FORMALIZING BPEL-TC THROUGH Π-CALCULUS

Preeti Marwaha[1], Hema Banati[2] and Punam Bedi[1]

[1]Department of Computer Science, University of Delhi, New Delhi, India
preeti_andc@yahoo.com, pbedi@cs.du.ac.in

[2] Department of Computer Science, Dyal Singh College, University of Delhi, New Delhi, India
bantihema@hotmail.com

ABSTRACT

WS-BPEL is way to define business processes that interact with external entities through web service operations using WSDL. We have proposed BPEL-TC, an extension to existing WS-BPEL which uses temporally customized Web Services (WSDL-TC) as a model for process decomposition and assembly. WSDL-TC handles both backward compatible and incompatible changes and also maintains various versions of the artifacts that results due to changes over time and customizations desired by the users. In this paper, we are using pi-calculus to formalize Business Process Execution Language- Temporal Customization (BPEL-TC) process. π -calculus is a model of computation for concurrent systems along with changing connectivity of interactive systems. Pi-calculus is an extension of the process algebra CCS, with added mobility to CCS while preserving its algebraic properties.

KEYWORDS

Web Services, WSDL, WSDL-TC, BPEL, BPEL-TC, π–Calculus, Temporally Customised Web Services

1. INTRODUCTION

A Web Service is a software application identified by a URI , whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols [1][2]. Web services are used to expose the functionalities provided by the service providers for the heterogeneous service consumers in a distributed and decentralized environment. Changes in the web services desired over time results in a new version of the service which is deployed at a new URL. Many a times, multiple customizations of the service is required to cater the need of multiple set of users. This forces service provider to deploy multiple Web Services customized for each set of users, which results in increasing cost of infrastructure and maintenance. Since, multiple versions of customized Web Services are deployed multiple times at different URLs, it is difficult and costlier to maintain, update and backup these services and their data. Extensions of WSDL [3,4] i.e. WSDL-T [5] and WSDL-TC [6] aim at reducing this cost by maintaining the different collaborative customized versions of operations of the Web Service in a single deployment. The approach also manages access control of these operations to their respective groups. WSDL-TC being an extension of WSDL-T is capable of managing versions of each customized operation that resulted due to the changes in their business requirements over a period of time. WSDL-TC also eases the task of Web Service administrators as they have to manage the single instance instead of multiple instances of Web Services. WSDL-TC defines different versions of the artifacts of the Web Services customized for different group of users referred as Entities. By using WSDL-TC, it is possible to customize any valid version of an artifact, available at a particular time for any client. This enables the service producer to create customized functionality within a service for each Entity. Through WSDL-TC, it is possible to customize temporal Web Service for multiple Entities running from a single Web Service instance. Here, an Entity denotes the set of users with same requirements. An Entity may also be

a set of users categorized on the basis of access rights/privileges assigned to them. The approach also isolates and maintains security among various Entities so that an Entity cannot have access to the operations not authorized for them. A new set of tags {<customization>, <EntitySet>, <Entity>, <AlsoApplicableTo>} have been introduced in the WSDL-TC. The function of the <customization> tag is to contain one or more <Entity> tag(s), which specify the Entities/clients for whom temporal Web Services are customized. The name attribute in the <Entity> tag can be any qualified name and the value attribute in the <Entity> tag can be assigned with the user defined Entity name. The user defined Entity name assigned to the value attribute is used to create logical bundle of artifacts for a particular Entity. The artifacts that need to be customized can have one or more <EntitySet> tags defined in their scope. EntitySet refers to the collection of Entities that share same customization. The <EntitySet> tag should have one <Entity> tag defined directly in its scope. This <Entity> defines a primary Entity. The <Entity> tag contains the required set of elements of WSDL that are usually defined in that artifact e.g. Input, Output, Outfault etc. in Operation artifact. If a customized artifact is required by multiple Entities then one can assign the same artifact to them without redefining it. This can be achieved by <AlsoApplicableTo> tag defined under <EntitySet> tag. The <AlsoApplicableTo> tag comprises of one or more <Entity> tag(s), which define the Entities for which the customized artifact is available, these <Entity> tag(s) defines secondary entities.

