International Journal of u- and e- Service, Science and Technology
Vol.6, No.5 (2013), pp.115-132
http://dx.doi.org/10.14257/ijunesst.2013.6.5.11

Corroboration Strategy for Web Services Choreography using Revise Buchi Automata

N. Danapaquiame[1] and E. Ilavarasan[2]

[1]*Research Scholar, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, India*
[2]*Associate Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, India*
[1]*n.danapaquiame@gmail.com,* [2]*eilavarasan@pec.edu*

Abstract

Web service choreography is a methodology for building value-added applications by aggregating several existing web services together according to business requirements. Web service choreography has been widely explored in the past and various approaches have been proposed. Only a few works have been carried out for verification of composed web services for deterministic system and no work has been carried for non-deterministic system. This paper introduces an approach for verifying the WSC using Revise Buchi Automata (RBA) for non-deterministic system. The correctness properties of the non-deterministic system have been evaluated based on sparseness and reachability problem. Primarily the web services are composed by using Web Service Choreography Description Language (WS-CDL) and then the composed service is translated into RBA. A new algorithm called Revise Buchi Sparseness Reachability (RBSR) has been developed to avoid the Sparseness problem and Reachability problem. A tool (WSCVT) has been developed using jave for verifying the RBSR algorithm. The proposed approach is verified using this tool and the results revealed better performance in terms of avoiding sparseness and reachability problems at 95% compared to the existing approaches.

Keywords*: RBA, Web service choreography, WSCDL, BPEL, RBSR*

1. Introduction

The term web services describes a standardized way of integrating web-based applications using the XML, SOAP, WSDL and UDDI open standards over an internet protocol backbone. Web services are therefore applications that provide services that can be obtained through the internet. Service Oriented Architecture (SOA) is an architectural strategy that enables web applications to be built using services and one way of implementing this architecture is by using web services. Basically, a service provider develops a service and publishes the service description in a service registry. In general, a service client to find the required service from the service registry sends a request to the service provider via SOAP and receives a response in return. The type of service could be such as retrieving available seat information in a transport service or buying stationary items from a web store.

Many researchers have focused on the discovery, selection and composition of web services whereas only minimal works have been explored in verification of composed web services.

In recent years, many WSC languages have been proposed and they are broadly classified as Web Service Orchestration (WSO) and Web service Choreography. WSO combines

ISSN: 2005-4246 IJUNESST

available services by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single sub-activities. An example is Business Process Execution Language (BPEL). WSC does not exploit a central coordinator but rather defines complex tasks via the definition of the conversation that should be undertaken by each contestant. Following this approach, the overall activity is attained as the composition of peer-to-peer interactions among the collaborating services. An example is WS-CDL.

The main contribution of this paper is summarized as follows:

- A new approach RBA is suitable for both deterministic and non-deterministic system. A deterministic system transit to single state on receiving the single input whereas a non-deterministic system transit to many states on receiving the same input.
- A new algorithm Revise Buchi Sparseness Reachability (RBSR) has been developed to avoid the Sparseness, Reachability problems.
- The proposed approach is compared with the existing approaches namely emptiness and reachability implemented using Timed Automata [1] and dead transition which is implemented using Interface Automata [2] and *etc.*,
- A tool to verify WSC namely WSCVT (Web Service Choreography Verification Tool) has been developed.
- Experimental results shows that the proposed approach outperforms the existing approaches in terms of avoiding sparseness and reachability problems,

The rest of this paper is organized as follows. Section 2, describes an application overview and our approach. Section 3, introduces our architecture, WSCVT Tool and Composition algorithm. Section 4 describes verification algorithm RBSR. Section 5, depicts the implementation with real time applications i.e. Case Study. Section 6, depicts the translation of WS-CDL to RBA. Section 7, depicts the evaluation with existing tool and proposed tool. Section 8, confer the recital of the system. Section 9 depicts the comparing the parameters satisfied by the existing approaches as well as proposed approach. Section 10, designates related work, recapitulate our main outcomes and give a view on future research.

