International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 6 (2014) pp. 270 - 278

BPEL-TC: Orchestration of Temporally Customized Web Services

Preeti Marwaha[1], Hema Banati[2] and Punam Bedi[3]

[1]Department of Computer Science
University of Delhi
Delhi, India.
preeti_andc@yahoo.com

[2]Dyal singh College
University of Delhi
Delhi, India.
banatihema@hotmail.com

[3]Department of Computer Science
University of Delhi
Delhi, India.
punambedi@ieee.org

Abstract: **WS-BPEL is way to orchestrate web services. It defines business processes that interact with external entities through web service operations using WSDL. The existing system defines service flow using Web Services based on WSDL. We have proposed BPEL-TC, an extension to existing WS-BPEL which uses temporally customized Web Services 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. A formal representation of BPEL-TC is also presented using Kleene Algebra with Test (KAT).**

Keywords: Versioning, Collaborative Customization, Temporal, Web Services, Orchestration, WSDL-TC, BPEL-TC

I. Introduction

Web services are useful in building distributed systems that deliver functionality as services and have become one of the preferred platforms for building online customized applications. The present infrastructure for building web services has set the stage for building dynamic, highly scalable and interoperable web applications. Changes in web applications are inevitable because the business requirements are very dynamic. Problem of management of these changes and versioning of web services have been addressed by several authors. In our earlier work we proposed extensions to WSDL [7] i.e. WSDL-Temporal (WSDL-T)[4] and WSDL-Temporal Customization (WSDL-TC)[6]. In WSDL-T, the concepts of linear temporal logic [2][3] as well as frame and slot versioning [17] are used for managing changes across multiple versions of a Web service. WSDL-T maintains different versions of the artifacts under same URI. WSDL-TC defines different versions of the artifacts of the web services customized for different group of users (Entities). It can correlate different versions of messages and process instances, recovery behaviour in case of failures and exceptional conditions. 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. WSDL-TC aims at reducing the cost by maintaining the different collaborative customized versions of the Web service(s) in a single deployment that can be accessed by various groups of clients. The approach also manages access control of these artifacts to their respective groups. These services are composed into business processes. WS-BPEL [13] allows us not only to define abstract process definitions, but to write exact executable specifications of processes which are supported by the majority of companies. There are software on which such processes can be developed (BPEL designers) and executed (BPEL servers). WS-BPEL is used to describe the message exchanges followed by the business process of a specific role in the interaction. The existing mechanisms of Web service orchestration such as WS-BPEL cannot be used for orchestration of web services that are based on WSDL-TC. In our previous work we have extended BPEL to BPEL-Temporal (BPEL-T) [5] to combine web services based on WSDL-T. BPEL-Temporal defines a language for specifying business process behaviour based on temporal Web Services (WSDL- T). BPEL-temporal allows invocation of new or updated versions of the artifacts maintained in WSDL-T file along with the access to old or obsolete versions of the artifacts. Using BPEL-Temporal it is possible to combine any version of the artifacts of temporal Web services. In the proposed work, BPEL-T is extended to BPEL-Temporal Customization (BPEL-TC) for invocation of customized web services based on WSDL-TC. Using BPEL-TC it is possible to define a service flow of WSDL-TC based web

services. BPEL-TC process specifies the exact order in which participating temporally customized web services should be invoked. We have represented BPEL-TC formally using Kleene Algebra with Test (KAT) [9]. Algebraic notations presented by Ginige et al. [10] are extended for composite WSDL-TC based processes.

The rest of this paper is organized as follows: section 2 discusses the related work with respect to the other BPEL extensions that are proposed for meeting various scenarios of the business world. The section also describes other works involving version management of business processes. Section 3 discusses our approach of BPEL-Temporal Customization (BPEL-TC) that tackles issues related with change management and customization. Section 4 describes the BPEL activities with additional attributes for versioning and customization. The formal representation of BPEL-TC is explained using Kleene Algebra with Test (KAT) in section 5. The section 6 discusses the case study of Frontline Demonstration of technologies for the farmers. Various scenarios and comparison between BPEL, BPEL-T and BPEL-TC have been discussed in section 7, Results and Discussions. Section 8 concludes the paper with merits of the approach.

II. Related work

In literature, many extensions to standard BPEL process have been proposed by several authors for different purposes that are required to run business processes smoothly in real world. Agrawal et al. [1] introduces a BPEL extension to address human interactions in BPEL. Kloppmann et al. [15] outlines an extension to WS-BPEL that allows for the definition of sub-processes that can be reused within the same or across multiple WS-BPEL processes. Lee et al. [12] also proposed an extension to BPEL to infuse user interactions into composite services. Nitzsche et al. [20] provide extensions to BEPL for semantic Web services. They use Ontologies as data model. Charfi [8] has discussed limitations of Web Service composition languages such as BPEL, with respect to modularity and adaptability. They have introduced the idea of aspect-oriented workflow languages and presented the design and implementation of AO4BPEL, an aspect-oriented extension of BPEL that supports dynamic weaving. Hackmann et al. [11] have proposed and evaluated a series of BPEL extensions that support the creation of flexible, standards-based pervasive computing applications, even when the devices involved are mobile. The aim of Modafferi et al. [18] is to present a Self-Healing plug-in for a WS-BPEL engine that enhances the ability of a standard engine to provide process-based recovery actions. Kopp et al. [16] provide classification of 62 commercially available extensions and scientifically published extensions. He classifies all the extensions into three categories: Design time only Extensions, Design Time and Runtime Extensions and Runtime Only Extensions. Tripathi and Hinkelmann [23] presented a methodology and system for changing SOA-based business process implementation.