To formalize BPEL-TC we are using π calculus i.e. process calculus which is a part of Process Algebra and is designed to consider the concurrent system whose configuration may change.

The rest of this paper is organized as follows: section 2 discusses the related work with respect to BPEL formalization techniques that help in better understanding of BPEL. Section 3 discusses the approach of BPEL-Temporal Customization (BPEL-TC) that tackles issues related with change management and customization. The formal representation of BPEL-TC is explained using π calculus in section 4. Section 5 explains the approach with the help of example of Travel Plan Selection Process. Section 6 concludes the paper with merits of the approach.

2. RELATED WORK

Process algebra is mathematical framework that models the concurrent system of interacting purpose [7]. R Milner [8] had introduced a new way of communication for which he developed a theory of π calculus based on process Algebra. Many authors tried to represent Web Services that interact with another Web Service through process algebra i.e. process algebra and their notations help to make a clear understanding of interacting Web Services. Salaun et al. [9] gave mapping between the process algebra and BPEL. Essential facets of Web Services are described using the process algebraic approach by them. Boardeaux [10] suggested that using process algebra it is possible to tackle several issues raised in the context of Web Services and the behaviour of composite Web Services. It helps in the study of specification of process. Process algebras provide a very complete and satisfactory assistance to the whole process of Web Service development. There are many process calculi (CCS, TSTP, LOTOS, Promela etc.) and accompanying tools (CWB-NC, CADP, MWB, SΠN etc) that offers important functionality. Chirichiello et al. [11] used process algebra to introduce different communication models such as synchronous and asynchronous communication. Brogi et al. [12] suggested formalism of choreography proposal and discussed the benefits of formalization. He suggested that π calculus is one of the Algebraic Process that is very adequate to Model check the Web Services mapped π-calculus to BPEL process model. Chunyung et al. [13] proposed a process algebraic framework that addresses the problem of compute information of all backend process that are involved in Web Services. They designed a framework that provide a public view for atomicity and reveal only partial process rather than the whole backend process.

Process algebra and their tools are used in the formal methods that help in the designing Web Services and its verification. Ferrara et al. [14] designed a Framework for the verification purpose of WSs and adding a two way mapping between abstract specification written using these calculi and executable Web Services. He proposed two options either design and verify BPEL4WS by using process algebra tools or design and verify process algebra and automatically obtaining the corresponding BPEL4WS code. Different approaches can also be combined on the use of different user level expressiveness, existence of reasoning tools, user expertise both for verification and for the hierarchical refinement design method. Lucchi et al. [15] formalized a novel orchestration language that improve the quality of BPEL specification and implementation.

3. BPEL-TC

In service-oriented environments, consumers use services for direct interaction or develop applications that use other services. These applications use multiple Web Services and are called composite Web Services. In literature, different ways of combining Web Services are referred to as composite Web Services, Web Service orchestration or choreography. An orchestration is defined using workflow languages such as BPEL [16], BPML and XLANG. Since, existing specifications of the Business Process Execution Language (BPEL) is not compatible with the specifications proposed in WSDL-T thus, a modification of existing WS-BPEL is proposed i.e. BPEL-T [10]. BPEL-T is designed for defining the service flow for the services based on WSDL-Temporal (WSDL-T). BPEL-T introduces a new attribute atTime. The atTime attribute has been added to invoke, receive, reply, onEvent, and onMessage activities present in WS-BPEL. The BPEL-T helps in easy and better management of business process. Further, BPEL-T is extended to BPEL-TC[17] for aggregating the services based on WSDL-TC. Specific customized version of the artifacts within the Web Service, defined in WSDL-TC, is invoked through partner links in BPEL-TC process. As BPEL-TC process is used to orchestrate temporally customized artifacts of the Web Services, BPEL-TC process should be able to detect and bind to a specific customized version of the artifact of the Web Service. Another attribute forEntity is added to invoke, receive, reply, onEvent, and onMessage activities present in BPEL-T in addition to atTime attribute. In case, these are not specified, BPEL-TC process is able to bind some default version of the artifact available in the Web Service i.e. base function defined in WSDL-TC.

