A Comparative Study of Web Service Composition via BPEL and Petri Nets

Mohammad Salah Uddin, *Member IACSIT*, S. Ripon, Nakul C. Das, and Orin Hossain

***Abstract*—Web services technology provides a platform on which we can develop distributed services. The interoperability among these services is achieved by various standard protocols. In recent years, several researches suggested that Petri Nets provide a satisfactory assistance to the whole process of web services development. Business transactions, on the other hand, involve the coordination and interaction between multiple partners. With the emergence of web services, business transactions are conducted using these services. The coordination among the business processes is crucial, so is the handling of faults that can arise at any stage of a transaction. BPEL models the behavior of business process interaction by providing a XML based grammar to describe the control logic required to coordinate the web services participating in a process flow. However BPEL lacks a proper formal description where the composition of business processes cannot be formally verified. Petri Nets, on the other hand, facilitates a formal foundation for rigorous verification of the composition. This paper presents a comparison of web service composition between BPEL and Petri nets.**

***Index Terms*—BPEL, web service, SOAP, petri nets, XML.**

I. Introduction

The term "web service" describes a standardized way of integrating web-based application using open standards (e.g. XML, SOAP, and UDDI) over an internet protocol backbone. Used primarily as a means for businesses to communicate with each other and with clients, web services allow organizations to communicate data without intimate knowledge of each other's IT system behind the firewall. Instead of providing GUI's to users, web services share business logic, data and processes through a programmable interface across the network.

Business Process Execution Language (BPEL), short for Web services Business Process Execution Language (WS-BPEL) is an OASIS standard executable language for specifying actions within business processes with web services. Processes in BPEL export and import information by using web service interfaces exclusively. For more about BPEL please see [1].

Petri nets [2], [3] are a well-founded process modeling techniques that have formal semantics. They have been used to model and analyze several types of processes including protocols, manufacturing systems, and business processes. A petri net is a directed, connected, and bipartite graph in which each node is either a place or a translation. Tokens occupy places. When there is at least one token in every place connected to a transition, we say that the transition is enabled. Any enabled transition may fire removing one token from every input place, and depositing one token in each output place. For more elaborate introduction to petri nets, the reader is referred to [2], [4], [5].

Manuscript received July 8, 2013; revised September 20, 2013

Mohammad Salah Uddin, S. Ripon, Nakul C. Das and Orin Hossain are with the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh (e-mail: akash.bangla@gmail.com).

II. Related Work

This paper presents our on-going research of comparing the composition of web services by Petri nets and BPEL. We model the transaction of a Car Broker web Service using both BPEL and Petri nets examining the expressiveness of both languages.

In the reminder of the paper, Section III gives an overview of the Petri nets and its constructs. Section V first briefly describes our case study web service and then model the web service in both BPEL and Petri nets. For brevity only an abstract version of Car Broker web Service is modeled in this paper. Finally, we conclude our paper and outline our future plans.

Several research issues, both theoretical and practical, are raised by web services. Some of the issues are to specify web services by a formally defined expressive language, to compose them, and to ensure their correctness; formal methods provide an adequate support to address these issues. Recently, many XML-based process modeling languages (also known as choreography and orchestration languages) such as WSCI, BPML, BPEL4WS, WSFL, XLANG have emerged that capture the logic of composite web services. These languages also provide primitives for the definition of business transactions.

There are some studies on the modeling and analysis of a single BPEL processes using Petri nets. [6]-[8]. Hinz [8] presents a Petri net model for BPEL, and this model covers the exceptional behavior (e.g., faults, events, compensation). Lohmann [6] uses Petri nets to decide the controllability of a service, i.e., the existence of a partner process, and compute its operating guideline. Ouyang *et al*. [7] present a comprehensive and rigorously defined mapping of BPEL constructs into Petri net structures. Ripon [11] shows the composition of web service through BPEL and cCsp Process Algebra.

III. Petri Nets

Petri nets is a graphical and mathematical modeling tool applicable to many systems [5]. It is very useful tool for specifying information processing systems, which are describe as being asynchronous, parallel, concurrent,

distributed or dynamical [5].

A. Atomic Process

The Petri nets model for an atomic process described by BPEL is a Petri nets $PN = <S, T, I, O>$ as shown in Fig. 1, where

$$S=\{s\};$$

It represents the service to be run when s includes.

Tokens.

$$T=\{t_s, t_e\},$$

where t_s represents beginning of the service and t_e represents ending of the service.

$I(t_s, s)$ and $O(s, t_e)$ represent Input and Output respectively.

