Modeling and Verifying WS-CDL Using Event-B

Hong Anh Le and Ninh Thuan Truong

VNU - University of Engineering and Technology
144 Xuan Thuy, Cau Giay, Hanoi
{anhlh.di10,thuantn}@vnu.edu.vn

Abstract. The Web Services Choreography Description Language (WS-CDL) is an XML-based language that describes web service composition in the view point of choreography by defining their common and complementary observable behavior, where ordered message exchanges result in accomplishing a common business goal [3]. However, WS-CDL does not come with formal specification, nor with official vefication tools. In this paper, we present an approach to formalize and verify choreography composition described in WS-CDL. In the first phase, we propose to use Event-B as a formal method to model choreography interactions by transforming WS-CDL entities to Event-B elements. We use the Rodin platform, in the next phase, to verify some properties of the translated model. Finally, we run an example to illustrate our approach in detail.

Keywords: WS-CDL, composition, verification, Event-B.

1 Introduction

Building platform-independent and distributed software such as web services is a growing trend in software architecture. A Web service is a software system designed to support machine-to-machine interaction over a network. It is mainly based upon WSDL, UDDI, and SOAP standards to describe data type's exchanges, make services discoverable and specify patterns to invoke specific services respectively. In order to group a number of web services into a complex one, we can use some approaches such as choreography and orchestration composition. Composition of web services is increasingly accepted as a paradigm for integration of applications within and across organization boundaries.

The choreography view focuses on the composition in the global observation, while the orchestration describes the interaction between one participant and the others. There are some XML-based languages such as BPEL[13], WS-CDL [3] which are used to describe the composition. WS-CDL focuses on describing the business protocol among different participant roles and the participants perform all the behaviors. Due to lacking of formal semantics and grounding, ambiguous interpretation of a WS-CDL description possibly occurs. Therefore, research in the verification of choreographies' properties has been a recently emerging topic.

Event-B [2] is an evolution of the B method [1] that is more suitable for developing large reactive and distributed systems. Software development in Event-B

P.C. Vinh et al. (Eds.): ICCASA 2012, LNICST 109, pp. 290–299, 2013.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2013

begins by abstractly specifying the requirements of the whole system and then refining them through several steps to reach a description of the system in such a detail that can be translated into code. The consistency of each model and the relationship between an abstract model and its refinements are obtained by formal proofs. Support tools have been provided for Event-B specification and proof in the Rodin platform.

In this paper, we propose an approach to formalize a choreography model by a formal method, e.g. Event-B. Specifically, we present some rules to transform a WS-CDL package to an Event-B model. Some properties of the model can be automatically (or interactively) proved through proof obligations generated from Rodin platform [2] such as deadlock freeness, order of exchanged messages and some business requirements. The advantage of our approach is that the use of Event-B as a method to model choreography interactions since its strong point is modeling multi-agents and reactive systems. Our main idea comes from the similarity between some parts of WS-CDL and Event-B; hence we can formalize a choreography scenario by an Event-B model naturally. Moreover, the approach is such practical that we can implement a tool transforming a choreography model to an Event-B model (semi-) automatically. It makes sense as we can bring the formal verification to web service implementation. It also overcomes one of disadvantages that make formal methods absent in the web service based software because of the complexity of modeling.

The rest of the paper is structured as follows: Section 2 provides some background of WS-CDL and Event-B. Followed by Section 3, we propose an approach to model a WS-CDL package by formalizing its elements using Event-B method. Section 4 presents an example of Purchase order to illustrate our approach. Section 5 summarizes some related works. We give some conclusion and present future works in Section 6.

2 Backgrounds

As our approach focuses on modeling of web service choreography by using Event-B method, in this section, we introduce briefly background of Event-B and give an overview of WS-CDL.