2. System Overview

2.1. Application Overview

2.1.1. JAX-RPC

JAX-RPC stands for Java API for XML-based RPC. It's an API for edifice web services and clients that used Remote Procedure Call (RPC) and XML. It is frequently used in a distributed client/server model, an RPC mechanism facilitates clients to execute procedures on other systems. In JAX-RPC, a remote procedure call is epitomized by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses.

These calls and responses are transmitted as SOAP messages over HTTP. Although JAX-RPC relies on complex protocols, the API hides this convolution from the application developer.

On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also cyphers one or more classes that implement those methods. Client programs are also easy to code. A client generates a proxy, a indigenous object representing the service, and then simply beseechs

techniques on the proxy. With JAX-RPC, clients and web services have an enormous benefit, the platform impartiality of the Java programming language. In addition, JAX-RPC is not restraining: a JAX-RPC client can access a web service that is not running on the java platform and vice versa. This litheness is conceivable since JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP and WSDL. WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

2.1.2. WS-CDL

WSC describes the global model of service interaction among the set of contestants. It is a language for describing how peer-to-peer participants collaborate.

The language uses XML, and some facets are stimulated by the pi-calculus. The service interactions are specified by using WS-CDL. The WS-CDL model consists of the following entities: Information Types, Tokens, Token Locators, Role Types, Relationship Types, Channel Types, Choreographies, Participants.

Information

WS-CDL is not only specifying the choreography of messages, nevertheless also, if indispensable the category of messages being exchanged. Choreography is cognizant of some of the content of message exchanges via tokens and token Locators.

Information Type: aliases to WSDL types, XSD or other typing mechanism

Token: this is essentially a variable of the choreography bound to document content

Token Locator: this is a condition expression,

2.1.3. Participants

This is the key of the WS-CDL specification.

Role: specifies the operations implemented by a given participant that will perform this role in the choreography.

Relationship: ultimately every choreography can be divided into a series of binaries associations. A correlation expresses just that, the assurances of apiece contestant to another.

Participant: a participant implements one or more role in the choreography.

Channels

Channels are used for communication purposes. There are two types of channels: a business channel where the business documents are exchanged and a technical channel to implement the business transaction protocol

A channel type description indicates the role the receiver (of the request or response message) is playing and which behavior the receiver performs on this channel. Channels can also be used to pass channel type variables in addition to passing application and state variables. Accordingly, a channel type description may show the names of the other channel types it can exchange.

2.1.4. Choreography

The following tabular column shows the syntax of the translation from WS-CDL to RBA term.

WS-CDL Term	RBA Term
Assign role Type=”user”> <source expression=”true”/> <target variable=”res” </assign>	Declaration>res=true </declaration>
<interaction……. <interaction name=”return places”> <exchange name=”get places Exchange”> <send variable=”ACKvariable”/> </exchange>	<transition…… <transition> <source ref=”init_interaction0_getplaces_interaction”/> <target ref=Init-interaction1_returnplaces_interaction”/> </transition>
<work unit…	<transition……. <Source ref=”Int choice0”/> <target ref= ”Int_interaction5_reserveticket_interaction”/> <label kind =”guard”>(res==true)</label> </transition>

A choreography description is a container for a top-level activity and optional exception and finalizer work units. The exception work unit may be activated if exceptions occur. The finalizer work unit may be triggered when the choreography has completed successfully, but a fiasco in another choreography means that this completed choreography must be rolled back.

2.1.5. RBA

2.1.5.1. Definition

A Büchi automaton is a type of ω-automaton, which extends a finite automaton to infinite inputs. It accepts an infinite input sequence if there exists a run of the automaton that visits (at least) one of the final states infinitely often.

Büchi Automata recognize the omega-regular languages, the infinite word version of regular languages; it consists of vertices and edges. The transitions are represented as vertices and states are represented as edges.