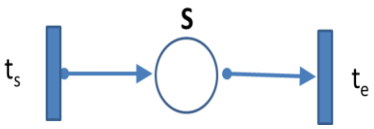

Fig. 1. Atomic process model

B. Sequence Process

The petri nets model for sequence composition of web services is a hierarchical Petri nets $SPN= <S, PN, I, O>$ as shown in Fig. 2, where

$$S = \{s_1, s_2, s_3 \ldots\ldots s_k\}$$
$$PN = \{PN_1, PN_2, PN_3\ldots\ldots\ldots\ldots PN_k\},$$

where PN_i is a subset of i-th service.

$$I = \{I(PN_i, s_i), O(s_i, PN_{i+1}) \mid i= 1,2,3,4,\ldots\ldots\ldots\ldots, k\text{-}1\}.$$

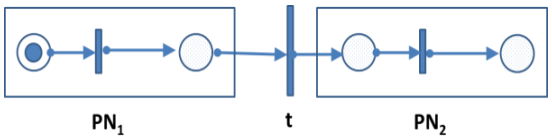

Fig. 2. Sequence model

C. Split Process

The Petri Nets model for split composition of web service is a hierarchical Petri nets $SPN = <S, PN, I, O>$ as show in Fig. 3, where,

$$S = \{s_{in}\}$$
$$T= \{t_1\}$$
$$PN = \{PN_1, PN_2, PN_3\ldots\ldots\ldots\ldots PN_k\},$$

where PN_i is a subset of i-th service.

$$I = \{I(s_{in}, t_1) \cup O(t_1, PN_i) \mid i= 1,2,3,4,\ldots\ldots\ldots\ldots, k\}$$

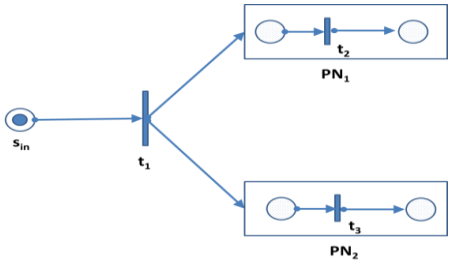

Fig. 3. Split model

D. The Split-Join Process

The Petri Nets model for split-join composition of web services is a hierarchical Petri nets $SJPN = <S, PN, I,O>$ as shown in Fig. 4, where,

$$S = \{S_{in} , S_{out}\}$$
$$T = \{t_1, t_2, t_3, t_4\}$$
$$PN = \{PN_1, PN_2, PN_3\ldots\ldots\ldots\ldots PN_k\},$$

where PN_i is a subset of i-th service.

$$I = I(s_{in}, t_1) \cup I(t_1, PN_i) \cup I(PN_i, t_2) \cup I(t_1, s_{out}),$$
$$i = 1, 2, 3\ldots k$$

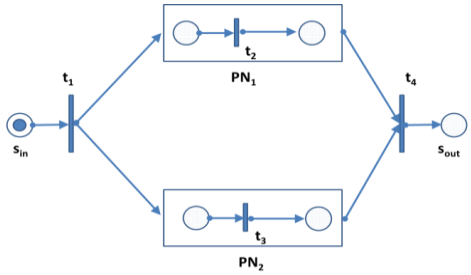

Fig. 4. Split-Join model

In this paper we briefly discuss some feature of BPEL and Perti Nets for more details about BPEL and Petri Nets see [1]-[5].

IV. BPEL

Business Process Execution Language for Web Services (BPEL or BPEL4WS) is a language used for the definition and execution of business processes using Web services. BPEL enables the top-down realization of Service Oriented Architecture (SOA) through composition, orchestration, and coordination of Web services. BPEL provides a relatively easy and straightforward way to compose several Web services into new composite services called business processes. Process of Petri nets is called activity in BPEL.

A. Atomic Activity

BPEL has several Atomic activities such as **receive, invoke, replay** etc. The receive activity is written as:-

<receive>……</receive>
<reply>……</replay>
<invoke>……</invoke>

B. Sequence

The BPEL model for sequence composition of web service is shown in Fig. 5. **<*sequence*>**tag is used for representing ***sequence*** activity. The structure of ***sequence*** in BPEL is:-

<sequence>
<receive>……..</receive>
……
<invoke>……..</invoke>
</sequence>

C. Split

The BPEL model for split/ parallel composition of web service is shown in Fig. 5. In BPEL **<*flow*>** tag is used for Split composition.