Some authors have addressed the versioning in WS-BPEL. BPEL servers, such as IBM WebSphere Process Server [19] and Oracle BPEL Process Manager [21] provide versioning support to some extent. They provide deployment time versioning and allow deploying different processes under the same name, but with different version numbers. Usually two approaches are used. First, only latest version of the process is accessible which has been deployed most recently. Previous versions are only available to finish existing running process instances. The second approach is to publish different versions of the process under different endpoint URLs, which basically means that each process version is published as a separate endpoint.

Juric et al. [14] have addressed the problem of versioning BPEL process. They provide versioning at two levels: scope level versioning and process level versioning. He introduced new activities as well as extended the existing activities. He proposed extensions to variables to provide version information and introduced version handler for selection of particular version. Their approach is different from ours as we are providing versioning of customized operations in WSDL-TC. BPEL-TC is using these temporally customized versions of the artifacts to define service flow. Ginige et al. [10] proposed the solution for change management in the BPEL process. Their solution is based on algebraic expressions. The purpose of these algebraic expressions is to easily identify the effect of service changes in the orchestrated process. After identifying these effects, changes can be carried out efficiently without disturbing the consistency of the overall BPEL document. In the presented work we have extended the algebraic notations given by him to represent BPEL-TC. Tahamtan et al. [22] used temporal logic to overcome the problem of lack of temporal management capabilities for definition, calculation and monitoring of temporal values such as activity duration and dead-lines as well as checking the temporal conformance of processes. They improve QoS and reduce costs. In their work they introduced an extension of WS-BPEL that makes business processes time aware.

III. BPEL-TC: Extending Business Process for WSDL-TC Based Web Services

In a present scenario if any change occurs in the BPEL process, the changed BPEL process is considered as completely new process. Even for a small change whole process is replicated and deployed again. Business requirements are ever changing so as the change required in the web service and their composition fulfilling those requirements. So, it would become very difficult to manage all the versions simultaneously. We have already proposed an extension to BPEL i.e BPEL-T and further we are extending BPEL-T to BPEL-TC for aggregating the services based on WSDL-TC. Specific customized version of the artifact 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. Two optional attributes *atTime* and *forEntity* are added to *invoke*, *receive*, *reply*, *onEvent*, and *onMessage* activities present in WS-BPEL. In case, these are not specified, BPEL-TC process should be able to bind some default version of the artifact in the web service available i.e. base function defined in WSDL-TC.

BPEL-T was designed for defining the service flow for the services based on WSDL-Temporal (WSDL-T). WSDL- T allows the clients to access of a particular version of the artifacts available within a web service. But, 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. BPEL-T introduces a new attribute *atTime*. The *atTime* attribute is based on linear temporal logic and facilitates the selection of the desired version of artifacts from a bunch of available versions within a single Web service available at a single point of time. BPEL-Temporal enables the processes to call a new or old version of the constituent Web service. 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. This paper extends BPEL-T to BPEL-TC (BPEL-Temporal Customization) to orchestrate/choreograph the services based on WSDL-TC.

Following are the features of BPEL-TC:

- Allows invocation of a particular customized ver- sion of operation describing a business process, such that its output can be an input to the particular customized version of another operation of a Web service describing some other business process.
- Allows access to newer customized version of operations added to a business process along with the older operations of the same or other business proc- ess to the existing and new clients.
- Allows access to the modified operations represent- ing a change in business process while continuing access to the operation before change in business process.
- Allows access to deleted operation representing a business process to the dependent operations of some business processes that are obsolete in new versions.

BPEL-TC allows sequencing of process activities in terms of Web service interactions, where web services are built using WSDL-TC as shown in Figure 1 and have different versions of artifacts customized for different Entities are embedded in it. Proposed *atTime* attribute in above mentioned WS-BPEL activities is optional and is assigned the *date-time* value which is compared to the *timeStamp* value associated with different versions of the artifacts defined in WSDL-TC. If *atTime* attribute is missing in the above mentioned activities then the version of the operation which is latest (i.e. having *validity* status set to LATEST in WSDL-TC) will be selected. *forEntity* attribute is also optional and is assigned a value which is a name of an entity for whom the artifact is customized. The entity name as- signed to *forEntity* attribute is compared with the entity names which are a part of some EntitySet. Entity can be primary entity or secondary entity for which atrifacts of the Web Service are customized in WSDL-TC. If *forEntity* attribute is not specified in above mentioned tags then the version of the operation which is not customized for anyone (i.e. Base Function) is being referred. Base Function is available to all entities not belonging to any of the EntitySet in WSDL-TC. Since, both the above mentioned attributes are optional; it is backward compatible with the existing BPEL programs that aggregates WSDL-TC web services.