2.1 Event-B

Event-B is a formal method for system-level modeling and analysis. Key features of Event-B are the use of set theory as a modeling notation, the use of refinement to represent systems at different abstraction levels and the use of mathematical proof to verify consistency between refinement levels [2]. An Event B model encodes a state transition system where the variables represent the state and the events represent the transitions from one state to another. A basic structure of an Event-B model consists of a MACHINE and a CONTEXT.

A MACHINE is defined by a set of clauses which is able to refine another MACHINE. We briefly introduce main concepts in Event-B as follows:

VARIABLES represent the state variables of the specification model.
INVARIANTS described by first order logic expressions, the properties of the attributes defined in the VARIABLES clause. Typing information, functional and safety properties are described in this clause. These properties are true in the whole model. Invariants need to be preserved by events clauses.
THEOREMS define a set of logical expressions that can be deduced from the invariants. Unlike invariants, they do not need to be preserved by events.
EVENTS define all the events that occur in a given model. Each event is characterized by its guard (i.e. a first order logic expression involving variables). An event is fired when its guard evaluates to true. If several guards evaluate to true, only one is fired with a non deterministic choice. The events occurring in an Event B model affect the state described in VARIABLES clause.

An Event B model may refer to a CONTEXT describing a static part where all the relevant properties and hypotheses are defined. A CONTEXT consists of the following items:

SETS describe a set of abstract and enumerated types.
CONSTANTS represent the constants used by the model.
AXIOMS describe with first order logic expressions, the properties of the attributes defined in the CONSTANTS clause. Types and constraints are described in this clause.
THEOREMS are logical expressions that can be deduced from the axioms.

2.2 WS-CDL

Web services composition integrates the existed available ones in order to form a new functionality. Recall that, choreography composition describes the interactions between collections of services without a central peer and all participants are treated equally. WS-CDL is a mark-up language for choreography composition which is first proposed in 2004. A WS-CDL choreography description is contained in a package which is essentially a container for a collection of activities performed the participants [6]. We introduce some elements of WS-CDL shortly as follows:

InformationType: This element identifies the type of information used within a choreography to avoid referencing directly to data types in an XML schema or a WS-CDL document.
RoleTypes: A RoleType enumerates the potential observable behaviors that a participant can exhibit in order to interact together.
RelationshipType: A Relationship Type describes the relationship between two parties in order to collaborate successfully.

The Choregraphy-Notation part specifies interactions between parties of a choreography. We address some concepts which are used frequently in a choreography:

Table 1. Translation from WSCDL static elements to Event-B

WS-CDL elements	Event-B concepts
informationType, participantType, channelType	Set
roleType	Constant
relationshipType	Axiom

Activities: There are three types of activity such as control-flow activities, Work-unit activities and basic activities. The first ones consist of three types namely sequence, choice and parallel activities. A work-unit activity describes the conditional and repeated execution of an activity. A basic activity includes Interaction, NoAction, SilentAction, Assign, and Perform element. Interaction is the most important element of WS-CDL.

Work-units: A work-unit describes constraints that need to be fulfilled to perform activities. It has guards and repeat conditions optionally. Enclosed activities are performed when the guard condition is evaluated to be true.

Variables: The information sent or received during an interaction is described by a named variable and an optional recordReference element in the exchange description. Variables contain values and have an informationType represented as a type of variables.

3 Formalizing and Verifying WS-CDL

From the similarity between some parts of WS-CDL and Event-B, we propose to use Event-B as a method to formalize a WS-CDL model. Since a WS-CDL package is composed of static and dynamic parts, we translate them to Event-B elements separately. After the transformation, we are able to verify some properties based on achieved Event-B model.

3.1 Formalizing Static Part

Before formalizing choreography interactions, we introduce some definitions related to Event-B specification that are useful in the modeling process.

Definition 1. A choreography scenario is modeled by a pair $ch = <S, In>$ where S is a set of static information, In represents for the dynamic part. S is stated as a 5-tuple: $S = <It, C, Rl, P, R>$, where It is a set of **InformationType** in a WS-CDL package, C and Rl indicate **Channels** and **Relationship** parts respectively, **Participants** and **Roles** elements are represented by P and R. The definition of In is discussed in Subsection 3.2.

