DESCRIBING AND VERIFYING WEB SERVICES COMPOSITION USING PI-CALCULUS

Saurabh Agrawal[1], Sumit Tiwari[2], Sumit Pathak[3]
[1,2]P. G. Scholar, CSE Dept., SRCEM Banmore,(M.P.) India
[3]Asst. Prof., Department of Computer Science,IPS-CTM Gwalior, India
Email: toc.saurabh@gmail.com[1], sumitgwalior23@gmail.com[2], sumitpathakcs@gmail.com[3]

ABSTRACT

This paper introduces a formal method for describing and verifying web services composition using Pi-Calculus. 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 of the ad-vantages of service oriented architecture is web services composition; we can compose existing web services to create new web services. 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.

Key Words: 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. CCS [7] is used to model concurrent systems, but it cannot

be used for mobility. In [8], 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 [9], 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 [10], 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 [11], Colored petri-nets(CPN) is used, it is similar to Petri-nets with an extra advantage that has programmable elements.

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

3. ONLINE MOVIE TICKET BOOKING EXAMPLE

a. Description of web services composition

We have taken an example of movie ticket booking service to describe the web services 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 then 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.

The messages are described as follows:

Publish_Multiplex (Pub_M) : Multiplex service publishes itself into Registry.

Publish_Bank (Pub_B) : Bank service publishes itself into Request 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_Info (ask) : WSCA asks the requirement of client.

Provide_Info (pro) : Client provides the detailed information such as name of Movie, date and number of seats etc.

Invoke_Multiplex (Inv_M) : WSCA invokes multiplex service with information provided by client.

Invoke_Bank (Inv_B) : Multiplex service invokes Bank service.

Inform_pay (Inf_pay) : Bank service sends a message to client for paying the required amount.

Pay : Client pays the required amount.

Pay_succsess (pay_succ) : Bank service informs multiplex service that payment has received.

Confirmation_Multiplex (Conf_M) : Multiplex service sends confirmation to WSCA.

Confirmation_WSCA (Conf_WSCA) : WSCA sends confirmation to Client.

Refuse (ref): WSCA sends this message to client when the required services to fulfill the request are not available in -registry.

Not_Available (not_av): Multiplex service sends this message to WSCA when required seats are not available, which is sent to client.

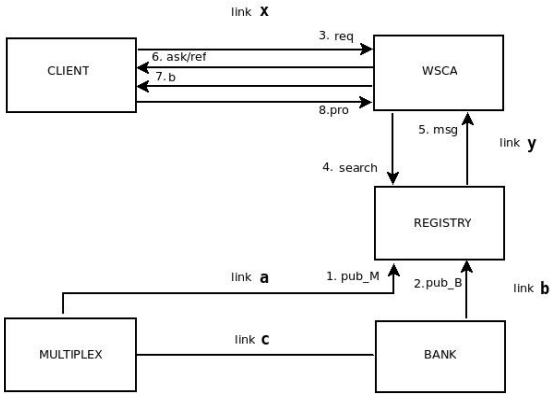


Figure 1: Web services composition model for online movie ticket booking system

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

Bank(b,c) $\bar{b}$ pub_B.c (inv_B). $\bar{b}$ inf_pay.b(pay). $\bar{c}$ pay_succ.Bank (b,c)

Multiplex(a,c)= $\bar{a}$ pub_M.a(msg).([msg=inv_M] $\bar{a}$ not_av.Multiplex(a,c)+[msg=inv_M] $\bar{c}$ inv_B.c(pay_succ). $\bar{a}$ conf_M.Multiplex(a,c))

Registry(a,b,y)=a(pub_M).b(pub_B).y(search). $\bar{y}$ msg. $\bar{y}$ a. $\bar{y}$ b.Registry(a,b,y)
Client(x,a)= $\bar{x}$ req.x(msg1).([msg1=ask]x(b). $\bar{x}$ pro.x(msg2).([msg2=not_av]0+b(inf_pay). $\bar{b}$ pay.x(conf_WSCA).Client(x,a)) + [msg1=ref]0)

WSCA(x,y,a)=x(req). $\bar{y}$ search.y(msg1).([msg1=found]y(a).y(b). $\bar{x}$ ask. $\bar{x}$ b.x(pro). $\bar{a}$ inv.a(msg2).([msg2=not_av] $\bar{x}$ not_av.WSCA(x,y,a)+[msg2=conf_M] $\bar{x}$ conf_WSCA.WSCA(x,y,z))+[msg1=not_found] $\bar{x}$ ref.WSCA(x,y,a))

b. Verification of web services 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 (nv_B).’b(inf_pay).b(pay).’c<pay_succ>.Bank<b,pub_B,c,inv_B,inf_pay,pay,pay_succ>