IV. 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.

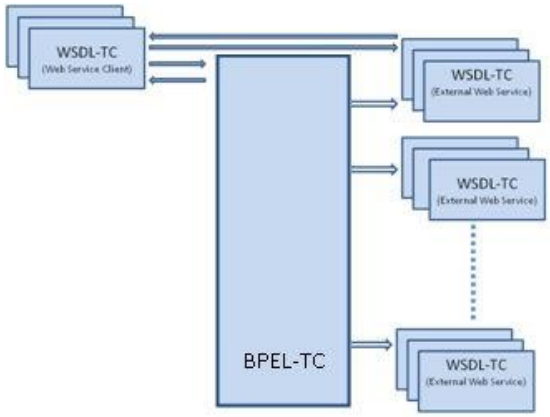

Figure 1. BPEL-TC Interaction with multiple versions of WSDL-TC based web services

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" stan
dard-attributes>
standard-elements
<correlations>?
<correlation set="NCName" initiate="yes|join|no"? />+
</correlations>
<fromParts>?
<fromPart part="NCName" toVariable= "BPELVariable-
Name" />+
</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 message Exchange 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=
“BPELVariable- Name” />+
</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= "BPELVariable-
Name"? 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"? faultVaria-
ble="BPELVariableName"?
faultMessageType="QName"? faultElement="QName"?>*
activity
</catch>
<catchAll>? activity</catchAll>
<compensationHandler>?activity
</compensationHandler>
<toParts>?
<toPart part="NCName" fromVaria-
ble="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" | ele-
ment="QName" )? variable="BPELVariableName"? mes-
sageExchange="NCName"? atTime=" xs:datetime" fo-
rEntity="EntityName">*
<correlations>?
<correlation set="NCName" initiate="yes|join|no"? />+
</correlations>
<fromParts>?
<fromPart part="NCName" toVaria-
ble="BPELVariableName" />+
</fromParts>
<scope ...>...</scope>
</onEvent>
```

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

```
<onMessage partnerLink="NCName" port-
Type="QName"? operation="NCName" varia-
ble="BPELVariableName"?
messageExchange ="NCName"? atTime="xs:datetime"
forEntity="EntityName"> >+
<correlations>?
<correlation set="NCName" initiate="yes|join|no"? />+
</correlations>
<fromParts>?
```

