2012 IEEE/ACIS 11th International Conference on Computer and Information Science

Research on Formal Modeling and Verification of BPEL-based Web Service Composition

Huiqun Zhao, Wenwen Wang, Jing Sun, Ying Wei
Department of Computer Science
North China University of Technology
Beijing, China
zhaohq6625@sina.com, www321y@126.com, sunjing8248@163.com, weiying_0913@126.com

***Abstract*—With the development of Web Service composition, more and more diversified and complex business demands are satisfied. But the logical validity cannot be guaranteed. After a short view of recent research efforts of formal modeling and verification about Web Service, this paper proposes a new formal model for WS-BPEL described Web Service composition. The specification language of the model is LOTOS. Model checking is adopted to ensure the validity of this model. Finally, an example is presented to illustrate the practicality of the model.**

Keywords- Web Service; BPEL; formal description model; LOTOS; model checking

I. INTRODUCTION

WS-BPEL[1](Web Service-Business Process Execution Language, BPEL in short) solves the problem of limited function using single Web Service. It composes several simple Web Services in order to provide advanced function for users. But because the Web Services are dynamic and the WS-Policy is changing, there may be problems in the business process of BPEL described composition. If an inaccurate business process is deployed without verification, problems will arise when running it. Repairing the system will be costly, too. It is necessary to verify the business process before coming into use.

The academia has paid attention to modeling and verification of Web Service composition, and some research results of this problem can be seen now. Most of the formal research results are about Petri net, process algebraic and automata theory. Some representative work is introduced below. 1) About Petri net: Reference [2] holds that the existing description models of Web Services composition depend on concrete composition process description language. Moreover, they cannot give a comprehensive picture on Web Service composition in order to resolve problems; a colored Petri net model was put forward to describe the Web Service. The model descriptions of five basic web composition structures were presented to construct the process of Web composition which fulfils actual requirement. At last they gave a demonstration for the actual application of the mode by an example modeling. Reference [3] prefers to use Petri net to model and verify Web Service composition. It also proposes an algorithm to translate WF-Net to BPEL. The validity of BPEL process can be guaranteed according to verifying the WF-Net process [4]. Colored Petri net is adopted to model ontology based Web Service composition in Reference [5]. Corresponding semantics and operators are constructed. It defines measure to judge the validity of composition. 2) About process algebra: Process algebra is studying concurrent systems in algebraic method. It includes CCS, CSP, Pi-calculus and so on. Reference [6] presents a Pi-calculus based formal description model for Web Service, and gives the mapping of BPEL4WS specification and WS-CDL specification. It also instructs that the mapping above is consistent in the model. The dynamic architecture of Web Service composition can be described by the method when it is used to design Web Service composition directly. Reference [7] proposes a Pi-calculus based formal description. It defines the mapping of concept between Pi-calculus and OWL-S. It also gives the method of verifying the validity of the model. 3) About automata theory: Reference [8] starts from the message interaction between services and formally describes the services into non-determinate Büchi automata with a FIFO message queue. It regards Web Service composition as a global session protocol which models the asynchronous message passing between services. It provides the feasible condition of the session protocol and the synchronization condition of messages. It describes these conditions and goal property of system in LTL assertion. SPIN is adopted to verify the correctness. Reference [9] uses finite state machine to model the BPEL-described Web Service composition. It verifies the safety and liveness of the process in order to assure the correctness of the service. Moreover, there is another model method called ontology. Reference [10] proposes a modeling and composition method which describes logic rules. This method is based on OWL-S and DL-ruled framework. The problem that DL cannot describe the dynamic feature of Web Service is solved. Reference [11] adopts a three-layer architecture ideology namely OWL-S to compose Web Service. It uses GA as middle model and Promela model as verification model. It fulfills the transition between Web Service composition model and GA model. It also transfers GA model to Promela model. The verification work takes advantage of SPIN tool.

Process algebra takes expression as describing method. The express ability of it is very strong and the form is concise. But Pi-calculus is short of intuitional graphic representation and supported tools. It is not very convenient to use it. Although the method of Petri net and automata in describing Web Service composition is more intuitional than

978-0-7695-4694-0/12 $26.00 © 2012 IEEE
DOI 10.1109/ICIS.2012.92

IEEE computer society

Pi-calculus. When the business process is complicated, it will lead to state space explosion.