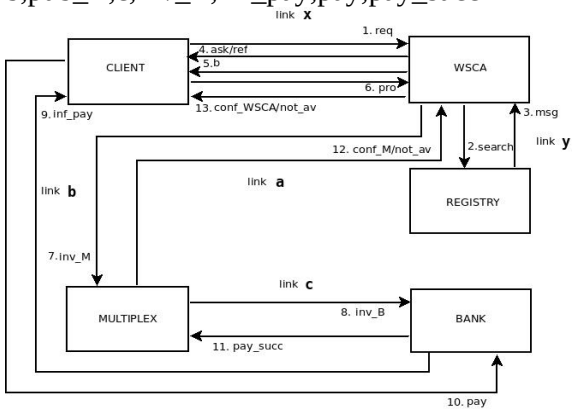


Figure 2: Web services composition model for online movie ticket booking system after transfer of link ‘a’ and ‘b’.
agent
Multiplex(a,pub_M,msg,inv_M,not_av,c,inv_B,conf_M,pay_succ)=’a<pub_M>.a(msg).([msg=inv_M]’a<not_av>.Multiplex<a,pub_M,msg,inv_M,not_av,c,inv_B,conf_M,pay_succ>+[msg=inv_M]’c<inv_B>.c(pay_succ).’a<conf_M>.Multiplex<a,pub_M,msg,inv_M,not_av,c,inv_B,conf_M,pay_succ>)

agent
Registry(a,pub_M,b,pub_B,y,search,msg)=a(pub_M).b(pub_B)y(search). ’y<msg>.’y<a>.’y<b>.Registry<a,pub_M,b,pub_B,y,search,msg>

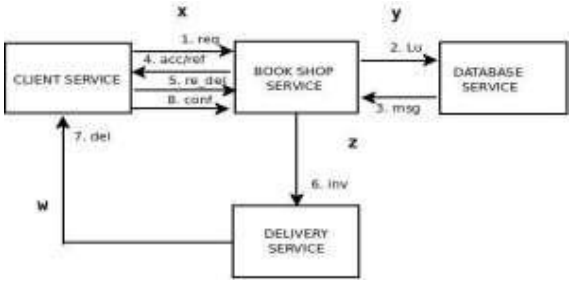
agent
Client(x,req,msg1,ask,b,pro,msg2,not_av,inf_pay,pay,conf_WSCA,ref)=’x<req>.x(msg1).([msg1=ask]x(b).’x<pro>.x(msg2).([msg2=not_av]0+b(inf_pay).’b<pay>.x(conf_WSCA).Client<x,req,msg1,ask,b,pro,msg2,not_av,inf_pay,pay,conf_WSCA,ref>)+[msg1=ref]0)
agent
WSCA(req,y,search,msg1,found,ask,b,pro,a,inv_M,msg2,not_av, conf_WSCA,not_found,ref)=x(req).’y<search>.y(msg1).([msg1=found]y(a).y(b).’x<ask>.’x<b>.x(pro).’a<inv_M>.a(msg2).([msg2=not_av]’x<not_av>.WSCA<req,y,search,msg1,found,ask,b,pro,a,inv_M,msg2,not_av,conf_WSCA,not_found,ref>+[msg2=conf_M]’x<conf_WSCA>.WSCA<req,y,search,msg1,found,ask,b,pro,a,inv_M,msg2,not_av,conf_WSCA,not_found,ref>)+[msg1=not_found]’x<ref>.WSCA<req,y,search,msg1,found,ask,b,pro,a,inv_M,msg2,not_av,conf_WSCA,not_found,ref>)

In verification process, we have checked deadlocks for different agents, it ensures the correctness of web services composition algorithm.

4. ONLINE BOOK PURCHASE SYSTEM

a. 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 authors 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 avail-ability of that book.



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 de-livery 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 avail-ability 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) = xreq.x(msg). ([msg=acc] xdel book. w(del).x conf.CS(x,w) + [msg=ref]0)BSS(x,y,z) + [msg=not found]zref.BSS(x,y,z))

Delivery service : DS(z,w) = z(inv):wdel.DS(z,w)

Database service : DBS(y) = y(lu):ymsg.DBS(y)

Composite Web service = (CSjBSSjDSjDBS)

b.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]^0 x < 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):^{0} y < lu > :y(msg):([msg = found]x < acc > :x(reqdel):^{0} z < inv > :x(conf):BSS < x; req; y; lu; msg; found; acc; req del; z; inv; conf; not found; ref > +[msg = not found]^0 x < ref > :BSS < x; req; y; lu; msg; found; acc; req del; z; inv; conf; not found; ref >)agent DS(z; inv; w; del) = z(inv):^{0} w < del > :DS < z; inv; w; del > agent DBS(y; lu; msg) = y(lu):^{0} y < msg > :DBS < y; lu; msg >

In verification process, we have checked deadlocks for different agents through MWB, it ensures the correctness of web services composition algorithm.

5. CONCLUSION

Formal description and verification is necessary before implementing a web service composition algorithm because it ensures the correct working of composition i.e. there is no deadlock. 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, Dynamic Web service composition : A new approach in building reliable web services, 22nd international conference on Advance information networking and applications, 2008.

[2] Snehit Prabhu, Towards distributed dynamic web service composition, eighth international symposium on autonomous decentralized systems(ISADS'07), 2007.

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

[4] J. Parrow, An introduction to the pi-calculus, Dep. Teleinformaties, Royal institute of technology, Stockholm.

[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, Uppsala University and University of Edinburgh, February 1994.

[7] Li Bao, Weishi Zhang, Xiuguo Zhang , Describing and verifying web services using CCS, proceedings of the seventh international conference on parallel and distributed computing, applications and technologies, 2006.

[8] Zahra Madani, Naser Nematbakhsh, A logical formal model for verification of web service choreography, Twelfth international conference on computer and information technology, December, 2009.

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

[10] Zhao Wei, Rongsheng Dong, Xiangyu Luo, Fang Liu, Model checking Airline tickets reservation system based on BPEL, third international conference on genetic and evolutionary computing, 2009.