International Journal of Computer Science Research (IJCSR)

FOREX Publication

Multi-disciplinary | Open Access | ISSN: 2347-47XX

Date of Publication: 31 March 2017 | Volume 5, Issue 1 | Pages: 1-5 | ID: IJCSR-050101 | DOI: N.A.

Checking of Web Services Composition through Online Movie Ticket Booking Example

Saurabh Agarwal, M.Tech (CSE) Scholar, SRCEM Banmore, (M.P.) India, toc.saurabh@gmail.com

Prof. Nirupama Tiwari, Assistant Professor in Dept. of CSE/IT, SRCEM Banmore, (M.P.) India, girishniru@gmail.com

Prof. Madhukar Dubey, Assistant Professor in Dept. of CSE/IT, SRCEM Banmore, (M.P.) India, madhukardubey@hotmail.com

ABSTRACT: Web services are ready mate services which is easily provided by various respected vendors. Web services are the basic unit of service oriented architecture (SOA). They are software applications that are implemented and published by web service providers and invoked by web service requesters over a network. The main purpose of web service technologies is to allow applications on different platforms to exchange business data. One the ad-vantages of service oriented architecture is web services composition; we can compose existing web services to create new web services. It has two types of property- first is Functional properties and other is Non-Functional properties. Functional properties show the basics detail of respected web services and Non-Functional show the properties which are used to identify the requirement of the users. This paper introduces a formal method for describing and verifying web services composition using Pi-Calculus. In agent based web services composition, agent accepts request from web service consumers and searches Universal description and discovery (UDDI) registry against the requirements of the consumer to find appropriate web services. By using in-formation provided by UDDI, agent invokes those web services. So this composition frees the web service requester to search the UDDI registry for required web services and then manually invoke those services. Before implementing a web services composition, it should be verified for correctness to improve the reliability. Formal methods are mathematical based techniques used for specification and verification of software systems. Pi-Calculus is a kind of process algebra which is used to model concurrent systems with mobility. In this work, we have formally described web services composition using Pi-Calculus. For verification we have used mobility workbench (MWB) tool.

Keywords: Web Services Composition, Mobility Workbench, Universal description and discovery (UDDI).

1. INTRODUCTION

Web services are the programs that can be accessed over internet. Web services are platform and language independent, they can use any operating system and programming language. The main components of web service architecture are service provider, service requester and service registry. Service provider implements web services and publishes information required to access those services. Service registry (also called UDDI registry) is used to store this information, any service requester can find information to access a particular web service from this registry. Reusability is one of the advantages of web services. We can compose two or more web services which already exist; it reduces the efforts of implementing new web services. Sometimes it is necessary to compose two or more web services because a single service is not sufficient for some complex applications.

Every web services composition algorithm should be validated before implementation for its correctness. We have used Pi-Calculus [4] for describing and modeling web service composition; for verification process we have used MWB [6].

2. RELATED WORK

In recent years several methods have been used to model web service composition. In [1], this paper proposes a method which can extract the model from a BPEL process and analyze it through probabilistic model checking with Prism model checker. In [2], Web Services have 2 types of properties Functional and Non Functional. This Paper mainly focus on Nonfunctional properties of web services. Modeling using FSM and checking through UPPAAL tool. In [3], This paper present sour finding on how NFR have been supported in the development of service-based applications by proposing a classification scheme cons is ting in five facets: (i) programming paradigm (object / service oriented); (ii) contribution (methodology, system, middleware); (iii) software process phase; (iv) technique or mathematical model used for expressing NFR; and (v) the types of NFR. CCS [10] is used to model concurrent systems, but it cannot be used for mobility. In [11], the method used is based on partial-order planning problems and uses first-order logic. In this method predicates are used to define every process specification of WSCDL, then these predicates are written in Prolog tool for verification. In [12], a method based on temporal logic of actions (TLA) is proposed. Web services are modeled as automata and described using TLA, then verified using TCL (a model checker of TLA). In [13], FSM (finite state automata) is used to describe the web service composition and it is translated into programs described by Promela and finally verified using model checking tool SPIN. In [14], Colored petri-nets (CPN) is used, it is similar to Petri-nets with an extra advantage that has programmable elements. In [15], this paper presents an overview of the different security framework for service based model proposed by different researchers across the globe.