In this paper, we adopt formal description language LOTOS to model BPEL described Web Service composition. Model checker Evaluator in CADP toolset is used to verify the built model. Then the correctness of the Web Service composition process can be guaranteed in order to discover the problem before the system comes into use. More loss can be avoided.

II. BASIC KNOWLEDGE

Formal methods are developed so as to solve "software crisis" at the end of the 1960's. Formal methods provide a appropriate framework to dispose the description, composition and verification problems. It has many formal specification language and advanced analysis tools. The meaning is that it can help discover indetectable inconsistency, ambiguity and imperfection of system specification. It is an effective way to reduce mistake in design and enhance the reliability of system.

A. Model checking

Model checking is a kind of verification technique in formal method. It verifies a certain property according to exhaustion the state space of the system. Automatic and complete safety analysis can be put into effect in Web Service composition. Counter-example will be given when the Web Service composition violates safety principle. It is very helpful to locate the safety problems in the system. The flow of model checking is system modeling, property modeling and verification.

System description language, property modeling method and model checking tool selecting are illustrated as follows:

B. LOTOS

Language of Temporal Ordering Specification (LOTOS for short) is a kind of formal description language. LOTOS is a Formal Description Technique (FDT) standardized by ISO for the design of distributed systems. It helps us describe concurrency, nondeterminism, synchronous and asynchronous communication within the system.

Complete LOTOS can be divided into four parts:

1) Specification declare part, this part declares the name, gate, exit type and so on.
2) Import abstract data type part, this part may include predefined type such as Natural type and user defined type.
3) Main behavior part, this part describes the general behavior of the system and the interaction between processes.
4) Process definition part, this part describes the specific behavior of the processes.

C. Property modeling

It is necessary to model the property while verifying a system using model checking technical. That is to express the system property in specific formula. There are several means of expression such as propositional logic, temporal logic, Pi-calculus, mu-calculus and so on. In this paper, mu-calculus is chosen to describe the property of system. The full name of mu-calculus is regular alternation-free mu-calculus. It can express temporal logic with data which is effective for model checking algorithm. Mu-calculus can express safety property, liveness property and fairness property.

- **Safety property.** Informally, safety property expresses that "something bad never happens." Typical safety property is those forbidding "bad" execution sequences in the LTS. For example, mutual exclusion can be characterized by the following formula:

[true*."OPEN!1".(not CLOSE !1")*."OPEN !2"]false

states that every time process 1 enters its critical section (action "OPEN !1"), it is impossible that process 2 also enters its critical section (action "OPEN !2") before process 1 has left its critical section (action "CLOSE !1").

Other typical safety properties are the invariants, expressing that every state of the LTS satisfies some "good" property. For example, deadlock freedom can be expressed by the formula below:

[true*] < true > true

- **Liveness property.** Informally, a liveness property expresses that "something good eventually happens." Typical liveness properties are potentiality assertions (i.e., expressing the reachability on a sequence) and inevitability assertions (i.e., expressing the reachability on all sequences).

Potentiality assertions can be directly expressed using diamond modalities containing regular formulas. For instance:

< true* . "GET !0" > true

states that there exists a sequence leading to a "GET !0" action after performing zero or more transitions.

Inevitability assertions can be expressed using fixed point operators. For instance, the following formula:

mu X . (< true > true and [not "START"] X)

states that all transition sequences starting at the current state lead to "START" actions after a finite number of steps.

- **Fairness property.** Fairness property is similar to liveness properties, except that they express reachability of actions by considering only fair execution sequences. One notion of fairness that can be easily encoded in the logic is the "fair reachability of predicates". A sequence is fair iff it does not infinitely often enable the reachability of a certain state without infinitely often reaching it. For instance:

[true* . "SEND" . (not "RECV")*]
< (not "RECV")* . "RECV" > true

states that from every state of such a circuit, there is still a finite sequence leading to a "RECV" action.