3.1 Activities in BPEL-TC

The receive activity allows the business process to wait for a matching message to arrive from the operation mentioned in the operation attribute of the receive activity. The receive activity completes when the message arrives. The portType attribute on the receive activity is optional. The optional message Exchange attribute is used to associate a reply activity with a receive activity. Note atTime and forEntity attribute in receive activity in Listing 1. The value of atTime attribute is compared with the value of the timeStamp attribute of the various versions of the corresponding operation in WSDL-TC. The value assigned to forEntity attribute is compared with the names of the Entities in an EntitySet for whom the operation is customized. The message is received from the version of the operation customized for the Entity assigned to forEntity attribute and whose timeStamp value is highest among all the timestamps which are less than or equal to the value assigned to atTime attribute of receive activity. If atTime and forEntity attributes are missing in the receive activity then the message is received from the base function of the operation with validity status set to LATEST. Both version of the Operation as well the version of the Entity should have valid validity status at the time assigned to atTime attribute.

```
<receive partnerLink="NCName" portType="QName"?
operation="NCName" variable="BPELVariableName"?
createInstance="yes|no"? messageExchange="NCName"?
```

```
atTime="xs:datetime"  forEntity=”EntityName” standard-attributes>
standard-elements
<correlations>?
<correlation set="NCName" initiate="yes|join|no"? />+
</correlations>
<fromParts>?
<fromPart part="NCName" toVariable= "BPELVariableName" />+
</fromParts>
</receive>
```

Listing 1. *receive* activity in BPEL-TC.

The *reply* activity allows the business process to send a message in reply to a message that was received by an inbound message activity (IMA), that is, *receive*, *onMessage*, or *onEvent*. The combination of an IMA and a *reply* forms a request-response operation on a WSDL portType for the process. The portType attribute on the *reply* activity is optional. If the portType attribute is included for readability, the value of the portType attribute must match the portType value implied by the combination of the specified partnerLink and the role implicitly specified by the activity. The optional messageExchange attribute is used to associate a *reply* activity with an IMA. The *atTime* and *forEntity* attributes helps in deciding to which version of the operations customized for a particular Entity in WSDL-TC, a message is sent in reply to a message that was received by an inbound message activity (IMA). If *atTime* and *forEntity* attributes are missing in the reply activity then the message is replied to the base function of the operation and validity status set to LATEST. Listing 2 shows the syntax of *reply* activity of BPEL-TC.

```
<reply partnerLink=”NCName” portType=”QName”?
operation=”NCName” variable=”BPELVariableName”?
faultName=”Qname”? messageExchange=”NCName”?
atTime=”xs:datetime” forEntity=”EntityName” standard-attributes>
standard-elements
<correlations>?
<correlation set=”NCName” initiate=”yes|join|no”? />+
</correlations>
<toParts>?
<toPart part=”NCName” fromVariable= “BPELVariableName” />+
</toParts>
</reply>
```

Listing 2. *reply* activity in BPEL-TC.

The *invoke* activity allows the business process to invoke a one-way or request-response operation on a portType offered by a partner. In the request-response case, the invoke activity completes when the response is received. The portType attribute on the *invoke* activity is optional. If the portType attribute is included for readability, the value of the portType attribute MUST match the portType value implied by the combination of the specified partnerLink and the role implicitly specified by the activity. Listing 3 shows the syntax of *invoke* activity of BPEL-TC.

```
<invoke partnerLink="NCName" portType="QName"?
operation="NCName" inputVariable= "BPELVariableName"? outputVariable="BPELVariableName"?
atTime="xs:datetime"  forEntity="EntityName" standard-attributes >
standard-elements
<correlations>?
<correlation set="NCName" initiate="yes|join|no"?
pattern="request|response|request-response"? />+
</correlations>
<catch faultName="QName"? faultVariable="BPELVariableName"?faultMessageType="QName"?
faultElement="QName"?>*
```

```
</catch>
<catchAll>? activity</catchAll>
<compensationHandler>?activity </compensationHandler>
<toParts>?
<toPart part="NCName" fromVariable="BPELVariableName" />+
</toParts>
<fromParts>?
<fromPart part="NCName" toVariable="BPELVariableName" />+
</fromParts>
</invoke>
```