Based on the formal definition, we translate WS-CDL elements in the static part to Event-B concepts as in Table 1.

3.2 Formalizing Dynamic Part

The dynamic part is the most important part of a WS-CDL package as it describes how the participants interact with each other to form a web service

composition. In order to make the approach more clear, we first model interactions in the choreography by an Event-B model. After that, we translate elements in the dynamic part of a WS-CDL package to an Event-B model.

Modeling Choreography Interactions: Interaction is the essential part of the composition which shows how a new functionality is composed from existing web services. We tranlsate the interaction among two colaboration parties to an Event-B EVENT and formalize the exchanged message by a pair $\{n \mapsto MSG\}$ where n is the order of message MSG. The guard of the translated event is also the order of the message as illustrated in Figure 1

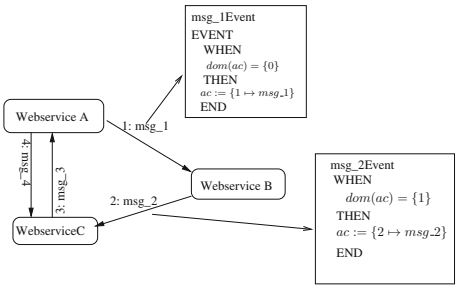

Fig. 1. Transformation from choregraphy interactions to Event-B EVENTs

Formalizing Interaction Part of a WS-CDL Package: We divide the choreography part of a WS-CDL file into two parts: variables and interactions description. Hence, we model In as a tuple $S = <Var, Ac>$, where Var specifies WS-CDL variables, Ac denotes WS-CDL interactions. We translate the former into Event-B variables, while, followed the approach of modeling interactions in the choreography, we formalize the later by Event-B events. In order to do the translation, we present some rules to transform most important WS-CDL entities involving in the dynamic part such as $Work-unit$, $Activity$ to Event-B concepts.

Activities: An activity entity comprises of several components including basic and ordering activities and work-units.

- Basic activities: A basic activity represents the lowest actions performed in a choreography such as an interaction, an assign, a silent, a noAction and a finalize activity. We encode a basic activity by a Event-B Event, more speficially, the syntax of translation is prensented in Table 2.
- Structured activities:
  A structured activity can be a sequence, parallel or choice activity. We model it by a set of Event-B events, for instance, a sequence activity is transformed to a set of Event-B events. We use an Event-B VARIABLE for representing

Table 2. Translation of WS-CDL basic activities to Event-B

WS-CDL basic activity	Event-B concepts
<assign $roleType$ = "$qname$" > <copy $name$ = "$ncname$" > <source $variable$ = "var_name1"/ > <target $variable$ = "var_name2"/ > </copy> < /assign>	WHERE Any THEN $var\_name2 := var\_name1$ END
<exchange $name$ = "$exchange\_name$" $informationType = infoType$ $action$ = "$request$" > <send $variable = var1$ / > < $receivevariable = var2$/ > </exchange>	INVARIANTS $var1 \in infoType$ and $var2 \in infoType$ EVENT WHEN $dom(msg) = \{i\}$ THEN $msg = \{i + 1 \mapsto exchange\_name\}$ END

Table 3. Transformation of a Workunit to an Event-B Event

WS-CDL work-unit	Event-B Event
<workunit $name$ = "$unitname$" $guard$ = "$xsd : booleanXPath - expression$"? $repeat$ = "$xsd : booleanXPath - expression$"? Activity-Notation < /workunit>	EVENT $unitname$ WHEN $guard$ and $repeat$ THEN body END

the order of each basic activity in the sequence and it is managed inside the body of each event. A parallel activity is formalized by a combination of events by means of parallel operator $\|$.