Using the above properties, most properties of Web Service composition can be well expressed such as deadlock and reachable.

D. Mode checker – evaluator

While verifying Web Service composition using model checking method, the state space often increases exponentially. If the state space is large, searching the space directly is actually impossible. This is state explosion problem. The model checker we choose in this paper which called evaluator can solve the state explosion problem.

Evaluator is the model checker of CADP toolset. CADP [14][15] (Construction and Analysis of Distributed Processes) is a popular toolbox for the design of communication protocols and distributed systems. CADP offers a wide set of functionalities, ranging from step-by-step simulation to massively parallel model-checking.

Evaluator can analyze designate property of the system model and judge whether the system model satisfy the designate property. A positive example or counter example will generate finally.

III. FORMAL MODELING FOR BPEL

BPEL is a kind of XML-based programming language. It can automatically fulfill the business process of Web Service composition. The syntax of business process is defined based on the interaction between the participants. BPEL adapts the advantages of Petri net and Pi-calculus. It is an abstract executable modeling language of high-level.

In order to verify the BPEL described business process in formal methods, we should translate BPEL process to formal language. Due to the excellent descriptive capacity of LOTOS, this paper adopts LOTOS to model the BPEL process. Now the methods how to translate BPEL into LOTOS will be presented.

A. Modeling for LOTOS main behavior

BPEL is a kind of orchestration language. BPEL, as the core process, describes the executive logic of Web Service application according to defining control flow. It rules the interaction of called services in order to fulfill function. BPEL process can be divided into three parts that are client process, BPEL process and invoked services. BPEL process starts from receiving request from client and ends with the replying to the client.

LOTOS main behavior, combined with its own character, can be built into a three-level model which include client process, BPEL process and invoked service processes. The interaction between client and BPEL process, BPEL process and invoked service processes are preceded on different gates. There is no interaction between invoked service processes. So the whole business process can complete. The example code of LOTOS main behavior is shown as follow:

```
behavior
    Client [request, response]
|[request, response]|
    BpelProcess [request,response,WS1,WS2,WS3]
|[WS1,WS2,WS3]|
(
    WebService1 [WS1]
|||
    WebService2 [WS2]
|||
    WebService3 [WS3]
)
```

Now the LOTOS main behavior framework has been set up. But the orchestration of BPEL has not been built. According to LOTOS syntax, we should import the definition of all the processes quoted in the main behavior. The most important is the definition of BpelProcess() which stand for the whole executive procedure of business process. The modeling of specific activities of BPEL is shown as follows.

B. Translation of BPEL basic activities

BPEL orchestrates business process by means of activities. An activity is a statement of BPEL or an executive procedure. BPEL activities consist of basic activities and structured activities.

BPEL basic activities include 9 activities. The translation from basic activities to LOTOS is modeled in TABLE I.

TABLE I. Translation rules of BPEL basic activities

BPEL activity	LOTOS	comment
<receive portType=”qname” variable=”m”…> </receive>	qname ? m : Nat ;	Receive the request from the client
<reply portType=”qname” variable=”m”…> </reply>	qname ! m;	Reply to the client
<invoke… portType=”qname” inputVariable=”mI” outputVariable=”mO”> </invoke>	qnane !mI ?mO : Nat ;	Call other deployed service
<… act1 …/> <assign><copy> <from expression=”5”/> <to var=”x”/> </copy></assign> <… act2 …/>	act1; exit(5)>> accept x:Nat in act2	Assign a value to another variable
<empty> standard-elements </empty>	process empty [] := endproc	Non-action
<wait (for=”duration”| until=”deadline> </wait>	i;	Wait for a certain time
<exit…> </exit>	exit	Exit
<throw faultName=”name” faultVariable=”f’> </throw>	qname ! fM;	Throw a exception
<rethrow faultName=”name” faultVariable=”f’> </rethrow>	qname ! fM;	Rethrow a exception

C. Translation of BPEL structured activities

Structured activities can describe complex business process with integrating basic activities. Handling of control pattern, data flow, breakdown, external events and message exchange between process instances can be represent by these structures. As container of basic activities, a structured activity can contain another structured activity. That is called nested container. The translation of structured activities is shown in TABLE II.