We have used Pi-calculus; it is used to model concurrent systems with mobility. In pi-calculus communication links are transferred as names.

3. MODELLING WEB SERVICES COMPOSITION

Date of Publication: 31 March 2017 | Volume 5, Issue 1 | Pages: 1-5 | ID: IJCSR-050101 | DOI: N.A.

3.1 Agent based web services composition

In agent based web services composition, agent accepts requests from web service consumers and searches UDDI registry against the requirements of consumer to find appropriate web services. When it finds the services, it returns the essential information which is used to invoke required web services to the Client.
The major components of agent based web services composition are Client, Web service composition agent (WSCA), UDDI, and Service provider. Service Providers publish information about their web services in UDDI registry. When a client needs to access a web service, it sends a request to web service composition agent (WSCA). Then WSCA search UDDI registry for the services which can fulfill the requirements of the client. When it finds the required services in UDDI registry, it asks detailed requirement from client to invoke that service. When client send the information, WSCA invoke the required web services.

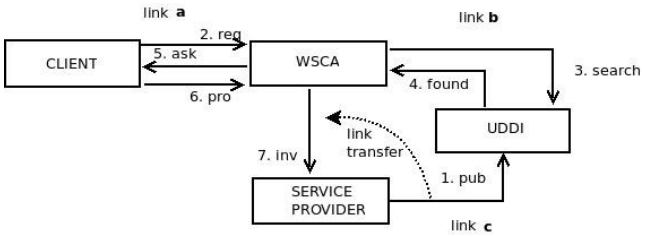

Figure 1: Agent based Web services composition

The different messages used are described as follows:

pub: Service providers publish their web services in UDDI registry.
req: Client sends a request to WSCA for accessing web services.
search: WSCA search UDDI registry for required web services.
found: UDDI registry send this message to WSCA when the required services are found.
ask: WSCA ask detailed requirements from client.
pro: Client sends detailed requirements.
inv: WSCA invoke the required web services with the information sent by Client.

3.1.1 Specification of Web service Composition

In specification process of a system, we describe the properties that the system must satisfy. We have used Pi-Calculus for specification of web services. Pi-Calculus is used to describe concurrent systems and it can also be used for mobility. Every component of the system is defined as an agent in Pi-Calculus. Here, a and b are the communication link of WSCA with client and UDDI respectively. Similarly, c is the communication link between UDDI and Service provider (SP). First, service provider publishes its service in UDDI by using link c. Client sends a request to WSCA over link a then WSCA sends a message to UDDI over link b to search the required services. If the services are found then UDDI send a message and also link c to WSCA. Now WSCA ask detailed requirements from client. By using this information, WSCA invoke service through link c.
The specification of different agents is given as follows:

Client(a) $=^{def}$ a req.a(ask).apro.Client(a)
WSCA(a,b,c) $=^{def}$ a(req).bsearch.b(found).b(c).aask.a(pro).cinv.WSCA(a,b,c)
UDDI(b,c) $=^{def}$ c(pub).b(search).bfound.bc.UDDI(b,c)
SP(c) $=^{def}$ cpub.c(inv).SP(c)
Composite Web service $=^{def}$ (ClientjWSCAjUDDIjSP)

3.1.2 Verification of web service composition

For verification process, we have used a tool mobility workbench (MWB) which is used to verify the specifications written in Pi-Calculus. In verification process, we have checked deadlocks for different agents, it ensures the correctness of web service composition.
The MWB code for different agents is given as follows:

```
agent Client(a,req,ask,pro)='a< req >.a(ask).'a< pro > .Client <a,
req ,ask,pro >
agent WSCA(a,req,b,search,found,ask,c,inv)=a(req).'b<search>
.b(found).b(c).'a<ask>.'c<inv>.WSCA<a,req,b,search,found,ask,
c,inv>
agent SP(c,pub,inv)='c<pub>.c(inv).SP<c,pub,inv>
agent
UDDI(c,pub,b,search,found)=c(pub).b(search).'b<found>.'b<c>.
UDDI<c,pub,b,search,found>
```