<flow>
<receive>……..</receive>

```
<replay>……</replay>
……
</flow>
```

D. The Split-Join

The split-join composition of web service can be shown in BPEL by adding <**sequence**> and <**flow**> tags together. Figure-5 shows the split-join activity in BPEL.

```
<sequence>
<receive>.....……</receive>
<flow>
<replay>……</replay>
<invoke> ……</invoke>
……
</flow>
<invoke> …… </invoke>
<replay> …… </replay>
……..
……..
</sequence>
```

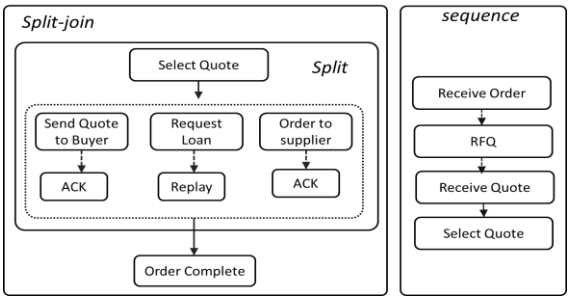

Fig. 5. Sequence, split and split-join in BPEL

V. Car Broker Web Service

A car broker web service negotiates car purchases for its buyers and arranges loans for these. The car broker uses two separate web services: a Supplier to find a suitable quote for the requested car model and a Lender to arrange loans. Each web service can operate separately and can be used in other web services. In this case study, our focus is on how the processes communicate with each other. For brevity, several details are abstracted from the description. The original car broker example can be found in [9], [10].

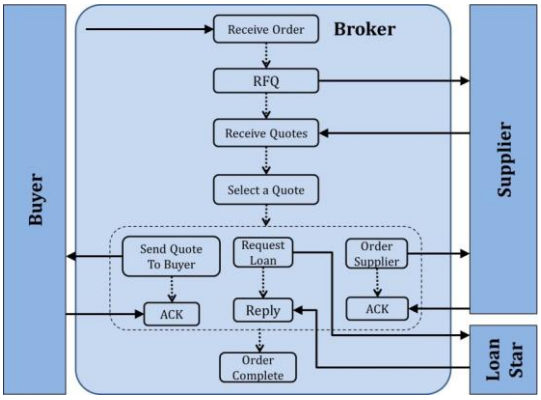

Fig. 6. Architectural view of car broker web service [11]

A. Broker Web Service

We model a car broker web service Broker. It provides online support to customers to negotiate car purchases and arranges loans for these. A buyer provides a need for a car model. The broker first uses its business partner Supplier to find the best possible quote for the requested model and then uses another business partner LoanStar to arrange a loan for the buyer for the selected quote. The buyer is also notified about the quote and the necessary arrangements for the loan. The process should be completed by talking the confirmation from buyer. We first model this web service using BPEL and then in Petri Nets. The behavior of the Broker we service in relation to BPEL modeling is illustrated in Fig. 6.

Composition of Car Broker Web Service using BPEL shows here:-

```
<process name="CarBroker".../>
<sequence>
<invoke partnerLink="BrokerPL"
operation ="cencelOrder"...../>
</sequence>
<sequence>
<receive partnerLink="order_Broker",
Variable="orderReq"...../>
.....
…….
<sequence>
<invoke partnerLink="RFQ_Suppiler",
outputVariable="SuppilerQuote",
inputVariable="orderReq".../>
<reply partnerLink="Quote_Broker",
variable="SuppilerQuote".../>
</sequence>
<flow>
<sequence>
<invoke partnerLink="BrokerPL"
operation ="cencelLoan"...../>
</sequence>
<sequence>
<invoke partnerLink="ReqLoan_loanstar",
outputVariable="Reply",
inputVariable="SupplierQuote"..../>
<reply partnerLink="Reply_broker",
variable="Reply"...../>
</sequence>
……....
……
<sequence>
.... ..
…
</sequence>
.....
</process>
```

BPEL construct sequence is defined to arrange the services in sequential order; flow is used to model the tasks in parallel. Due to brevity, a limited part of the BPEL is presented here. The Petri Nets (graphical) representation of the Broker web service is defined in Fig. 7.

According to the Petri Nets, the Car Broker Web Service can be described as,

$$Car = < Place, Tran, In, Out >$$

where $Place = \{Buyer, Broker, Supplier, LoanStar\}$

Since Broker has some sub place so,

$$Broker = \{b_1, b_2\text{……..}, b_{11}\}$$
$$Tran = \{B_1, t_1 - t_8, S_0, L_0\}$$
$$In = \{(B_1, b_1), (B_1, b_1), (b_1,t_1), (t_1,b_2), (b_2,t_2)\text{…….}, (t_8, b_7)\text{,……..}, \}$$
$$Out = \{(Buyer, B_1), \text{…….}, (L_0, LoanStar), (S_0, Supplier)\}$$

The Broker received a token from Buyer (the token contains information about car). After receiving the token Broker add more parameter with the token and send it to Supplier. Supplier also adds some parameter with the token

and return back it to Broker. After that, the Broker simultaneously send the token to LoanStar, Buyer and Supplier. The token has different meanings. LoanStar receives the token as request for Loan, Buyer receives it as a notification for confirmation and Supplier receives it as a delivery order. The process is terminated when all of those tokens are return back to Broker.