TABLE II. Translation rules of structured activities

BPEL activity	LOTOS	comment
<sequence> <…act1…/> <…act2…/> </sequence>	act1 ; act2	Sequence activities
<if condition=”x>=0”> <…act1…/> <else><…act2…/> </else></if>	[x>=0] -> act1; [] [x<0] -> act2;	Condition structure
<while condition=”x>=0”> <…act1…/> <while>	process while[] := [x<0] -> i;[] [x>=0] -> act1; while1[] endproc	Loop structure
<repeatUntil> <…act1…/> <condition=”x>=0”/> </repeatUntil>	Process repeatUntil[]:= act1; ([x<0] -> i; [] [x>=0]->act1;while1[]) endproc	Similar with loop structure, but the activity executes at least once
<pick><onMessage portType=”q1”> <…act1…/> </onMessage><onMessage portType=”q2”> <…act2…/> </onMessage></pick>	(q1 ? m1:Nat; act1) [] (q2 ? m2: Nat; act2)	A set of mutual exclusion.
<flow ><..act1…> <source linkname=”link1” condition=”cond1”/> </act1> < ...act2... > <target linkname="link1"/> </act2></flow>	act1; ([cond1]->link1 !1; [] [not(cond1)]-> link1 !0;) || (link1 ?x:Bool; ([x=1] -> act2 [] [x=0] -> i;))	A set of paralleled activities

D. An example of transition

A simple example will be present to explain the LOTOS formal model raised above.

The bank should provide different down payment and loan rate to house buyers according to the quantity of house they possess. The more houses they possess the higher down payment and loan rate they should undertake.

- **The BPEL process of example.** In order to enforce the function mentioned above, the BPEL process receives the request of client at first. Then it invokes the houseloanagency Web Service to get the quantity of house.

At last, it invokes different Web Services according the quantity.

The BPEL process after simplification is shown as follows:

```
<process>
  <receive portType=”Client”/>
  <invoke portType=”HouseLoanAgency”/>
  <switch>
    <case condition=”houseNum=0”>
      <invoke portType=”Bank0”/>
    </case>
    <case condition=”houseNum=1”>
      <invoke portType=”Bank1”/>
    </case>
    <case condition=”houseNum=2”>
      <invoke portType=”Bank2”/>
    </case>
    <case condition=”houseNum>2”>
      <invoke portType=”Bank3”/>
    </case>
  </switch>
  <reply portType=”client”/>
</process>
```

- **Modeling of example.** The LOTOS model is shown as follows:

```
specification houseLoanBroker [client, houseLoanAgency, Bank0, Bank1, Bank2, Bank3] : noexit
behaviour
    Client [client]
    |[client]|
    houseLoanBroker[client, houseLoanAgency, Bank0, Bank1, Bank2, Bank3]
    |[houseLoanAgency, Bank0, Bank1, Bank2, Bank3]|
    (
    houseLoanAgency [houseLoanAgency]
    |||
        Bank0 [Bank0]
    |||
        Bank1 [Bank1]
    |||
        Bank2 [Bank2]
    |||
        Bank3 [Bank3]
    )
where
process houseLoanBroker[client, houseLoanAgency, Bank0, Bank1, Bank2, Bank3] : noexit :=
    client ?a : Nat;
    houseLoanAgency !a;  houseLoanAgency ?b: Nat;
    (([b=0] -> Bank0 !a ?c : Nat; exit(c))[]
      ([b=1] -> Bank1 !a ?c : Nat; exit(c))[]
      ([b=2] -> Bank2 !a ?c : Nat; exit(c))[]
      ([b>2] -> Bank3 !a ?c : Nat; exit(c))
    )>>accept c : Nat in client !c;stop
endproc
endspec
```

Not all the process definitions are listed above because of the length of article. The main behavior is modeled referring to the model presented in chapter 3.1. The BPEL process is modeled according to chapter 3.2 and chapter 3.3.