```
<fromPart part="NCName" toVaria-
ble="BPELVariableName" />+
</fromParts>
activity
</onMessage>
```

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

V. KAT for BPEL-TC

In the following section we are giving algebraic notations for BPEL-TC using Kleene Algebra with Test (KAT). The axioms of Kleene Algebra (KA) and Kleene Algebra with Tests (KAT) are presented below. Kleene Algebra (KA) is an algebraic structure (K, +, · , * , 0 , 1) that satisfies the following axioms;

- + and · operators are associative
  a + (b + c) =(a + b) + c and a(bc) = (ab)c for all a, b, c in K
- + is commutative
  a + b = b + a for all a, b In K
- + and · are distributive
  a(b + c) = (ab) + (ac)
  and (b + c)a = (ba) + (ca) for all a, b, c in K
- for + and · there exists an element 0 in K such that for all a in K: a + 0 = 0 + a = a and a0 = 0a= 0
- for + and · there exists an element 1 in K such that for all a in K: a1 = 1a = a
- for * there exists an elements 1 and a in K such 1+aa* = a and 1+ a*a = a. In other words *behaves like the Kleene Star operator in formal language theory.

Kleene Algebra with Test (KAT) is a two-sorted algebraic structure (B, K, +, · , * , 0 , 1, ¬), where B is a subset in K and ¬ is a unary operator, similar to negation, defined only on B such that (K, +, · , * , 0 ,1) is a Kleene Algebra and (B, +, · , ¬, 0 , 1) is a Boolean algebra [9].

The elements in B are usually called tests or guard Elements. Ginige et al.[10] referred φ_1, φ_2, φ_3, ….., φ_n elements (φ elements – PHIs) in B and α_1, α_2, α_3, ….., α_n elements (α elements- ALPHAs) in K.

The basic building blocks of WSDL and BPEL are elements. As WSDL is by nature used for defining services, it does not directly correspond to the state of a process. Hence we consider all WSDL elements to be operating under B (PHIs). Ginige et al.[10] differentiated these elements into three types.

Type 1: Any process-related element that directly influences the state of the process is defined to be in K, an ALPHA. Elements that affect the state of the process *(e.g. <receive>, <reply>, <invoke>, <throw>, <terminate>, <empty>, <compensate> etc*)

Type 2: Other elements that define primitives or provide conditions for ALPHAs to take place belong to B, hence they are also known as guard elements. *(e.g. <variables>, <partnerLinks>,<faultHandlers>, <partners>, <correlationSets>, <eventHandlers>, <assign> etc).*

Type 3: The elements that support the control of the flow are replaced with the KAT expressions in accordance to KAT axioms. Elements that are used for the orchestration of the flow of the process *(e.g. <sequence>, <switch>, <while>,<pick>, <flow>, <scope>, <wait> etc).*

We are extending φ_i to φ^{TC}_j and α_n to α^{TC}_m where

$$\varphi^{TC}_j = \begin{cases} \varphi_{i\#Vz(t,v,ES)} & \text{for WSDL-TC} \\ \varphi_i & \text{for BPEL-TC} \end{cases}$$

and

$$\alpha^{TC}_m = \alpha_n(t',E')$$

where $\varphi_{i\#V(t,v,ES)}$ denotes the version V of i^{th} element/artifact with *timeStamp t*, validity v and customized for EntitySet *ES*. V can be of the format x.y.z. φ_i denotes the PHI elements of WSDL-TC and BPEL-TC that don't have *version number, validity* and *timeStamp* value attached to it (Same as standard WSDL and WS-BPEL). $\alpha_n(t',E')$ denotes the n^{th} activity of BPEL-TC process with *atTime* t′ and *forEntity* E′.

Operating under Guard elements: The Activity that affect the state of the process (ALPHAs (α^{TC}) in K) can be made to operate iff certain guard elements (PHIs (φ^{TC}) in B) are true.

$$(\varphi^{TC}_i\ \varphi^{TC}_j\ \varphi^{TC}_k)\alpha^{TC}_n \quad \text{……} \quad (1)`$$

e.g. if $\varphi^{TC}_i = \varphi_{1\#1.0.1(09/05/2012\ 15:50:39,\ L,\ E1)}$
$\varphi^{TC}_j = \varphi_{2\#1.0.0(07/11/2011\ 11:20:33,\ P,\ E2)}$
$\varphi^{TC}_k = \varphi_3$
$\alpha^{TC}_n = \alpha_{1\ (07/06/2012\ 15:50:39,\ ES1)}$
the Equation 1 can be written as
$(\varphi_{1\#1.0.1(09/05/2012\ 15:50:39,\ L,\ ES1)}\ \varphi_{2\#1.0.0(07/11/2011\ 11:20:33,\ P,\ ES2)}\ \varphi_3)\ \alpha_{1\ (07/06/2012\ 15:50:39,\ ES1)}$

The above expression denotes activity α_1 with attribute *atTime*=07/06/2012 15:50:39 and *forEntity*=E1 is executed if guard elements $\varphi_{1\#1.0.1(07/06/2012\ 15:50:39,L,ES1)}$, $\varphi_{2\#1.0.0(07/11/2012\ 11:20:33,P,ES1)}$ and φ_3 are true. $\varphi_{1\#1.0.1(07/06/2012\ 15:50:39,L,ES1)}$ denotes the guard element (artifact) φ_1 *version* 1.0.1 with *timeStamp* 07/06/2012 15:50:39, *validity* status- LATEST(L) and customized for entity E1. $\varphi_{2\#1.0.0(07/11/2011\ 11:20:33,\ P,\ ES2)}$ denotes the guard element (artifact) φ_2 *version* 1.0.0 with *timeStamp* 07/11/2011 11:20:33, *validity* status-PAST(P) and customized for entityset ES2. Here φ_1 and φ_2 are artifacts of WSDL-TC where as φ3 a PHI element of BPEL-TC.

Sequence: we use · to map the sequential activities that affect the state of the process or ALPHAs. For example, if α^{TC}_i, α^{TC}_j and α^{TC}_k are activities that need to take place in sequence in a given order. Hence we write;