Work-unit: Since a Work-unit acts similarly to an Event-B event, we transform guard and repeatable conditions of a Work-unit to guard clauses of an Event-B event. Activities inside a Work-unit are represented by operations in the body part of an Event-B event. These operations will activate events corresponding to each Activity. The transformation is illustrated in Table 3.

4 An Example

In this section, we describe a well-known example of Purchase order. We then translate the WS-CDL package of this example to an Event-B model according to the rules presented in Subsection 3

4.1 Purchase Order Scenario Description

The scenario involves four participants: Buyer, Seller, CreditCard and Store services. The Buyer initiates the choreography by sending a purchase request to the Buyer with product and credit card information. The Seller then request to check product information with the Store service and the validity of the Buyer's credit card. If both results are positive then the Seller will reply a Purchase Order confirmation, otherwise the Seller rejects the request.

4.2 Mapping Purchase Order Model in WS-CDL to Event-B

In this Subsection, we describe the Purchase order scenario by a WS-CDL package. We then transform the description in the format of WS-CDL into Event-B model following the rules we define in Section 3.

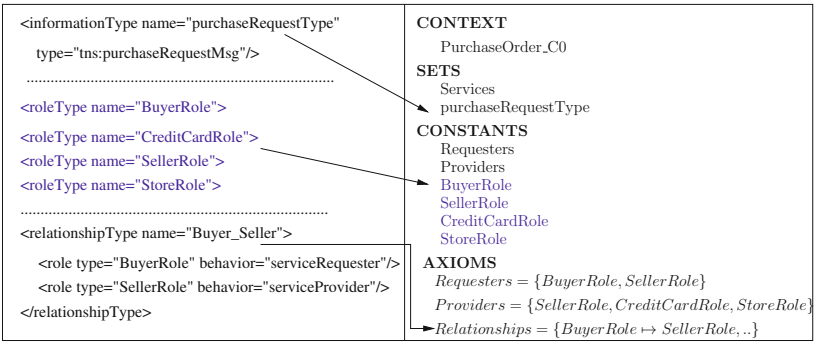

Fig. 2. Transformation from WS-CDL static part to an Event-B model

As illustrated in Figure 2, four participants in the choreography are described by four WS-CDL RoleType elements which are modeled by a set of CONSTANTS such as *BuyerRole, SellerRole, CreditCardRole and StoreRole*. We define two Event-B SETS which are namely *Requesters, Providers* indicating *fromRole* and *toRole* properties of *relationshipType* items in this WS-CDL package. Since *InformationType* elements are variable types, we model them as clauses of Event-B SETs.

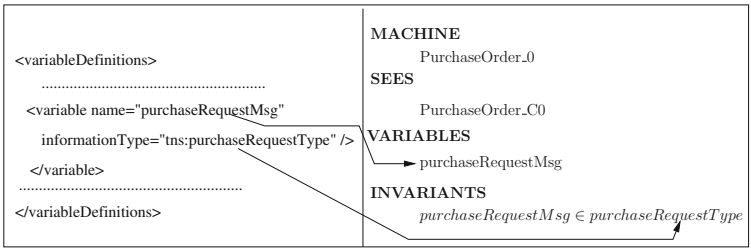

Fig. 3. Transformation from VARIABLES of WS-CDL to an Event-B VARIABLES.

In Figure 3, we model variable *PurchaseRequestMsg* which has type of *PurchaseRequestType* to an Event-B VARIABLE with the same name.

