2.2/Superstructure/PDF

[12] A. Jakimi, A. Sabraoui, M. Elkoutbi and A. Idri, "A New Approach for Composing UML Scenarios and Code Generation," *Proceedings of the IEEE* 4*th International Conference on Sciences of Electronics*, *Technologies of Information and Telecommunication*, Hammamet, Tunisia, 25-29 March 2007.

[13] "JDOM, Version 2.0.0," 2012. http://www.jdom.org/

[14] Object Management Group, "Object Constraint Language, OMG Available Specification, Version 2.0," 2006. http://www.omg.org/spec/OCL/2.0

[15] O. Patrascoiu, "YATL: Yet Another Transformation Language," *Proceedings of the* 1*st European MDA Workshop*,

MDA-IA, University of Twente, Enschede-Noord, 2004, pp. 83-90.

[16] O. Patrascoiu, "Model Transformations in YATL. Studies and Experiments," *Technical Report No.* 3-04, University of Kent, Kent, 2004.

[17] T. Mens, P. Van Gorp, D. Varró and G. Karsai, "A Taxonomy of Model Transformation," *Electronic Notes in Theoretical Computer Scienc*e, Vol. 152, 2006, pp. 143-159. doi:10.1016/j.entcs.2005.10.022

[18] D. Steinberg, F. Budinsky, M. Paternostro and E. Merks, "EMF: Eclipse Modeling Framework," 2nd Edition, Addison-Wesley Professional, New York, 2008.

[19] I. Jacobson, M. Christerson, P. Jonsson and G. Overgaard, "Object-Oriented Softaware Engineering: A Use Case Driven Approach," Addison-Wesley, New York, 1992.

[20] D. Skogan, R. Grønmo and I. Solheim, "Web Service Composition in UML," 8*th IEEE International Enterprise Distributed Object Computing Conference*, Monterey, 20-24 September 2004, pp. 47-57.

[21] B. Bordbar and A. Staikopoulos, "Modelling and Transformation of Behavioural Aspects of Web Services," 3*rd Workshop in Software Model Engineering* (WiSME), Inc Conjunction with UML, 2004.

[22] W. Nam, H. Kil and D. Lee, "On the Computational Complexity of Behavioral Description-Based Web Service Composition," *Theoretical Computer Science*, Vol. 412, No. 48, 2011, pp. 6736-6749. doi:10.1016/j.tcs.2011.04.020

[23] J. M. Ko, C. O. Kim and I. H. Kwon, "Quality-of-Service Oriented Web Service Composition Algorithm and Planning Architecture," *The Journal of Systems and Software*, Vol. 81, No. 11, 2008, pp. 2079-2090. doi:10.1016/j.jss.2008.04.044

[24] B. Zibanezhad, K. Zamanifar, R. Sadjady and Y. Rastegari, "Applying Gravitational Search Algorithm in the QoS-Based Web Service Selection Problem," *Journal of Zhejiang University*, Vol. 12, No. 9, 2011, pp. 730-742.

[25] J. Lim and K. Lee, "Constructing Composite Web Services from Natural Language Requests," *Web Semantics*: *Science*, *Services and Agents on the World Wide Web*, Vol. 8, No. 1, 2010, pp. 1-13. doi:10.1016/j.websem.2009.09.007

[26] Y. Ni and Y. S. Fan, "Model Transformation and Formal Verification for Semantic Web Services Composition," *Advances in Engineering Software*, Vol. 41, No. 6, 2010, pp. 879-885. doi:10.1016/j.advengsoft.2010.01.005

[27] H.-N. Talantikite, D. Aissani and N. Boudjlida, "Semantic Annotations for Web Services Discovery and Composition," *Computer Standards & Interfaces*, Vol. 31, No. 6, 2009, pp. 1108-1117. doi:10.1016/j.csi.2008.09.041

[28] S.-H. Yeganeh, J. Habibi, H. Rostami and H. Abolhassani, "Semantic dweb Service Composition Testbe," *Computers & Electrical Engineering*, Vol. 36, No. 5, 2010, pp. 805-817. doi:10.1016/j.compeleceng.2008.04.007