Listing 3. *invoke* activity in BPEL-TC.

The version to be invoked depends upon optional *atTime* and *forEntity* attribute of invoke element. The value assigned to *atTime* in invoke is compared with *timeStamp* values given to different versions of port type/operation. The value assigned to *forEntity* attribute is compared with the names of the Entities in an EntitySet for whom the operation is customized. The version of the port type/operation customized for the Entity assigned to *forEntity* attribute and whose *timeStamp* value is highest among all the timestamps which are less than or equal to the value assigned to *atTime* attribute of invoke element is invoked. If *atTime* and *forEntity* attributes are missing in the invoke activity then the base function of the operation with validity status assigned as LATEST is invoked.

Similarly, *onEvent* and *onMessage* has *atTime and forEntity* attributes (as shown in Listing 4 and Listing 5 respectively) and these values when compared with the timestamps and Entity Name associated with various versions of operations helps in deciding which version of customized operation is to be selected for the desired actions.

```
<onEvent partnerLink="NCName" portType="QName"? operation="NCName" ( messageType="QName"
| element="QName" )? variable="BPELVariableName"? messageExchange="NCName"? atTime="
xs:datetime" forEntity="EntityName">*
<correlations>?
<correlation set="NCName" initiate="yes|join|no"? />+
</correlations>
<fromParts>?
<fromPart part="NCName" toVariable="BPELVariableName" />+
</fromParts>
<scope ...>...</scope>
</onEvent>
```

Listing 4. *onEvent* activity in BPEL-TC.

```
<onMessage partnerLink="NCName" portType="QName"? operation="NCName"
variable="BPELVariableName"?
messageExchange ="NCName"? atTime="xs:datetime" forEntity="EntityName"> >+
<correlations>?
<correlation set="NCName" initiate="yes|join|no"? />+
</correlations>
<fromParts>?
<fromPart part="NCName" toVariable="BPELVariableName" />+
</fromParts>
activity
</onMessage>
```

Listing 5. *onMessage* activity in BPEL-TC.

4. Π -CALCULUS FOR BPEL-TC

π –calculus [8] is a model of computation for concurrent systems. It is based on process algebra and is most suitably defines processes which are able to exchange names over channels. The syntax of π-calculus helps representing processes, their parallel composition, synchronous communication among processes, replication of processes, and nondeterminism. π-calculus consists of Action and Action Prefix. Action Prefix represents either sending or receiving a message (a name), or making a silent transition (). Table 1 [7] describes Action Syntax and the set of π-calculus.

Table 1. π-calculus

Action Syntax		Process syntax
P::= $\overline{x}(y)$	receive y along x	P:=0(null)
$x(y)$	send y along x	$\|\Sigma_{i\in I}\pi_i P_i\|$(Prefixed Sum)
τ	silent action	$\|P_1\|P_2$(Parallel composition)
		$\|(\nu a)P$ (Restriction)
		$\|!P$ (Replication)

Here we are extending the formal specification of BPEL process provided by Abouzaid [7] to describe activities and process of BPEL-TC. The annotations associated with actions specify information about the *partnerlink, porttype, operation,atTime and forEntity*.