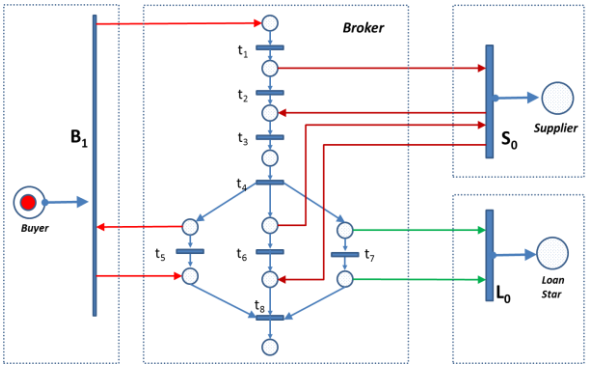

Fig. 7. The Petri Nets representation of car broker web service

VI. Conclusion

Having been able to model the web services both by using Petri nets and BPEL confirms suitability of a proper formal verification of the web service composition. As this composition is crucial to business organizations our comparative case study made a significant leap onwards the development of a verified web services.

References

[1] *Web Services Business Process Execution Language Version 2.0*, OASIS, 2007.
[2] P. J. Petri, *Net Theory and Modeling of Systems*, Prentice Hall, 1981.
[3] P. C., "Kommunikation mit automation," PhD thesis, University of Bonn, Germany, 1962.
[4] R. W. Petri Nets, "An introduction, EATCS monographs" in *Theoretical Computer Science*, vol. 4, Berlon: Springer-Verlag, 1985.
[5] M. T., "Petri Nets: properties, analysis and applications", in *Proc. the IEEE*, vol.77, 1989, pp. 541-580.
[6] N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg, "Analyzing interacting BPEL processes," in *Proc. Int. Conf. Business Process Management*, 2006, pp. 17–32.
[7] C. Ouyang, E. Verbeek, W. M. P. V. D. Aalst, S. Breutel, M. Dumas, and A. H. M. t. Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," *Science of Computer Programming*, vol. 67, issue2-3, pp. 162-198, 2007.
[8] S. Hinz, K. Schmidt, and C. Stahl, "Transforming BPEL to petri- nets," in *Proc. Int. Conf. Business Process Manage*, 2005, pp. 220–235.
[9] K. J. Turner, "Formalising web services," in *Formal Techniques for Networked and Distributed Systems - FORTE 2005*, F. Wang, Ed., Springer-Verlag, October 2005, pp. 473–488.
[10] K. J. Turner, "Representing and analysing composed web services using CRESS," *Network and Computer Applications*, vol. 30, no. 2, pp. 541–562, 2007.
[11] S. Ripon, M. Salah Uddin and A. Barua, "Web service composition: BPEL vs cCSP process algebra," in *proc. International Conference on Advance Computer Science and Technology*, ACSAT 2012, pp.35.

Mohammad Salah Uddin is a lecturer in the Department of Computer science and Engineering, Central Women's University, Dhaka, Bangladesh He published several journal and international conference paper from his research work. He is a member of IACSIT and Software Engineering and Formal Method Research Group of East West University. He received his B.Sc in Computer Science and Engineering from East West University, Dhaka, Bangladesh in 2012. He is interested in:- Web Service Composition, Semantic Web service, knowledge representation, Mobile apps, Software Engineering, Software Product Line, Modeling and Verification.

Shamim Ripon is an associate professor in the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh where he leads Software Engineering and Formal Method Research Group. Previously, he was a Research Associate in the Department of Computing Science, University of York, UK and Research Fellow in the Department of Computing Science, University of Glasgow, UK. He also served as a Lecturer in Khulna, University, Bangladesh. He is a member of IAENG, Senior member of IACSIT.

Dr. Ripon holds a B.Sc. in Computer Science and Engineering from Khulna University, MSc in Computer Science from National University of Singapore and PhD in Computer Science from University of Southampton, UK. His research interests focus on the Requirement Engineering, Software Product Line, Semantic Web, Natural Language Processing. His current research examines the formal representation and verification of knowledge based requirement specification.

Nakul Chandra Das completed his B.Sc in Computer Science and Engineering from East West University, Dhaka, Bangladesh in 2013. He is a member of Software Engineering and Formal Method Research Group, East West University. He is interested in:- Web Service Composition, Semantic Web service, Software Engineering, and Software Product Line.

Orin Hossain completed her B.Sc in Computer Science and Engineering from East West University in 2013. She is a member of Software Engineering and Formal Method Research Group, East West University. She is currently working towards formal verification of web services composition. She is also interested in semantic web based requirement analysis of product line requirements.