IV. DESIGN AND IMPLEMENTATION OF TRANSITION TOOL FROM BPEL TO LOTOS

The transition syntax rules have been introduced above. But for developers of Web Service composition, describing the system in LOTOS is relatively difficult. So an automatic transition tool from BPEL to LOTOS is designed and implemented in this article. This tool is based on the rules present above.

In order to translate from BPEL to LOTOS, the information contained in BPEL should be extract at first. This article adopts DOM (Document Object Module) to parse BPEL file. DOM can parse BPEL into tree structure with element, property and text. While parsing BPEL, the root element of the document is parsed at the beginning. Then every branch of root element will be parsed. Eventually the whole BPEL can be parsed.

The parse algorithm is shown as follows.

1) Start to model when read beginning tag <process>.
2) Judge the activity type when read activity name.
   a) If the activity is assign, receive, invoke, reply, throw, rethrow, exit, empty or wait, model it as convention. Then jump to step 3).
   b) If the activity is pick, if, flow, while or repeatUntil, model it as convention and every subactivity should be modeled as in step 2), then jump to step 3).
   c) If the activity is sequence, suppose there are n subactivities.
      If n=1, model the subactivities as in step2.
      If n>1, model the subactivities according to the sequence.
3) If the activity is </process>, modeling ends. If not, jump to step 2.

The transition tool can automatically generate LOTOS files after reading BPEL files. Evaluator can model check corresponding properties on the LOTOS file. If the system satisfies the property, the result will be true. If not, a counter example will be given. The principle figure of translate and evaluate is shown below:

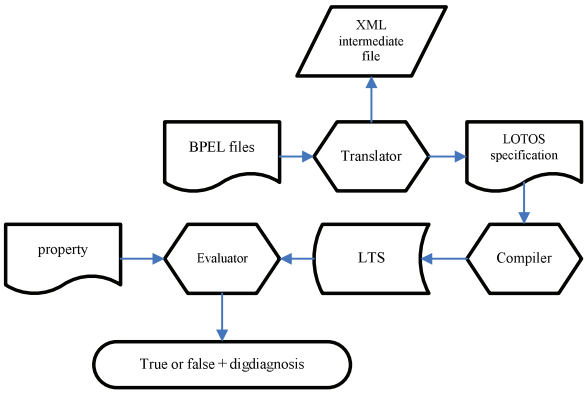

Figure1. Principle figure of translate and evaluate

V. APPLIED CASE

The case is based on the internet of things which is hot at present. Another team of our laboratory develops a Web Service. The enterprise can subscribe information of certain products according to sending message to our Web Service. The Web Service invokes different ALE (short for Application Level Event) terminals to get corresponding information. Then the Web Service can reply to the enterprise. The Web Service is a kind of BPEL process.

The tool raised above can translate the BPEL file to LOTOS file automatically. Evaluator of CADP can model check the LOTOS file on-the-fly. The temporal logic formula is written in mu-calculus mentioned in chapter 2.3.

Property 1:

[true*] < true > true

The result is true. It represents that the system has no deadlock. The result chart is shown in figure2.

Property 2:

[true*.'ALE !*'] <true* . 'ALE1 !* !* !*'> true

The result is true. It represents that the enterprise will receive a response no matter what it requests. The result is shown in figure 3.

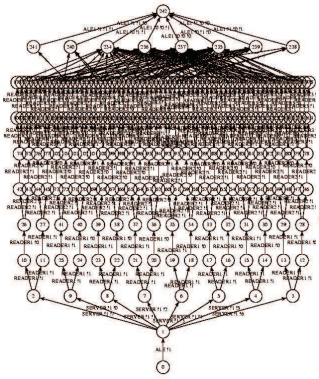

Figure 2. result of property 1

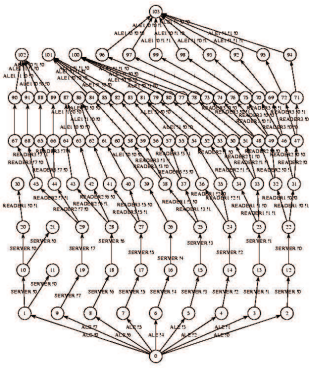

Figure 3. result of property 2

VI. CONCLUSION

With the development of Web Service composition, the academia has given several formal description of BPEL process. They use different formal languages in modeling. But research on LOTOS language is seldom or they did not give detailed mapping method. Most of the methods are aimed at the early BPEL4WS specification but not WS-BPEL2.0. The property is written in LTL which is short for Linear Temporal Logic. The description ability of LTS is a little weak.