1. $\mathbf{P^{TC} = x(y)_{\{pt:PT,pl:PL,\ atTime=date\text{-}time,forEntity=ESName\}}}$ denotes the receive activity of BPEL-TC process. Through partnerlink PL, porttype PT the message is received from the version of the operation x customized for the Entity ES assigned to *forEntity* attribute and whose *timeStamp* value is highest among all the timestamps which are less than or equal to the date-time value assigned to *atTime* attribute of *receive* activity.
   <receive partnerlink=PL portType=PT operation =x variable =y ***atTime=date-time forEntity=ESName****>*
2. $\mathbf{P^{TC} = x(\bar{z})_{\{pt:PT,pl:PL,atTime=date\text{-}time,forEntity=ESName\}}}$ denotes the reply activity of BPEL-TC process.
   <reply partnerlink=PL portType=PT operation =x variable =z ***atTime=date-time forEntity=ESName*** *>*
3. $\mathbf{P^{TC}=\{\bar{x}(y).x(z)\}_{\{act:invoke,pt:PT,pl:PL,\ atTime=date\text{-}time,forEntity=ESName\}}}$ denotes the invoke activity of BPEL-TC process.
   <invoke partnerlink=PL portType=PT operation =x inputvariable =y outputvariable=z ***atTime=date-time forEntity=ESName>***

Table 2 represents some expressions of BPEL-TC process in π calculus.

Table 2. π-calculus for BPEL-TC activities

π-calculus	**BPEL/BPEL-TC**	**Remarks**
{0}	<invoke partner = “” operation= “” >	Inert Process
$\{\nu x Q^{TC}\}_{\{var1,var2,var3\}}$	<scope standard-attributes> <variables><variable name=x/></variables> <correlationSets>CS</correlationSets> <faultHandlers>FH</ faultHandlers> <compensationHandler>CH	Scope breaks business process into logical units. It allows to define variable that are visible and usable within a scope level.

	< /compensationHandler > <eventHandlers>EH</eventHandlers> Activity </scope>	
$\{!Q^{TC}\}$	<while condition="exp='yes'"> <sequence> <activity> <activity> </sequence> </while>	It defines iterations of the BPEL-TC Process.
$\{Q_1^{TC} \mid Q_2^{TC}\}_{\{act:flow\}}$	<flow> <activity> <activity> </flow>	Specifies Processes that can be performed concurrently.
$\{Q_1^{TC} \mid Q_2^{TC}\}_{\{act:seq\}}$ or $Q_1.y() \mid y(u).Q_2$	<sequence> <activity> <activity> </sequence>	Sequential composition by a parallel operation
$\{\{\overline{y}(v_1)\, Q_1^{TC} + \overline{y}(v_2)\, Q_2^{TC}\}_{\{act:pick\}}$	<pick> <onMessage...variable="v1"> <...activity1...> </onMessage> <onMessage...variable="v2"> <...activity2...> </onMessage> </pick>	Pick BPEL Activity is executed when it receives one message defined in its *onMessage* tag [7].

5. EXAMPLE

In this section, we have chosen Travel Plan Selection Process as an example to advocate the use of π calculus Process Algebra to describe and formalize Travel Plan BPEL-TC process. Travel Plan BPEL process is summarized in Figure 1. On the reception of a request, initialized by the customers of the service, and depending upon the need of the client, a travel plan is returned to a customer. Two choices are available with the client, in Travel Plan A client can select the flight service along with the hotel service, with recommendation of places to visit in that area. In the Travel plan B users are provided with train and hotel booking only. Depending upon the availability of seats and accommodation, a suitable travel plan is returned to the client.

Table 3 shows a piece of the BPEL-TC code corresponding to the Travel Plan Selection along with the corresponding π calculus formalization. The table shows formalization for the request and reply, declaration of message types, sequence and flow constructs, receive and invoke activities for a BPEL-TC process.