3.2 Web Service composition Examples

3.2.1 Online movie ticket booking system

(a) Description of web service composition

We have taken an example of movie ticket booking service, to describe the web service composition. The services used in this example are Client service, web service composition agent (WSCA), Registry service, Bank service and Multiplex service. Web service composition agent (WSCA) service accepts request from client service and searches the services which are required to fulfill the request in the Registry. If the required services found in registry than WSCA asks information from client, and with this information it invokes the multiplex service, otherwise it sends a refuse message to client. If required seats are available then Multiplex service invokes Bank service otherwise it sends a message to WSCA indicating that seats are not available, which is sent to client by WSCA. In case seats are available, Bank service sends a message to client for payment. When the amount is paid by client, Bank confirms by sending a message to multiplex service and Multiplex service sends this confirmation to WSCA, eventually which is sent to client.

FOREX Publication
Multi-disciplinary | Open Access | ISSN: 2347-47XX

Date of Publication: 31 March 2017 | Volume 5, Issue 1 | Pages: 1-5 | ID: IJCSR-050101 | DOI: N.A.

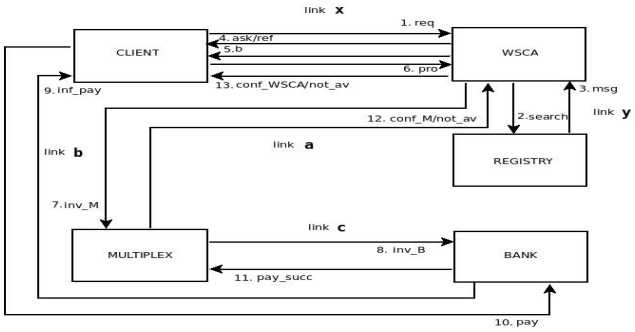

Figure 2: Web services composition model for online movie ticket booking system after transfer of link 'a' and 'b'.

The messages are described as follows:
pub M : Multiplex service publishes itself into Registry.
pub B : Bank service publishes itself into Registry.
req : Ticket booking request from Client.
search : message from WSCA to Registry for searching the required services.
msg : message from registry to WSCA that the services are found or not.
ask : WSCA asks the requirement of client.
pro : Client provides the detailed information such as name of movie, date and number of seats etc.
inv M : WSCA invokes multiplex service with information provided by client.
inv B : Multiplex service invokes Bank service.
inf pay : Bank service sends a message to client for paying the required amount.
pay : Client pays the required amount.
pay succ : Bank service informs multiplex service that payment has received.
conf M : Multiplex service sends confirmation to WSCA.
conf WSCA : WSCA sends confirmation to Client.
ref : WSCA sends this message to client when the required services to fulfill the request are not available in registry.
not av : Multiplex service sends this message to WSCA when required seats are not available, which is sent to client by WSCA.
The communication link of WSCA with Client and Registry are x and y respectively. Similarly, link of Registry with Multiplex and Bank are a and b respectively. Multiplex and Bank services communicate over link c. Each web service is described as an agent in Pi-Calculus and communication links are transferred as names in this model. If the required services are found then Registry sends a message with links a and b to WSCA otherwise sends a message indicating required services are not found. If services are found then WSCA sends a message asking the detailed information about movie ticket booking and also link b to Client. If services are not found in Registry then WSCA sends a refuse message to Client. When Client sends the detailed information such as name of movie, number of seats, show time etc., with this formation WSCA invokes the Multiplex service by using the communication link a. If required seats are available then multiplex service invokes Bank service using link c otherwise it sends a message to WSCA indicating that seats are not available, which is sent to client by WSCA. In case seats are available, Bank service sends a message over link b to client for payment. When the amount is paid by client, Bank confirms by sending a message to multiplex service and Multiplex service sends this confirmation to WSCA, eventually which is sent to client.
The description of different agents in pi-calculus is given as follows:

Bank (b,c) =def bpub B.c(inv B).binf pay.b(pay). cpay succ.Bank(b,c)

Multiplex(a,c) =def apub M.a(msg).([msg=inv M] anot av.Multiplex(a,c) +[msg=inv M]c inv B.c(pay succ).aconf M. Multiplex(a,c))

Registry(a,b,y) =def a(pub M):b(pub B):y(search):ymsg.y a.y b.Registry(a,b,y)