$$(\varphi^{TC}_i\alpha^{TC}_i)(\varphi^{TC}_j\alpha^{TC}_j)(\varphi^{TC}_k\alpha^{TC}_k)$$

The non-commutativity of · allows us to write the above expression to be sequential, as $(\varphi^{TC}_i\alpha^{TC}_i)(\varphi^{TC}_j\alpha^{TC}_j)(\varphi^{TC}_k\alpha^{TC}_k) \neq (\varphi^{TC}_j\alpha^{TC}_j)(\varphi^{TC}_i\alpha^{TC}_i)(\varphi^{TC}_k\alpha^{TC}_k) \neq (\varphi^{TC}_k\alpha^{TC}_k)(\varphi^{TC}_j\alpha^{TC}_j)(\varphi^{TC}_i\alpha^{TC}_i)$, etc.

Choice: if there are some α^{TC}_i, α^{TC}_j and α^{TC}_k elements that

need to be presented as a choice under the guard elements φ^{TC}_i, φ^{TC}_j and φ^{TC}_k, it will be written as follows;

$$\varphi^{TC}_i\alpha^{TC}_i+\varphi^{TC}_j\alpha^{TC}_j+\varphi^{TC}_k\alpha^{TC}_k.$$

This allows α^{TC}_i, α^{TC}_j and α^{TC}_k to be carried out in any order (provided that the guard conditions are satisfied), as $\varphi^{TC}_i\alpha^{TC}_i+\varphi^{TC}_j\alpha^{TC}_j+\varphi^{TC}_k\alpha^{TC}_k=\varphi^{TC}_j\alpha^{TC}_j+\varphi^{TC}_i\alpha^{TC}_i+\varphi^{TC}_k\alpha^{TC}_k=\varphi^{TC}_k\alpha_k+\varphi^{TC}_j\alpha_j+\varphi^{TC}_i\alpha_i$, etc.

Parallelism: In BPEL, the <flow> construct represent the parallel activities that can take place. Let us consider α^{TC}_i, α^{TC}_j and α^{TC}_k elements that need to occur in parallel under guard elements φ^{TC}_i φ^{TC}_j and φ^{TC}_k until a merger condition φ^{TC}_m is satisfied. This is written as follows:

$$(\varphi^{TC}_m\,(\varphi^{TC}_i\alpha^{TC}_i+\varphi^{TC}_j\alpha^{TC}_j+\varphi^{TC}_k\alpha^{TC}_k))^*$$

This is interpreted as: the occurrence of activities φ^{TC}_i, φ^{TC}_j and φ^{TC}_k can iterate under the * operator, in any order until φ^{TC}_m is satisfied.

Wait: If action α^{TC}_i needs to be performed after a certain deadline defined in φ^{TC}_i. This Waiting is modeled as:

$$\varphi^{TC}_i\,(\alpha^{TC}_i)$$

Switch: If α^{TC}_i is allowed to take place under the guard condition φ^{TC}_i otherwise α^{TC}_j is allowed to happen. This is written as;

$$\varphi^{TC}_i\,(\alpha^{TC}_i)+\neg\varphi^{TC}_i\,(\alpha^{TC}_j)$$

where $\neg\varphi^{TC}_i$ presents the situation where φ^{TC}_i condition is not true.

Empty: In BPEL <empty> actions can be modeled using the special element 1ϵ K. Let us consider that there is a choice to either perform the activity α^{TC}_i or have an empty element. This is modeled as:

$$(\alpha^{TC\,v}_i + 1)$$ using KAT based notations.

VI. Case Study

We have implemented WSDL-TC web services for FLD (Frontline Demonstration) for different crops like Maize, Rice, Wheat etc. Frontline Demonstration is a participatory research, emphasizing scientist-farmer interaction, refining and validating research findings, developing leadership amongst farmers for multiplier effect to horizontally disseminate technology. The FLDs provide an effective learning situation as the farmers observe the technologies, practice it and interact with the scientists and extension functionaries [24]. It is very necessary to record the observation and get the feedback from the farmers and the extension workers for all the FLD experiments. It also helps in analyzing the FLD experiment as well as FLD program as a whole. Since, the Internet connectivity is not readily available at farmers' field, so a Web service based approach is used for collecting the data from the Maize farmers' field.

Also, FLD performas' tend to change a bit over time and for different crops, so a new approach based on WSDL-Temporal Customization Web Service has been used for developing the system. The data and feedback collection system has been designed using WSDL-TC. Listing 6 shows the snippet of the FLD web service maintaining different versions of the customized operations. BPEL-TC is used to define the service flow for these temporally customized web services. Initially, we designed web service and their clients for FLD for Maize which we called as Base function. This function is non- customized function which is available to all the clients of this web service. Then, we extended our work for FLD for Rice crop. There we incorporated some changes according to the Rice crop. Thus, we modified some operations of the existing service and its clients resulting in a new customized version of an artifact within the same service. Over the time some functionality of FLD for Maize crop changes, which resulted in a new version of the operation in the service. Now, two versions of operation FLD for maize exist and the base function is also customized for FLD for Rice crop. FLD for wheat is nearly same as FLD for Rice thus same operation can be accessed for wheat FLD also. Figure 2 shows BPEL-TC process accessing multiple versions of an operation from a single instance of WSDL-TC based FLD web service. The figure also shows the enhanced BPML modelling notation to represent multiple versions of an operation. Listing 7 shows the BPEL-TC code snippet for aggregating WSDL-TC based web Services.

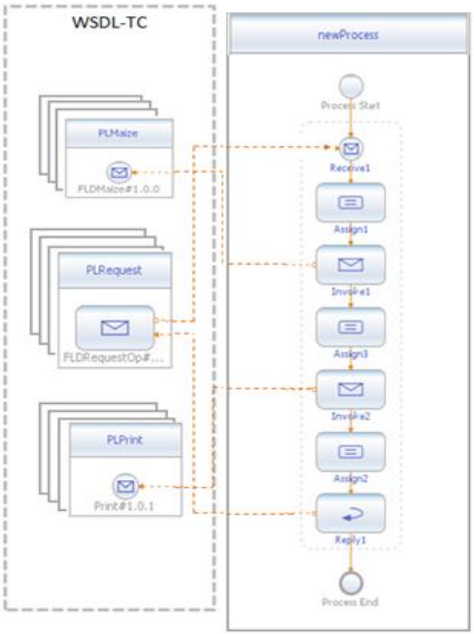

Figure 2. BPEL-TC process for FLD System