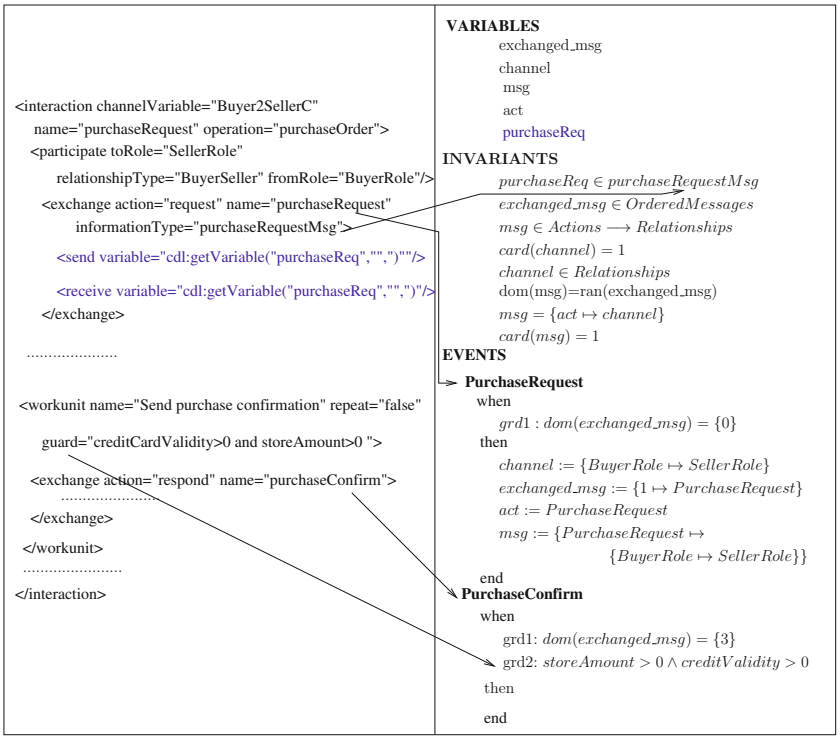

Fig. 4. Transformation from WS-CDL activities to Event-B EVENTS

The choreography part is modeled as depicted in Figure 4, the variable *exchanged_msg* represents messages exchanged between services which is ordered. The variable *channel* present the relationship between services. The exchange action *PurchaseRequest* and Work-unit *PurchaseConfirmation* modeled by two Event-B Events. Guard of this Work-unit is transformed to one of guards of *PurchaseConfirmation* Event-B event.

4.3 Verifying Purchase Order Model

Taking advantages of Event-B method and its support tool, after the transformation, we are able to verify some properties of the choreography interactions model as follows:

– First, we can verify that if the order of messages exchanged between collaboration parties are precise as choreography description . In our example, the order of exchanged messages is represented in the variable $exchanged\_messages$ and its preserving is showed by the following axiom:

$$exchanged\_messages \in \{0 \mapsto Init, 1 \mapsto PurchaseRequest, 2 \mapsto CheckCreditCard, 3 \mapsto CheckStore, 4 \mapsto PurchaseConfirm, 4 \mapsto PurchaseReject\}$$

– Second, messages are needed to be proved that they are exchanged between right source and destination web services. This property is modeled by two axioms in our example:

$$dom(msg) = ran(exchanged\_msg)$$
$$msg = \{act \mapsto channel\}$$

– Third, the last message of a choreography usually is the result of the composition, hence it is needed to be verified. In our example, the last message is either "PurchaseConfirm" or "PurchaseReject" but can not be both. This property is described by the axiom such as:

$$ran(last\_msg) = \{PurchaseConfirm\} \lor ran(last\_msg) = \{PurchaseReject\}$$
$$card(last\_msg) = 1$$

– Finally, live lock freeness and no deadlock properties are also proved through this Event-B model since there is at least one event is triggered and is no conflict between guards of the events.

These properties are proved to be preserved through all EVENTS of the model by the Rodin tool.

5 Related Works

Many papers have been proposed for verifying web service composition. G.Salaun *et al.* [15] developed a process algebra to derive the interactive behavior of a business process out from a BPEL specification, while A. Brogi *et al.* [7] presented the formalization of Web Service Choreography Interface (WSCI) using a process algebra approach(CCS), and showed the benefits of the formalization.

More recently, Yahong Li et al. [16] introduced a small language CDL in order to formalize WS-CDL. However, in order to verify the choreography composition, this formal model is translated into notations of SPIN and the general transformation rules are not given yet.