Client(x,a) =def xreq.x(msg1).([msg1=ask]x(b).xpro.x(msg2). ([msg2=not av] 0+b (inf pay). bpay.x(conf WSCA).Client (x,a)) + [msg1=ref] 0)

WSCA(x,y,a) =def x(req):ysearch.y(msg1).([msg1=found]y(a).y(b). xask. xb.x(pro) .ainv.a(msg2).([msg2=not av]xnot av.WSCA(x,y,a) +[msg2 = conf M]xconf WSCA.WSCA(x,y,z)) + [msg1=not found] xref.WSCA(x,y,a))

Composite Web service =def (Bank|Multiplex|Registry|Client| WSCA)

(b)Verification of web service composition

The MWB code for different agents is given as follows:
agent Bank(b,pub_B,c,inv_B,inf_pay,pay,pay_succ) = 'b<pub_B>.c(inv_B) .b(inf_pay).b(pay).'c<pay_succ>.Bank < b,pub_B,c,inv_B ,inf_pay,pay, pay _succ >

agent Multiplex(a, pubM, msg, invM, notav, c, invB, confM, paysucc) =' a < pubM > .a(msg).([msg = invM]'a < notav > .Multiplex < a, pubM, msg, invM, notav, c, invB,confM, paysucc > +[msg = invM]'c < invB > .c(paysucc).'a < confM > .Multiplex< a, pubM, msg, invM, notav, c, invB, confM, paysucc >)

agent Registry(a, pubM, b, pubB, y, search, msg) = a(pubM).b(pubB) .y(search).'y < msg > .'y < a > .'y < b > .Registry < a, pubM, b, pubB, y, search, msg >

agent Client(x, req, msg1, ask, b, pro, msg2, notav, infpay, pay, conf_WSCA, ref) =' x < req > .x(msg1).([msg1 = ask]x(b).'x < pro > .x(msg2). ([msg2 = notav]0 + b(infpay).'b < pay > .x(conf_WSCA).Client < x, req, msg1, ask, b, pro, msg2, notav, infpay, pay, conf_WSCA,ref >) + [msg1 = ref]0)

agent WSCA(x,req, y, search, msg1, found, ask, b, pro, a, invM, msg2, notav, confM,conf_WSCA,notfound, ref) = x(req).'y < search > .y(msg1).([msg1 = found]y(a).y(b).'x < ask > .'x < b > .x(pro).'a < invM > .a(msg2).([msg2 = notav]'x < notav > .WSCA < x,req, y, search, msg1,found, ask, b, pro, a, invM, msg2, notav, confM,conf_WSCA, notfound, ref > +[msg2 = confM]'x < conf_WSCA > .WSCA < x,req, y, search, msg1, found, ask, b, pro, a, invM, msg2, notav,confM,conf_WSCA, notfound, ref >) + [msg1 = notfound]'x < ref > .WSCA < x,req, y, search, msg1,found, ask, b, pro, a, invM, msg2, notav, confM,conf_WSCA, notfound, ref >)

FOREX Publication
Multi-disciplinary | Open Access | ISSN: 2347-47XX

Date of Publication: 31 March 2017 | Volume 5, Issue 1 | Pages: 1-5 | ID: IJCSR-050101 | DOI: N.A.

4. ONLINE BOOK PURCHASE SYSTEM

4.1 Description of web service composition

The services used by this system are client service, Book shop service, Database service and Delivery service (fig. 3). Here x and w are communication links of Client service with Book shop service and Delivery service, respectively. Similarly, y and z are communication links of Book shop service with Database service and delivery service respectively. First, Client service sends a request to Book Shop service for purchasing a book by specifying required information such as name of book and author's name. Book shop service then sends a lookup message to database service whether the requested book is available or not. Database service sends a message to Book Shop service based on availability of that book.