2.1.5.2. Formal Definition

Formally, a deterministic Büchi automaton is a tuple A = (Qp_{rba}, Σp_{rba}, δp_{rba}, $q0p_{rba}$, Fp_{rba}) that consists of the following components:

Qp_{rba} is a finite set. The elements of Qp_{rba} are called the states of A.
Σp_{rba} is a finite set called the alphabet of A.
δp_{rba}'': $Qp_{rba} \times \Sigma p_{rba} \rightarrow Qp_{rba}$ is a function, called the transition function of A.
$q_0 p_{rba}$ is an element of $Qp_{rba,}$ called the initial state.
$Fp_{rba} \subseteq Qp_{rba}$ is the acceptance condition. A accepts exactly those runs in which at least one of the infinitely often occurring states is in Fp_{rba}.

2.1.6. Revise Buchi Acceptance

We say M accept a ω-word $\alpha \in \sum^{\omega}$ if and only if there exist a run r of M on α satisfying

Inf(p)∩F≠0

i.e., at least one accept state in F has to be visited infinitely often during the run p, in that case the language has been defined for of M to be the set $L(M) := \{\alpha \in \sum^{\omega} \mid M \text{ accept } \alpha\}$

A non-deterministic ω-automaton M over an finite alphabet $\sum$ is a tuple (Qp, q_{0p},δp, Fp,$\sum_p$), where Q is a finite set of states, q_0 is an initial state,

$$\delta : Q \times \sum \rightarrow 2^{q}$$

is a transition relation and Fp is an final state (acceptance condition).

3. System Architecture

The proposed architecture consists of two parts namely composition and verification. In the composition part the user request is collected. Based on the user request the related web services will be invoked. The invoked services are composed by using WS-CDL.

WS-CDL is given as the input to the verification part and WS-CDL is converted into RBA. The RBSR algorithm is verified using the verification tool WSCVT.

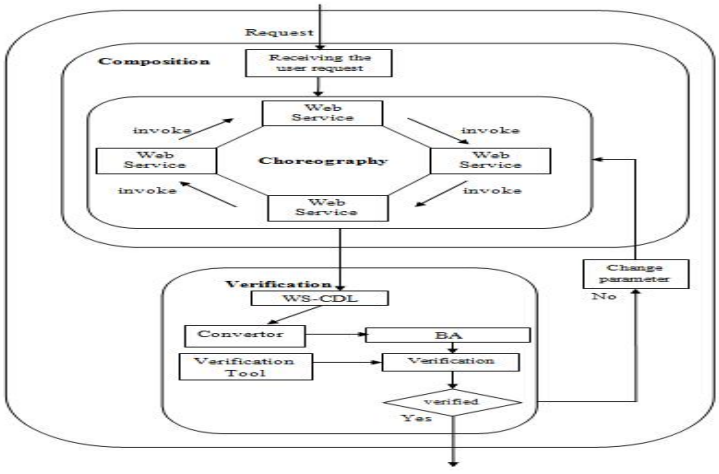

Figure 1. Our Framework for Use of Web Service Choreography

The benefits of this architecture are:

- The system is easily scalable.
- No central repository is required.
- Ensure interoperability.
- Increase robustness.

3.1. WSCVT

The aim of this tool is to deal with the implementation, verification and validation of WSC. In the implementation phase web services are created by using JAX-RPC and then the implemented services are composed by using WS-CDL. The WS-CDL is used to obtain the interaction between the web services. For the verification and validation phase RBA representation of the system is used, which is automatically obtained from the WS-CDL description. Finally the non-deterministic automata diagram is generated from the RBA representation. The WSCVT tool is depicted in the following Figure.

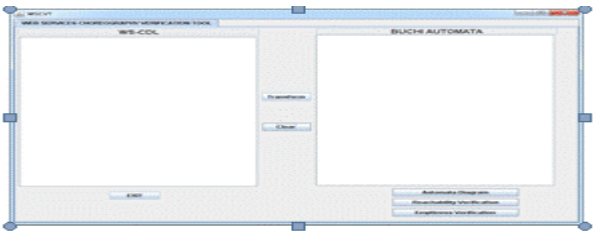

Figure 2. WSCVT Tool

3.1.1. Composition Algorithm

WS-CDL is used for composition. The following steps are for creating the WS-CDL from the created web services is described. Initially the web services are created using JAX-RPC and based on the user request the related services are identified. The composer program will set the entities required for creating the WS-CDL. After completing the composition, the WS-CDL will be given to the verification part.