A LOTOS model of Web Service composition is raised in this article. The model which is based on model checking can describe all kinds of BPEL activities. The new activities have also been translated. At the same time, the properties of system are described with mu-calculus which can model the property in detail. At last, the algorithm of transition from BPEL to LOTOS is given for convenience.

ACKNOWLEDGMENT

This paper is jointly sponsored by the Natural Science Foundation of China (Ref: 61070030，61111130121) and Beijing Government and Education Committee (Ref: PHR20 1107107). Thank those 2 institutions for their support.

REFERENCES

[1] T.Andrews, F.Cubera, H.Dholakkia. Business Process Execution Language for Web Services2.0.http://www.128.ibm.com/ developerworks/library/specification/ws-bpel/,2003

[2] Li Jingxia, Zhao Huijuan. Description and validation of Web Service composition based on coloured petri net[J].Computer Application and Software. 20112011,28(3):80-82.

[3] VAN DER ALAST W M P,DUMASM,TER HOFSTED A H M.Web serivce composition language:old wine in new bottles[C]//Proceedings of the 29th EU ROMICRO Conference on New Waves in System Archiecture. Washington, D.C.,USA:IEEE,2003:298-305.

[4] VAN DER AALST W M P,LASSEN K B. Translating workflow nets to BPEL4WS[R]. Eindhoven, The Netherlands:Eindhoven University of Technology,2005.

[5] Luo Nan, Yan Junwei, Liu Min.Verification mechanism for semantic Web Service composition based on colored Petri-nets[J]. Computer Integrated Manufacturing Systems. 2007, 13(11) : 2203-2210.

[6] Hu Jing, Feng Zhiyong. Pi-calculus based formal description model for Web Services[J]. Application Research of Computers. 2011,28(6):2168-2173.

[7] Li Yanhong, Yue Houguang.Research on Automatic composition of Web Service based on Pi-calculus and OWL-S[J].China Computer and Communication.2011,4:42-43.

[8] BULTAN T,FU X,HULL R,et al.Conversation specification:a new approach to design and analysis of Web Service composition[C]//Proceedings of World Wide Web Conference. New York, N.Y., USA:ACM,2003:403-410.

[9] FOSTER H, UCHITEL S, MAGEE J, et al. Compatibility verification for Web Service choreography[C]//Proceedings of IEEE International Conference on Web Services. Washington, D.C., USA:IEEE, 2004:738-741.

[10] Liu Sipei, Liu Dayou, Qi Hong. Composition Semantic Web Service with Description Logic Rules[J].Journal of Computer Research and Development. 2011,48(5):831-840.

[11] Teng Xingquan, Li Zengzhi. Study of verification of Web Services Composition Based on OWL-S[J]. Microellectronics and Computer.2004, 24(12):62-65.

[12] Wing J. M. A Specifier's Introduction to Formal Methods[J]. IEEE computer, 1990, 23(9):8-24

[13] ISO/IEC 8807-1989, Information process systems, open systems interconnection, LOTOS—— A formal description technique based on the temporal ordering of observational behavior[S].

[14] Serge P Hoogendoom, Piet H L Bovy. Dynamic user-optional assignment in continuous time and space, Transportation Research Part, 2007: 571-592

[15] Andreas Schadschneider, Wolfgang Knospe, Ludger Santen. Michael schreckenberg, optimization of highway networks and traffic forecasting. Physica, 2005:165-173.

[16] D. Kozen. "Results on the Propositional Mu-Calculus." Theoretical Computer Science, v. 27, p. 333-354, 1983.

[17] E. A. Emerson and C-L. Lei. "Efficient Model Checking in Fragments of the Propositional Mu-Calculus." Proceedings of the 1st LICS, p. 267-278, 1986.