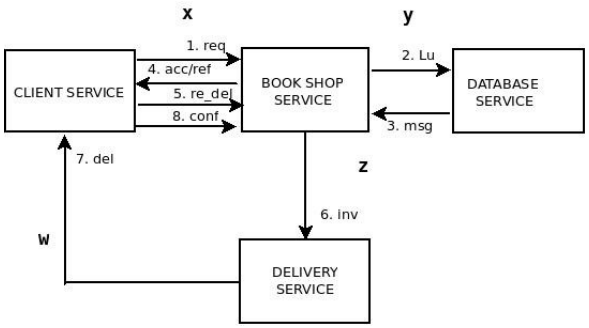

Figure 3: Web services composition model for online book purchase system

Now Book Shop service replies to Client service about the requested book by sending accept or refuse message. If book is available then it sends accept message otherwise sends refuse message. Now Client service confirms its request by sending a request delivery message. This message contains information about client i.e. address and phone number etc. After receiving this message, Book Shop service invokes delivery service with appropriate information. Delivery service then delivers the book to client and client sends a confirmation message to Book Shop service. For simplicity, we assumed that amount for book is paid at the time of delivery i.e. in person.

The various messages used in this system are described as follows:

req: Client service requests for a book.
Lu: Book Shop service sends this message to Database service to check the availability of that book.
msg: Database service sends availability information to Book Shop service.
acc: Book Shop service sends this message to client service if book is available.
ref: Book Shop service sends this message to client service if book is not available.
req del: Client confirms its request by sending its information such as address and phone number etc.
inv: Book Shop Service invokes delivery service.
del: Delivery service delivers the required book to client.
conf: Client service sends confirmation to Book Shop Service.

The description of different agents in pi-calculus is given as follows:

Client service : CS(x,w) $=^{def}$ xreq.x(msg).([msg=acc] xdel book. w(del).x conf.CS(x,w) + [msg=ref]0)
Book shop service :BSS(x,y,z) $=^{def}$ x(req):ylu.y(msg).([msg=found]x acc. x(req del).zinv.x(conf).BSS(x,y,z) + [msg=not found]z ref.BSS(x,y,z))
Delivery service : DS(z,w) $=^{def}$ z(inv):wdel.DS(z,w)
Database service : DBS(y) $=^{def}$ y(lu):ymsg.DBS(y)
Composite Web service$=^{def}$ (CS | BSS | DS | DBS)

4.2 Verification of web service composition

The MWB code for different agents is given as follows:

agent CS(x; req; msg; acc; del book; del; conf; ref) =0 x < req > :x(msg):([msg = acc]0x <del book > :w(del):x < conf > :CS < x; req; msg; acc;Del book; del; conf; ref > +[msg = ref]0)

agent BSS(x; req; y; lu; msg; found; acc; req del; z; inv; conf; not found; ref) = x(req):0y < lu > :y(msg):([msg = found]x < acc > :x(reqdel):0z < inv > :x(conf):BSS < x; req;y; lu; msg; found; acc; req del; z; inv; conf; not found; ref > +[msg = not found]0x < ref > :BSS < x; req; y; lu; msg; found; acc; req del; z; inv; conf; not found; ref >)

5. CONCLUSION

Formal description and verification is necessary before implementing a web service composition algorithm because it ensures the correct working of composition. Non- Functional Requirement is very important for web services composition which is used to check the correctness of our requirement. Online movie ticket booking example is used to show the modeling process. The Pi calculus is useful to describe formally this system and the concurrent actions of the processes. This paper describes the model using formal approach Pi calculus and checks correctness of model with the help of MWB.

REFERENCES

[1] Pat. P.W. Chan, Michael R. Lyu. DynamicWeb service composition: A new approach in building reliable web services. In 22nd international conference on Advance information networking and applications, 2008.

[2] Snehit Prabhu. Towards distributed dynamic web service composition. In eighth international symposium on autonomous decentralized systems (ISADS07), 2007.

[3] R. Miler. The polyadic pi-calculus: a tutorial. In research report ECS-LFCS-91-180, University of Edinburgh, October 1991.

[4] J. Parrow. An introduction to the pi-calculus. In Handbook of Process Algebra, ed. Bergstra,Ponse, Smolka, pages 479-543, Elsevier, 2001

[5] R. Milner, J. Parrow and D. Walker. A calculus of mobile process (part 1 and 2). Journal of information and computing, 100:1-77, September 1992.

[6] Bjorn Victor and Faron Moller. The Mobility Workbench: A tool for the pi-calculus. In Uppsala University and University of Edinburgh, February 1994.