```
operation name="FLDMaize#1.0.0" validity="PAST"
timeStamp="11-01-2010 14:20:08">

<! --base functionality initially designed for MaizeFLD -->

<wsdlct:EntitySet name="FLD1" validity="LATEST"
timeStamp="11-01-2010 14:20:08">

<wsdlct:Entity name="RiceFLD" value="RiceFLD"
validity = "LATEST" timeStamp="11-01-2010 14:20:08">

 <!-- Customized functionality for Rice FLD goes here -->

</wsdlct:Entity>

<wsdlct:AlsoApplicableTo  name="ATFLD"
validity="LATEST" timeStamp="11-06-2010 17:40:10">

<!-—same Rice FLD Customization for Wheat FLD -->

<wsdlct:Entity name="WheatFLD" value="WheatFLD"
validity="LATEST" timeStamp="11-06-2010 17:40:10"/>

</wsdlct:AlsoApplicableTo>

</wsdlct:EntitySet>

</operation>
```

```
<operation name="FLDMaize#1.0.1" validity="PAST"
timeStamp="11-01-2010 14:20:08">

<!--new version of operation FLD#1.0.0-- >

...



</operation>
```

Listing 6. WSDL-TC snippet for FLD Service

```
<?xml version="1.0" encoding="UTF-8"?>
<process    name="FLDBpelProcess" ……..>
.
.
<partnerLinks>

<partnerLink name=" PLRequest" ... />
<partnerLink name=" PLValidate" ... />
<partnerLink name=" PLMaize" ... />
<partnerLink name=" PLPrint" ... />
<partnerLink name=" PLRequest" ... />
</partnerLinks>
.
.
<sequence >
.
 <receive name="Receive1" atTime="11-04-2010
14:20:08" forEntity = "RequestFLD"
createInstance="yes" partnerLink=" PLRequest"
operation="FLDRequestOp" ... />
 <invoke name="Invoke1" atTime="11-04-2010
14:20:08" forEntity = "RiceFLD"
partnerLink="PLMaize" operation="FLDMaize" ... />