Input: Request for Web service

Output: Composed service

a) Web services registered in registries
b) User enters request for Web service
c) Request goes to composition part
d) Invoke the related services from the registry
e) Composer composes invoked services
   i. Identification of the tokens from the invoked services
   ii. Identification of the roles of each services
   iii. Participants are identified and declared
   iv. Relationship between the user requested service and invoked services are identified
   v. Description of the channels
   vi. Choreography
      a. Declaration of the variables
      b. Identify the variable channel
      c. Assign value to the variable
      d. Interaction between the services

f) Composer sends result to the verification part

Figure 3. Web Service Composition Algorithm

4. RBSR Algorithm

The proposed algorithm checks the Sparseness and Reachabiltiy. An automaton B is called empty iff L (B) = ε, *i.e.*, iff it does not accept any word over its alphabet. Here L (B) is called as language. For emptiness verification the state is identified by using the transition function. The transition function takes the state and a symbol as input and gives a new state as output.

```
1.  Initialize list->ø,
2.  Fprba= Fprba(F1rba,F2rba,F3rba,……,Fnrba),
3.  B=(Qprba,∑prba, δprba,q0prba,Fprba),
    ω={abcdef…..}
4.  Webverification (Brba, list)
5.  begin
6.  list.add(q0prba)
7.  emptinesssearch(ω) begin
8.  bool empt="false"
9.  state= δprba(q0prba, ω)
10. if(state is ϕ)then begin
11.   empt="true"
12. end
13. while (state≠ϕ) do begin
14.   list.add(state)
15.     if(state is in Fprba) then begin
16.         state= δprba(state, ω)
17.         if(state is ϕ)then begin
18.               empt="true"
19.               break
20.         End;       end
21. end while
22. if (empt is true) then begin
23.       sparseness problem ;end
24. else begin
25.     no sparseness problem;    end
26. end
27. Initialize list->ø, bool reach="false"
28. reachabilitysearch(Initialstate q0prba, Finalstate fprba)
    begin
29.   list.add(q0prba)
30.       for all(Qprba) begin
31.             State =next (Qprba)
32.             list.add (State)
33.             if(State in Q) begin
34. if (State==fprba)
35.                   reach="true"
36.           Exit;   end
37.         End;         end for
38.  if (reach is true) begin
39.  if is reachable
40.   end
41.   else begin
42.     if is not reachable
43.   End;end
```

Figure 4. Revise Buchi Sparseness & Reachability (*RBSR* Algorithm)

If the return state is empty then sparseness problem arises. For checking the reachability problem, the initial state and final state will be given and checks whether the destination state is reached or not. If reached means it sets the boolean variable is true, otherwise false.

4.1. WSC

Here WS-CDL is used for composing the web services [1]. The problem that concerned here is the first step of this scenario – given a request r, finding right web services for r. In particular, in this case where one has to combine multiple web services to satisfy r since no single one can. Consider the following motivating example.

Example

Suppose there are two web services available in the registry. (1) GetPlacesService returns the name of the tourist places according to the given input district. (2) Reserve Ticket returns the confirmation for the ticket reserved to the particular tourist places. I want to reserve the ticket for the particular place in a given district. Let us call this request as r. Note that neither of two Web services can satisfy r alone. GetPlacesService can only provide the name of the tourist places and reserve Ticket can only for reserving the ticket to the particular tourist places. Thus there is a need to combine both the services to satisfy the request r. Here WS-CDL is used for composing the Web Services.

The fortitude of the Web WS-CDL is to define multi-party contracts, which describe the externally observable behavior of web services and their clients (usually other Web Services), by describing the message exchanges between them.