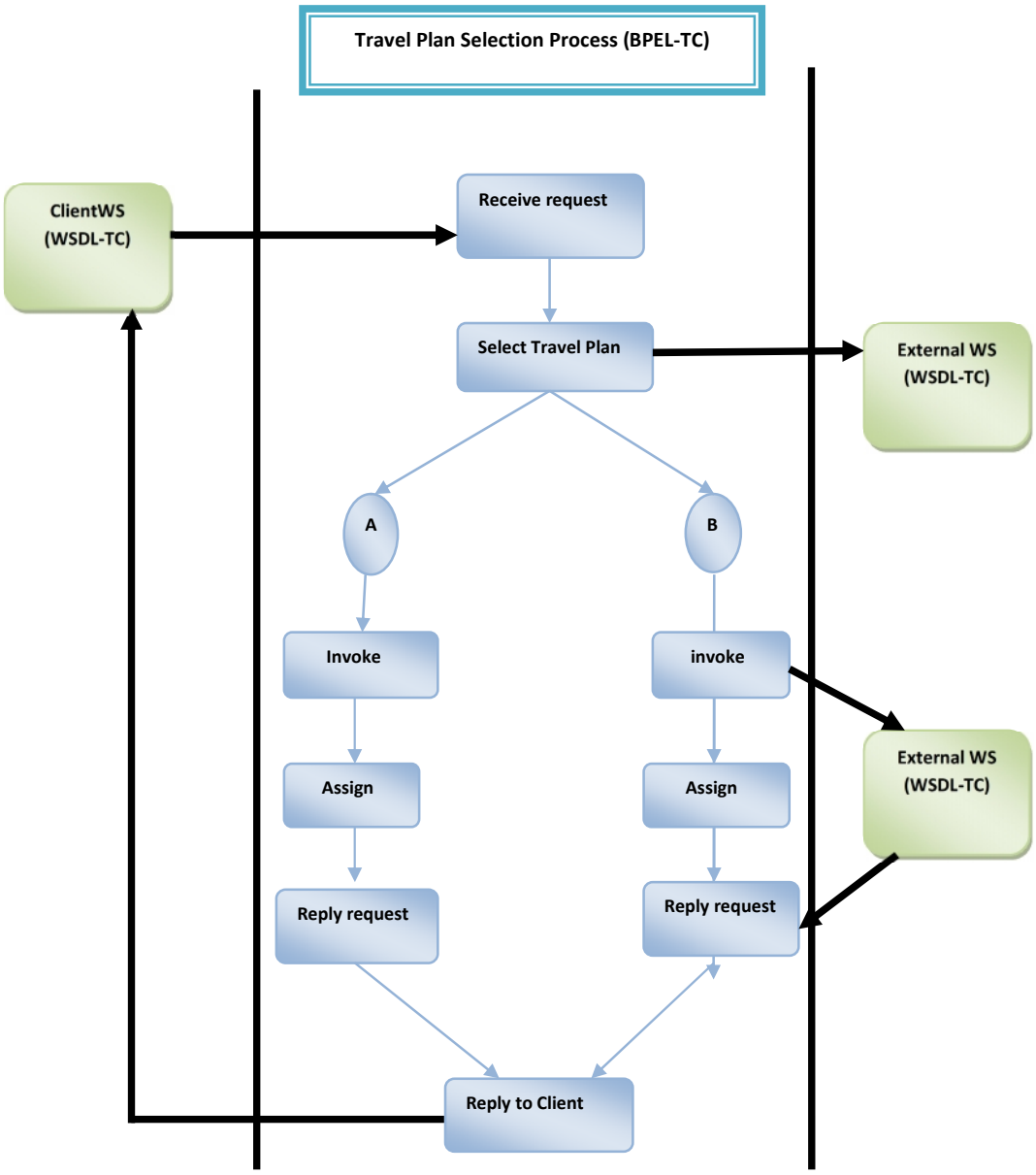

Figure 1. BPEL-TC: Composite Service (WSDL-TC)

Table 3. Mapping of BPEL-TC to π-Calculus

BPEL-TC	**π Calculus**
<Process name=”TravelPlan”>	TravelPlan(request,TravelPlanResponse)