<invoke name="Invoke2" partnerLink="PLPrint"
operation="Print" .../>

<reply name="Reply1" atTime="11-04-2010 14:20:08"
forEntity = "RequestFLD" partnerLink="PLRequest"
operation=" FLDRequestOp" />
 </sequence>
    .....
</process>
```

Listing 7. BPEL-TC snippet for FLD Services based on WSDL-TC.

VII. Results and Discussions

Let us say op1#1.0.0 and op1#1.0.1 denotes two versions of the operation op1 available with the values of validity and timestamp as shown in Table 1. These versions of the operations are Non customized (NC) versions which are not customized for any Entity and is the base function available to everyone except those for whom customizations are defined. $(op1\#1.0.1)_{s1}$ denotes the version of operation op1 customized for the entity set ES1. Three such versions exist with associated validity status and the timestamp values. $(op1\#1.0.1)_{s2}$ denotes the customization of the operation op1 available to Entity Set ES2.

Table 1. Sample Scenarios showing values of additional attributes of BPEL-TC for Versions and Customizations

Id	Operation	Customized-For	Validity	TimeStamp
v1	op1#1.0.0	NC	PAST	07/11/2011 16:53:34
v2	op1#1.0.1	NC	LATEST	07/06/2012 11:50:34
v2.1	$(op1\#1.0.1)_{s1}$	ES1->E1,E2	PAST	07/06/2012 15:50:39
v2.2	$(op1\#1.0.1)_{s1}$	ES1->E1,E2	PAST	07/07/2012 15:50:39
v2.3	$(op1\#1.0.1)_{s1}$	ES1->E1,E2	LATEST	07/09/2012 15:50:24
v2.4	$(op1\#1.0.1)_{s2}$	ES2->E3,E4	LATEST	07/09/2012 05:50:24

BPEL-TC invokes an operation op1 *forEntity*="E1" and *atTime*= "07/07/2012 15:50:39" then those versions of operation customized for the entity E1 and have a respective timestamp less than or equal to the timestamp attached to *atTime* attribute are selected (versions of the operation with ids v2.1 and v2.2) and finally the version whose time stamp is greatest among all selected versions are invoked (version with id v2.2). If BPEL-TC invokes an operation op1 *forEntity*="E1" and *atTime*= "07/09/2012 15:50:24" then those versions of operation customized for the entity E1 and have a respective timestamp less than or equal to the timestamp attached to *atTime* attribute(i.e. 07/09/2012 15:50:24) are selected (versions of the operation with ids v2.1,v2.2 and v2.3) and finally the version whose time stamp is greatest among all selected versions are invoked (version with id v2.3). If BPEL-TC invokes a partnerlink without *forEntity* and *atTime* attributes then the non- customized version of the operation with validity status LATEST is invoked (i.e. v2).

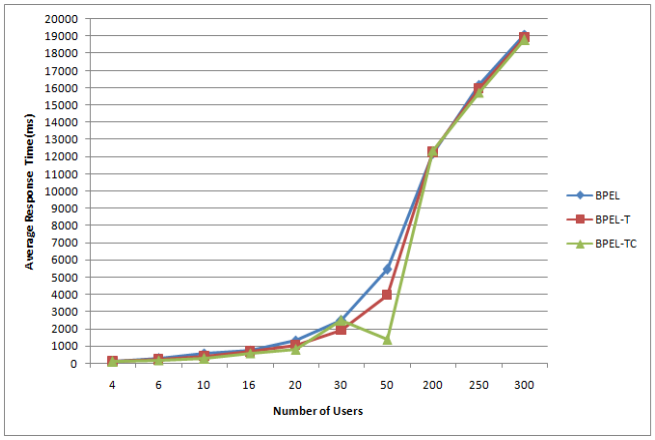

Figure 3. Average response time of BPEL-TC, BPEL-T as compared with WS-BPEL.

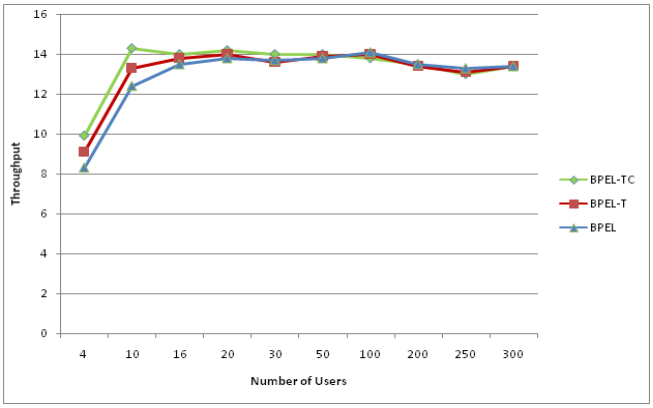

Figure 4. Throughput of BPEL-TC and BPEL-T as compared with WS- BPEL.

Using Apache Jmeter, we have run and compared the throughput and average response time (for different number of concurrent requests) of the standard BPEL process with BPEL-T and BPEL-TC process that uses temporal and temporally customized web services and reached to a conclusion that response time and the throughput of both BPEL-T and BPEL-TC are comparable to BPEL as shown in Figure 3 and Figure 4. The graph in Figure 3 shows that as the number of user increases the average response time increases slowly but after a point the average response time increases. This is due to the fact that till the system resources are available, the user requests are served up to the satisfaction levels. After saturation level is reached and there are no more resources left as a consequence more number of users are put to wait state. So, the average response time rises sharply. Figure 4 shows that although the average response time increases sharply after a point, the throughput increases initially and remains almost constant afterwards.

Thus, the graphs in Figure 3 and Figure 4 show that BPEL-T or BPEL-TC do not degrade the average response time or throughput and the overhead required to process the BPEL-T/BPEL-TC files accessing multiple versions from WSDL-T/TC is minimal. It means that when WSDL-TC based web services are deployed and their orchestration is done to fulfil the requirements of a business processes, the service producers may deploy multiple versions for their multiple clients from a single instance. This in turn has a clear reduction in terms of infrastructure requirements as number of instances per service is reduced to one. Man power requirements for managing and taking back-up of multiple versions are also reduced because there is only single instance per service is required to be deployed. It also allows ease in patch management as the security patches or bug fixing in the non-customized and non-versioned segment of the web service is required to be done at a single place rather than in all the versions.

VIII. Conclusion

BPEL-TC specifies business process behaviour 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 artifacts within services at different URIs.

BPEL-TC allows invocation of new or updated customized versions of the artifacts maintained in WSDL-TC file along with the access to old or obsolete versions of the artifacts within same WSDL-TC file. Using BPEL-TC it is possible to combine any customized version of the artifact(s) of temporally customized Web service(s). Clients can continue to use any version without being forced to upgrade to the latest version.

REFERENCES

[1] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, K. Plösser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, M. Zeller. "WS-BPEL Extension for People (BPEL4People)". Version 1.0, *White Paper*, 2007.

[2] J.F. Allen. "Maintaining Knowledge about Temporal Intervals". *Communications of the ACM*, XXVI, pp. 832-843, 1983.

[3] J.F. Allen. "Actions and Events in Interval Temporal Logic". *Journal of Logic and Computation*, IV, pp. 531-579, 1994.