4.2. Web Services Composition Verification

Web service is composed by using WS-CDL. WS-CDL is converted into RBA, it can be given as the input to our proposed tool WSCVT. A transition diagram is obtained from RBA and the following property has been verified using RBSR algorithm. They are:

Reachability properties. A specific condition holds in some state of the model's potential behaviors. If the particular state is not reached means it calculates how many states are visited divided by the total number of states.

$$N = \frac{\text{Number of states visited-number of empty states moved}}{\text{Total number of states.}}$$ **X100**

It is easy for identifying the visited states instead of knowing unknown states.

Sparseness Problem

The language recognized by RBA if and only if there is a final state that is both reachable from initial state and lies on a cycle.

It reduces the problem of finding the cycle and nested search not needed.

We can avoid sparseness problem through Reachability problem.

For NBA, it is decidable whether $L(A) = \phi$

Method

1. Find maximal strongly connected components (SCC) in graph A disregarding the edge labels.

2. A MSC component is called non-trivial if $C \cap F \neq \phi$

3. Find all nodes from which there is a path to a non-trivial SCC. Call the set of these nodes as N.

4. L (A) =0 iff N∩I=0
Time Complexity O (|Q|+|S|).

Safety properties. A specific condition holds in all the states of an execution path

Liveness properties. A specific condition is guaranteed to hold eventually (at some moment)

5. Case Study (Tourist Places, Transport, Hotel)

The importance of web services for electronic government has become more and more increasing, because providing services and exchanging information related to the government administration.

In this case study, we present a service oriented system that manages the lists of registered services in different application. We distinguish three different applications like Tourist Places, Transport, and Hotel respectively.

A user interacts with Provider through Registry. Provider contains all the information regarding registry web Services. User gets the registry information about web service sand taken from the provider. Provider acting as a repository databases.

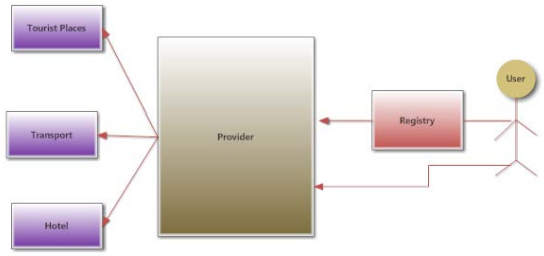

Figure 5. System Diagram

The system consists of three parts: Transport System, Hotel System and Tourist Places System.

Hotel System lists out the Hotel information in a given district. Also, it provides the services for checking the room availability and reserving the room. In Hotel System, getHotels method is used to determine the hotel information; checkRoomavailability method is used to find the availability of the rooms.

Tourist Places System provides the details about the tourist places based on the request of the user. In Tourist Places System, getPlaces method is used to determine the tourist places. For executing this method, district name is passed as an argument.

Transport System provides the functionality such as checking the seat availability, reserving the ticket, cancel the ticket. In Transport System, CheckAvailability method is used to determine the availability of the seats in a given travel on a particular date. Reserve Ticket method is used to reserve the ticket to a particular place on a particular date.

Figure 7 shows the WSCVT with automatically obtained WS-CDL code from the created related web service. But we can also generate the RBA corresponding to the WS-CDL code. The obtained RBA are used to check some properties of system with WSCVT.

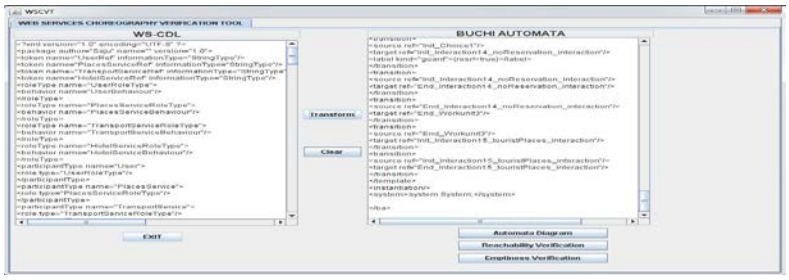

Figure 6. WSCVT-WS-CDL to BA Conversion

Figure 7 shows the automata diagram for the tourist system. This diagram is generated while clicking the Automata Diagram Button in WSCVT. This diagram describes the states, conditional variables and Work unit.