<Variables> *<variable messageType="TravelPlanRequestMessage" name="request"/>* *<variable messageType="CurrentravelPlan" name="InfoMessage"/>* *<variable messageType="TravelPlanResponseMessage" name=" TravelPlanAResponse"/>* *<variable messageType=" TravelPlanResponseMessage" name=" TravelPlanBResponse"/>* *.....* *</variables>*	vrequest P^{TC} vInfoMessage P^{TC} vTravelPlanAResponse P^{TC} vTravelPlanBResponse P^{TC} . . . . . .
<Sequence>	$P^{TC=}$ { P_0^{TC} |P_1^{TC} | P_2^{TC} | P_3^{TC} | P_4^{TC} }$_{\{act:seq\}}$
<receive name="TravelPlan" partnerLink="client" portType="TravelPlanRequest" operation="selectPlan" variable="Request" createInstance="yes" atTime="03/02/2010 10:23:23" forEntity="ES1">	P_0^{TC}= SelectPlan (request)$_{\{pt=}$ TravelPlanRequest pl= client atTime=03/02/2010 10:23:23,forEntity=ES1 }
<invoke partnerLink="currentPlan" portType="getTravelPlan" operation="getTravelInfoRequest" inputVariable="TravelInfoReq" outputVariable="TravelInfoResponse" atTime="03/02/2010 10:23:23" forEntity="ES1>	P_1^{TC}={getTravelInfoRequest (TravelInfoReq). $\overline{\text{getTravelInfoRequest}}$ (TravelInfoResponse) }$_{\{pt=}$ getTravelPlan pl= currentPlan atTime=03/02/2010 10:23:23,forEntity=ES1 }
<flow>	P_2^{TC} = {Q_1^{TC} | Q_2^{TC} }$_{\{act:flow}$
<invoke partnerLink="TravelPlanA" portType="TravelPlan" operation="getInfoA" inputVariable="TravelPlanARequest" outputVariable="TravelPlanAResponse">	Q_2^{TC}={getInfoA (TravelPlanARequest). $\overline{\text{getInfo}}$ (TravelPlanAResponse) }$_{\{pt=}$ TravelPlan pl=TravelPlanA atTime=03/02/2010 10:23:23,forEntity=ES1 }
<invoke partnerLink="TravelPlanB" portType="TravelPlan" operation="getInfoB" inputVariable="TravelPlanARequest" outputVariable="TravelPlanAResponse">	Q_3^{TC}={getInfoB (TravelPlanBRequest). $\overline{\text{getInfo}}$(TravelPlanBResponse) }$_{\{pt=}$ TravelPlan pl=TravelPlanA atTime=03/02/2010 10:23:23,forEntity=ES1 }
</flow>	

<switch>	$P_3^{TC} = Q_4^{TC} + Q_5^{TC}$
<case> *<assign>* *<copy>* *<fromVariable="TravelPlanARequest"/>* *<toVariable=" TravelPlanResponse"/>* *</copy>* *</assign>* *</case>*	Q_4^{TC}
<otherwise> *<assign>* *<copy>* *<fromVariable="TravelPlanBRequest"/>* *<toVariable=" TravelPlanResponse"/>* *</copy>* *</assign>* *</otherwise>*	Q_5^{TC}
</switch>	
<reply partnerLink="client" portType="TravelPlanSelection" operation="selectPlan" variable="TravelPlanResponse" atTime="03/02/2010,10:23:23" forEntity="ES1 />	$P_4^{TC}=\{\overline{\text{selectPlan}}\ (\text{TravelPlanResponse})\}_{\{pt=}$ TravelPlanSelection pl= client atTime=03/02/2010 10:23:23,forEntity=ES1 }
</sequence>	

6. CONCLUSION

BPEL-TC specifies business process behavior based on temporally customized Web Services (WSDL-TC), in which different customized versions of the artifacts are deployed at same URI, instead of maintaining these versions of services at different URIs. The π-calculus is mathematical tool for expressing systems and reasoning about their behavior. It has been used to formalize the processes defined using BPEL. In the presented work, we have extended this mapping to BPEL-TC based Processes. Formal description of Web Services and their processes in algebraic notations such as Process Algebra (PA) can help to describe an abstract specification of the system to be developed giving a preliminary model which can be validated by the reasoning tools, and then be used as a reference for the implementation. Moreover, it is useful for reverse engineering purposes, where translation in the other direction is needed to extract an algebraic description from existing Web Service processes. This allows the use of reasoning techniques to analyze running services.