[4] H. Banati, P. Bedi, P. Marwaha. "WSDL-Temporal: An Approach for Change Management in Web Services". In *Proceedings of IEEE International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE)*, pp. 44-49, 2012.

[5] H. Banati, P. Bedi, P. Marwaha. "Extending BPEL for WSDL-Temporal based Web Services". In *Proceedings of IEEE 12th International Conference on Hybrid Intelligent Systems (HIS)*, pp. 484-489, 2012.

[6] H. Banati, P. Bedi, P. Marwaha. "WSDL-TC: Collaborative Customization of Web Services". In *Proceedings of IEEE International Conference on Intelligent System Design and Applications (ISDA)*, pp. 692-697, 2012.

[7] D. Booth, C.K. Liu. "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer". *W3C Recommendation*, http://www.w3.org/TR /2007/REC-wsdl20- primer-20070626, 2007.

[8] A. Charfi, M. Mezini. "Aspect-Oriented Web Service Composition with AO4BPEL". In *Proceedings of European Conference on Web Services (ECOWS), LNCS 3250*, pp. 168-182, 2004.

[9] D. Kozen, "Kleene Algebra with Tests". *ACM Transactions on Programming Languages and Systems (TOPLAS)*, XIX(3), pp. 427-443, 1997.

[10] J. Ginige, U. Sirinivasan, A. Ginige. "A mechanism for efficient management of changes in BPEL based business processes: an algebraic methodology". ICEBE, In *Proceedings of IEEE International Conference on e-Business Engineering*, pp. 171–178, 2006.

[11] G. Hackmann, C. Gill, G.C. Roman. "Extending BPEL for Interoperable Pervasive Computing". In *Proceedings of IEEE International Conference on Pervasive Computing*, pp. 204-213, 2007.

[12] J. Lee, Y.Y. Lin , S.P. Ma, S.J. Lee. "BPEL Extensions to User-Interactive Service Delivery". *Journal of Information Science and Engineering*, XXV, pp. 1427-1445, 2009.

[13] D. Jordan, J. Evdemon. "Web Services Business Process Execution Language Version 2.0". *OASIS*

Standard, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf, 2007.
[14] M.B. Juric, A. Sasa, I. Rozman. "WS-BPEL Extensions for Versioning". *Information and Software Technology*, LI(8), pp. 1261-1274, 2009.
[15] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. Riegen, P. Schmidt, I. Trickovic. "WS-BPEL Extension for Sub-processes - BPEL-SPE". *White Paper*, 2005.
[16] O. Kopp, K. Görlach, D. Karastoyanova, F. Leymann, M. Reiter, D. Schumm, M. Sonntag, S. Strauch, T. Unger, M. Wieland, R. Khalaf. "A Classification of BPEL Extensions". *Journal of Systems Integration*, IV(2), pp. 3-28, 2011.
[17] S. Marwaha, P. Bedi. "Temporal Extension to OWL Ontologies". *International Journal of Information Technology*, IV(1), pp. 53-60, 2007.
[18] S. Modafferi, E. Mussi, B. Pernici. "SH-BPEL: A self-healing plug-in for Ws-BPEL engines". In *Proceedings of 1st Workshop on Middleware for Service Oriented Computing*, MW4SOC, ACM, pp. 48-53, 2006.
[19] J. Neth, M. Smolny, C. Zentner. "Versioning Business Processes and Human Tasks in WebSphere Process Server". *IBM developerWorks*, http://www.ibm.com/developerworks/websphere/library/techarticles/0808_smolny/0808_smolny.html, 2008.
[20] J. Nitzsche, T. Van Lessen, D. Karastoyanova, F. Leymann. "BPEL for Semantic Web Services (BPEL4SWS)". In *Proceedings of 3rd International Workshop on Agents and Web Services in Distributed Environments (AWeSome'07) - On the Move to Meaningful Internet Systems: OTM 2007 Workshops. Lecture Notes in Computer Science*; 4805/2007, Springer, 2007.
[21] Oracle, "Oracle BPEL Process Manager Developer's Guide" 10 g (10.1.3.1.0), Part Number B28981-03,http://download.oracle.com/docs/cd/B31017_01/integrate.1013/b28981/toc.htm, 2007.
[22] A. Tahamtan, C. Osterle, M. Tjoa, A. Hameurlain, "Temporal Management of WS-BPEL Processes". *Business Information Processing*, CII, pp. 256-269, 2012.
[23] F. Tripathi, K. Hinkelmann. "Change Management in Semantic Business Processes Modeling". ISADS, In *Proceedings of Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)*, pp. 155–162, 2007.
[24] V. K. Yadav, S. Dass, R. Choudhary, K. P. Singh. "Frontline Demonstration in Adoption of Technology and Socioeconomic Upliftment". In *Proceedings of National Conference on Doubling Maize Production*, pp.106-111, 2007.

Author Biographies

Preeti Marwaha is a Ph.D. scholar in the Department of Computer Science, University of Delhi. She is an Assistant Professor in the Department of Computer Science, A.N.D. College, University of Delhi. Her research interest includes Web Services and Composite Web Services, Semantic Web Services etc.

Dr Hema Banati completed her Ph,D.(2006) after her Masters in Computer Applications(M.C.A) both from Department of Computer Science, University of Delhi, India. At present she is an Associate Professor in the Department of Computer Science, Dyal Singh College, University of Delhi. She has over 18 years of teaching experience to both undergraduate and postgraduate classes. Over the past decade she has been pursuing research in the areas of Web engineering, software engineering, Human Computer Interaction, Multi-Agent systems, E-commerce and E-learning. She has many national and international publications to her credit

Punam Bedi received her Ph.D. in Computer Science from the Department of Computer Science, University of Delhi, India in 1999 and her M.Tech. in Computer Science from IIT Delhi, India in 1986. She is an Associate Professor in the Department of Computer Science, University of Delhi. She has about 24 years of teaching and research experience and has published about 100 papers in National/International Journals/Conferences. Dr. Bedi is a member of AAAI, ACM, senior member of IEEE, and life member of Computer Society of India. Her research interests include Web Intelligence, Soft Computing, Semantic Web, Multi-agent Systems, Intelligent Information Systems, Intelligent Software Engineering, Intelligent User Interfaces, Requirement Engineering, Human Computer Interaction (HCI), Trust, Information Retrieval and Personalization.