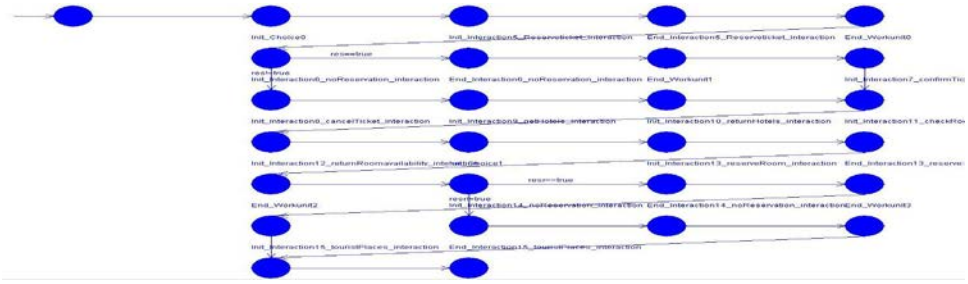

Figure 7. Automata Diagram

Figure 8 shows the Reachability Verification window. When we click the Reachability Verification Button in WSCVT, the reachability verification window will be opened. We need to specify the final state and click the Reachability button in that window, it shows the path for reaching the final state textually and diagrammatically. The STATES textbox in Reachability window shows all the available states for the Tourist domain.

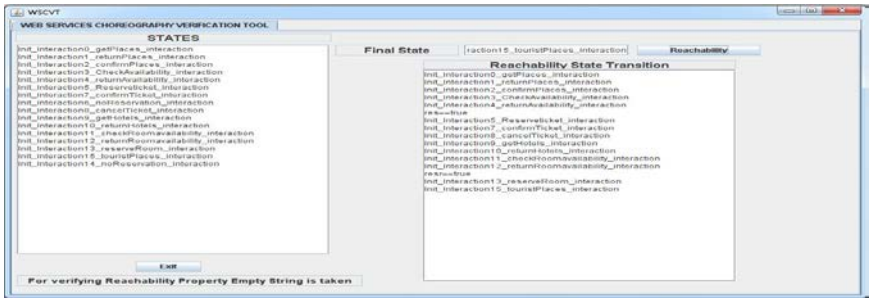

Figure 8. Reachability Verification

The following Figure (9) shows the sparseness verification window. When we click the sparseness verification button in WSCVT, the sparseness verification window will be opened. We need to specify both the input symbol and an accepting state. By using the input symbol, a set of accepting states are identified. Finally, the automata diagram is generated for all the accepting states. The symbols, other than the input symbol, are represented as an empty symbol. '?' is used as an empty symbol.

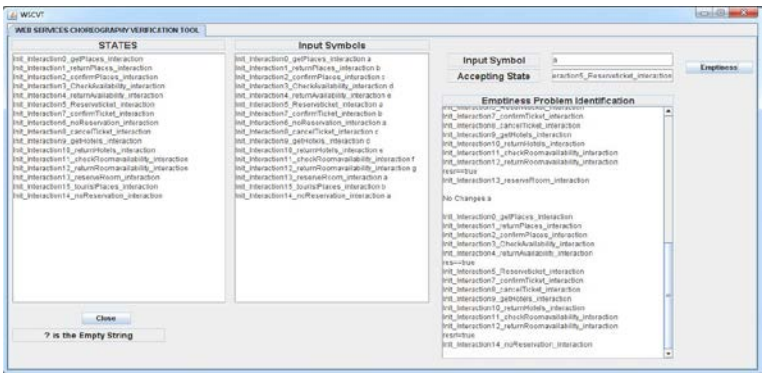

Figure 9. Sparseness Verification

If any unknown symbol is given as input, then a message box will be opened with the message "The Symbol is wrong ". If the state is invalid, then "The Accepting state is wrong" message is opened. The STATES textbox in sparseness window shows all the available states for the Tourist domain. The INPUT SYMBOLS textbox shows all the states and the corresponding symbol to reach that particular state.