Date of Publication: 31 March 2017 | Volume 5, Issue 1 | Pages: 1-5 | ID: IJCSR-050101 | DOI: N.A.

[7] Hongbing Wang, Chen Wang, Yan Liu. A logic based approach to web service composition and verification. In Second world conference on services, 2009.

[8] Zhao Wei, Rongsheng Dong, Xiangyu Luo and Fang Liu. Model Checking Airline Tickets Reservation System Based On BPEL. In Third International Conference on Genetic and Evolutionary Computing, 2009.

[9] Hongbing Wang, Qianzhao Zhou, Yanqi Shi. Describing and Verifying Web Service Composition using TLA Reasoning. In IEEE International Conference on Services Computing, 2009.

[10] Koshkina M., van Breugel F. Modeling and verifying Web service orchestration by means of the concurrency workbench. In ACM SIGSOFT SEN, 29(5):1-10, 2004.

[11] Li Bao, Weishi Zhang, Xiuguo Zhang. Describing and Verifying web service using CCS. In Seventh International conference on parallel and distributed computing, applications and technologies (PDCAT 06), 2006.

[12] Zhang Jia, Chung Jen Yao, Chang C. K., Kim S. WS-Net: A petri-net based specification model for web services. In Second IEEE conference on Web Services, San Diego, California,USA, 420-427,2004.

[13] Web Service definition by W3C. http://www.w3.org/TR/ws-gloss/.

[14] Web service definition by IBM. http://xml.coverpages.org/ni2001-02-19-b.html.

[15] Hitesh Seth. Microsoft .Net: Kick start, page 285.

[16] PlácidoA.SouzaNeto,Designing service based applications in the presence of non-functional properties: A mapping study, 0950-5849/© 2015 Elsevier B.V. Information and Software Technology 69 (2016) 84–105.

[17] Yuemin Li, Shenghui Zhao, Hailun Diao and Haibao Chen, A Formal Validation Method for Trustworthy Services Composition, 978-1-4673-9803-9/15 $31.00 © 2016 International Conference on Networking and Network Applications.

[18] Hongxia Tong, Jian Cao, Shensheng Zhang, and Minglu Li, A Distributed Algorithm for Web Service Composition Based on Service Agent Model, 1045-9219/11/$26.00 © 2011 IEEE, IEEE Transactions On Parallel And Distributed Systems, Vol. 22, No. 12, December 2011.

[19] Chengyang Mi, Huaikou Miao, Jinyu Kai, Honghao Gao, Reliability Modeling and Verification of BPEL-Based Web Services Composition by Probabilistic Model Checking, 978-1-5090-0809-4/16/$31.00 copyright 2016 IEEE SERA 2016, June 8-10, 2016, Baltimore, USA.

[20] Ilyass EL KASSMI, Zahi JARIR, Security Requirements in Web Service Composition: Formalization, Integration and Verification, 978-1-5090-1663-1/16 $31.00 © 2016 IEEE DOI 10.1109/WETICE.2016.47.

[21] Schahram Dustdar and Wolfgang Schreiner, A survey on web services composition, Int. J. Web and Grid Services, Vol. 1, No. 1, 2005.

[22] Bandita Sahoo, Prachet Bhuyan, A SELECTION APPROACH IN SERVICE COMPOSITION OF SOA, 978-1-4673-9802-2/16/$31.00© 2016 IEEE 2016 Fifth International Conference On Recent Trends In Information Technology.

[23] Abdul Muttalib khan, Alankar Mishra, Riya Agarwal, Security Framework based on QoS and Networking for Service Oriented Architecture, 978-9-3805-4421-2/16/$31.00_c 2016 IEEE, 2016 International Conference on Computing for Sustainable Global Development (INDIACom).

[24] Hongbing Wang, Peisheng Ma, Qi Yu, Danrong Yang, Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware services composition,0743-7315/© 2016 Elsevier inc. J.Parallel Distrib. Comput. 100(2017) 71-84.

[25] Aissam Belghiat, Allaoua Chaoui and Mokhtar Beldjehem, Capturing and Verifying Dynamic Systems Behavior Using UML and π-Calculus, Theoretical Information Reuse and Integration, Advances in Intelligent Systems and Computing 446, DOI 10.1007/978-3-319-31311-5_3.