Pengcheng Zhang *et al.* [14] introduced an approach to model and verify WS-CDL using different UML diagrams. Gregorio Diaz *et al.* proposed a method to analyze WS-CDL by translating it into timed automata [10].

Idir Ait-Sadoune *et al.* [4] presented the transformation rules from an orchestration language, namely BPEL to an Event-B model and a support tool called BPEL2B. The structure of WS-CDL is clearly more complicated than the one of BPEL as the collaboration between participants in the choreography model is more complex without a central one.

6 Conclusion

In this paper, we have proposed an approach to formally model and verify web services choreography using Event-B. Our contribution includes the definition of rules to transform WS-CDL entities to Event-B elements and an example to

illustrate the approach. The aim of the transformation is automatically verifying some properties such as order of messages, live lock freeness, deadlock, etc. in WS-CDL model. However, our approach is just suitable for a simple collection of interactions but not for complex position one. Our future work is focusing on using Event-B refinement mechanism and handling time out case in choreography. We are building a tool which allows a WS-CDL model to automatically transform to an Event-B model according to our defined rules.

Acknowledgments. This work is partly supported by the research project No. QG.11.32 granted by Vietnam National University, Hanoi.

References

1. B method web site, `http://www.bmethod.com`
2. Event-b and the rodin platform, `http://www.event-b.org`
3. Web services choreography description language version 1.0., `http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/`
4. Ait-Sadoune, I., Ait-Ameur, Y.: From bpel to event-b. In: IM FMT 2009, Dusseldorf, Germany (February 2009)
5. Ait-Sadoune, I., Ait-Ameur, Y.: Stepwise Design of BPEL Web Services Compositions: An Event_B Refinement Based Approach. In: Lee, R., Ormandjieva, O., Abran, A., Constantinides, C. (eds.) SERA 2010. SCI, vol. 296, pp. 51–68. Springer, Heidelberg (2010)
6. Dumas, M., Barros, A., Oaks, P.: A critical overview of web service choreography description language, ws-cdl (2005)
7. Brogi, A., Canal, C., Pimentel, E., Vallecillo, A.: Formalizing web service choreographies. Electron. Notes Theor. Comput. Sci. 105, 73–94 (2004)
8. Bryans, J.W., Wei, W.: Formal Analysis of BPMN Models Using Event-B. In: Kowalewski, S., Roveri, M. (eds.) FMICS 2010. LNCS, vol. 6371, pp. 33–49. Springer, Heidelberg (2010)
9. Decker, G., Puhlmann, F., Weske, M.: Formalizing Service Interactions. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 414–419. Springer, Heidelberg (2006)
10. Diaz, G., Pardo, J.-J., Cambronero, M.-E., Valero, V., Cuartero, F.: Automatic translation of ws-cdl choreographies to timed automata. In: EPEW 2005/WS-FM 2005, Berlin, Heidelberg, pp. 230–242 (2005)
11. Foster, H., Kramer, J., Magee, J., Uchitel, S.: Model-based verification of web service compositions. In: 18th IEEE International Conference on Automated Software Engineering (ASE), pp. 152–165 (2003)
12. Hoang, T.S., Iliasov, A., Silva, R., Wei, W.: A survey on event-b decomposition. ECEASST 46 (2011)
13. Jordan, D.: Web services business process execution language (ws-bpel). standard version 2.0
14. Zhang, Y.P., Muccini, H., Li, B.: Model and verification of ws-cdl based on uml diagrams. International Journal of Software Engineering and Knowledge Engineering 20, 1119–1149 (2010)
15. Salaün, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: ICWS 2004, Washington, DC, USA, p. 43 (2004)
16. Yang, H., Zhao, X., Qiu, Z., Pu, G., Wang, S.: A formal model forweb service choreography description language (ws-cdl). In: ICWS 2006, pp. 893–894. IEEE Computer Society, Washington, DC (2006)