In this example all the possessions verified has been gratified.

7. Comparison of WSCVT with Uppaal

UPPAAL is an integrated tool environment that allows users to model the behavior of systems in terms of states and transitions between states, and to simulate and analyze the resulting models. The generated automata representation given to the WSCVT and UPPAAL tool for verification. At the end of the verification the performance of these two tools are compared.

- Transitions without any conditional statement the reachability property is verified correctly with UPPAAL tool. But reachability property is not verified correctly when conditional statements are available in the automata representation. This problem is avoided in the proposed tool WSCVT.
- Sparseness problem is not addressed by the UPPAAL tool. In WSCVT sparseness problem is addressed.
- The reachability graph for the given final state will be shown in the WSCVT tool but not in UPPAAL tool.
- WSCVT is fully developed by java. So it is platform independent. But UPPAAL is developed by using C++ and only the GUI is developed using java. So it is platform dependent.

The following table gives the summary of the comparison between UPPAAL tool and WSCVT tool.

Table 6. Comparison of WSCVT with UPPAAL

Problem	UPPAAL	WSCVT
Reachability	✖	✓
Emptiness	✖	✓
Automata Diagram for individual state	✖	✓
Platform independence	✖	✓

8. Comparisons with Existing Work and Proposed Work

Table 8.1. Comparison of Existing Work and Proposed Work

S.No	Researchers	Formal Model	Transformation Verification				Type of System	Specification	Tool Used	Level of Development
			Reachability	Emptiness	Correctness Property	Safetyness				
1	M.Emilia Cambronero	Timed Automata	No	Yes	No	No	Deterministic System	WSCDL	WST and UPPAAL	Design Level
2	Jia Mei	Interface Automata	No	Yes	No	No	Deterministic System	BPEL, Promela	SPIN	Implementation Level
3	Jin Song Dong	Orchestration on computation via Timed automata	No	Yes	No	No	Deterministic System	LTL	UPPAAL	Design Level
4	Valentin Valero	Colored Petrinet Method	Yes	Partial	Partial	No	Deterministic System	BPEL	CPN	Implementation Level
5	Guangquan Zhang	Refinement Checking Method	No	No	No	No	Deterministic System	BPEL	UPPAAL	Design Level
6	Our Work	Revise Buchi Automata	Yes	Yes	Yes	Yes	Both Deterministic and non-Deterministic System	WS-CDL	WS_CVT	Implementation Level

The above tabular column shows the comparing existing approach with proposed approach.

Only limited number of techniques is used for verifying the composed web services. Even though the models are used for verifying the composed web services, it in suitable only for deterministic rather than non-deterministic system. But our system covers all the problems and shown the performance result.

9.1. Performance evaluation-statistical approach

The Reachability problem and sparseness problem solved by using RBSR algorithm.

9.1.1. Coefficient of Variance

Coefficient of variance is the ratio of standard deviation to the mean
Table 1: Dead transition values for different models

a. Reachability

S.No	Number of States	Reachabiliy	
		Proposed model using RBA	Existing TA
1	8	75	25
2	12	78	58
3	16	82	64
4	20	86	75

$$RF_{rba} = \frac{\text{Number of states moved - Empty string states moved}}{\text{Total number of states}} \times 100$$

$$RF_{rba} = \left(F_p - Esm_p\right) / Q_{0p} \times 100$$

Let us consider the example 7-1/8*100=6/8*100=75. Where 7 is the number of states moved using the diagram 7and 1 is the empty string state is moved referred in the Figure 5, 8 is the total number of states. Finally the output will be the 75%.

Mean $\overline{x} = \sum x / n$ =83.75. Adding the values of RB and sum can be divided by total number of terms ie 4.

Variance $\sigma^2 = \sum (xi - \overline{x})^2 / n$ = 22.31. Subtract each value of RBA from X_1, find the square of the value and finally calculate the sum . This sum can be divided by total number of terms.