REFERENCES

[1] Jeffrey C. Schlimmer, (2002) "Web Services Description Requirements", W3C Working Draft, Available from http://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028.

[2] Daniel Austin, Abbie Barbir and Sharad Garg, (2002) "Web Services Architecture Requirements", W3C Working Draft, Available from http://www.w3.org/TR/2002/WD-wsa-reqs-20021011.

[3] Roberto Chinnici, Martin Gudgin, Jean-Jacques Moreau and Sanjiva Weerawarana, (2003) " Web Services Description Language (WSDL) Version 1.2", W3C Recommendation, Available from http://www.w3.org/TR/2003/WD-wsdl12-20030303.

[4] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman and Sanjiva Weerawarana,(2007) "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", W3C Recommendation Available from http://www.w3.org/TR/wsdl20.

[5] Hema Banati, Punam Bedi & Preeti Marwaha, (2012) "WSDL-Temporal: An approach for change management in Web Services", Proceedings of the IEEE International Conference on Uncertainty Reasoning and Knowledge Engineering, pp. 44-49.

[6] Hema Banati, Punam Bedi & Preeti Marwaha, (2012) "WSDL-TC: Collaborative Customization of Web Services, Proceedings of the IEEE International Conference on Intelligent System Design and Applications (ISDA), pp. 692-697.

[7] Faisal Abouzaid, (2006) "A Mapping from Π-Calculus into BPEL", Proceedings of the conference on Leading the Web in Concurrent Engineering: Next Generation Concurrent Engineering, pp. 235-242.

[8] R. Milner, (1999) "Communicating and Mobile Systems: The pi-Calculus". Cambridge University Press, Cambridge, UK.

[9] Salaun Gwen, Bordeaux Lucas & Schaerf Marco, (2004) "Describing and Reasoning on Web Services using Process Algebra", Proceedings of the IEEE International Conference on Web Services, pp. 43-50.

[10] Bordeaux Lucas and Gwen Salaün (2005) "Using process algebra for Web Services: Early results and perspectives", Technologies for E-Services, Springer Berlin Heidelberg, pp. 54-68.

[11] Chirichiello Antonella & Salaun Gwen, (2005) "Encoding Abstract Descriptions into Executable Web Services: Towards a Formal Development", Proceedings of the IEEE International Conference on Web Intelligence, pp. 457 – 463.

[12] Brogi Antonio, Canal Carlos, Mentel Ernesto & Vallecillo Antonio, (2004) **"**Formalizing Web Service Choreographies**",** Electronic Notes in Theoretical Computer Science (ENTCS) Volume 105, pp. 73-94.

[13] Chunyang Ye, S.C. Cheung, W.K. Chan & Chang Xu , (2009) "Atomicity Analysis of Service Composition across Organizations", IEEE Transaction on Software Engineering, Volume 35, No. 1, pp. 2-28.

[14] Ferrara Andrea, (2004) "Web Services: A Process Algebra Approach", Proceedings of the 2nd International Conference on Service oriented computing (ICSOC '04), pp. 242-251.

[15] Roberto Lucchi and Manuel Mazzara, (2007) "A π-calculus based semantics for WS-BPEL" The Journal of Logic and Algebraic Programming, Volume 70, No. 1, pp. 96–118.

[16] Diane Jordan, John Evdemon, Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto et al. (2007) "Web Services Business Process Execution Language Version 2.0", OASIS Standard, Available from http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[17] Hema Banati, Punam Bedi & Preeti Marwaha, (2012) "Extending BPEL for WSDL-Temporal based Web Services", Proceedings of the IEEE 12th International Conference on Hybrid Intelligent Systems (HIS), pp. 484-489.