Standard Deviation S.D=$\sqrt{\sigma^2}$ = 4.72.Finding square root of the variance is called Standard deviation.

Coefficient of variance CV=Standard Deviation/Mean

= 4.72/83.75

=0.053

Similarly, calculate the mean, variance and standard deviation for the Timed Automata-sparseness property values

$$RF_{ta} = \frac{\text{Number of states moved}}{\text{Total number of states}} \times 100$$

Or $RF_{ta} = (SM_p) / Q_{0_p} \times 100$

Let us consider the example 2/8*100=25. Where2is the number of states moved using the diagram 7, divvied by total number of states .Finally the output will be the 25%./

Mean $\bar{x} = \sum x / n$ =56

Variance $\sigma^2 = \sum (xi - \overline{x^2}) / n$ =347.5

S.D= $\sqrt{\sigma^{\ 2}}$ =18.64

Coefficient of variance CV=Standard Deviation/Mean

= 18.64/56 =0.33

Here RBA value is lesser than the TA value. So TA is riskier than RBA model

9.1.2. T.Test

It can be used to determine if two sets of data are significantly different from each other, and is most commonly applied when the test statistic would follow a normal distribution if the value of a scaling term in the test statistic were known. When the scaling term is unknown and is replaced by an estimate based on the data, the test statistic (under certain conditions) follows a Student's *t* distribution.

$\overline{x_1} = \sum x_1 / n$ =83.75, this value is obtained by adding the RBA values and divided by number of terms.

$\bar{x}_2 = \sum x_2 / n$ =56, similarly for x2 also. n1=4 and n2=4

$\sigma^2 d = \sigma_1^{\ 2} / n1 + \sigma_2^{\ 2} / n2$, where $\sigma_1^{\ 2}$ is the value of standard deviation for HA and $\sigma_2^{\ 2}$ is the value of standard deviation for TA.

= 22.31/4+347.5/4

= 92.45

Standard Deviation (S.D) $\sigma d = \sqrt{\sigma^2 d}$ = 9.61

T= $\bar{x}_1 - \overline{x_2} / \sigma d$ = 2.88. Where T is obtained by subtracting the value first mean and second means which is divided by standard deviation.

Enter T-table at (n1+n2-2) degrees of freedom *i.e.*, 4+4-2=6
The calculated value is 2.88 and Tabulated value for 6 degrees of freedom in p=0.05 is=2.45 in table [21].

So, concluding that the calculated value is greater than the tabulated value. So there is a difference between these two *i. e.*, p=0.05, 95% difference with the model TA than the RBA.

b. Sparseness Property

9.1.3. Coefficient of Variance

S.No	Number of States	Emptiness	
		Proposed model using RBA	Existing TA
1	10	80	60
2	14	93	86
3	16	94	81
4	20	90	80

$$RF_{rba} = \frac{\text{Number of states moved} + \text{Empty string states move}}{\text{Total number of states}} \times 100$$

Let us consider the example 6+2/10*100=80. Where 6 is the number of states moved using the Figures 7and 2 is the empty string state is moved referred in the Figure 7, 10 is the total number of states. Finally the output will be the 80%.

Mean $\overline{x} = \sum x/n$=89.25. Adding the values of RB and sum can be divided by total number of terms ie 4.

Variance $\sigma^2 = \sum (xi - \overline{x})^2 / n$ = 31. Subtract each value of RBA from X_1, find the square of the value and finally calculate the sum . This sum can be divided by total number of terms.

Standard Deviation S.D=$\sqrt{\sigma^2}$ = 5.56.Finding square root of the variance is called Standard deviation.

Coefficient of variance CV=Standard Deviation/Mean

= 5.56/89.25

=0.062

Similarly, calculate the mean, variance and standard deviation for the Timed Automata-sparseness property values

$$RF_{ta} = \frac{\text{Number of states moved}}{\text{Total number of states}} \times 100$$

Or $RF_{ta} = (SM_p)/Q_{0_p} \times 100$

Let us consider the example 6/10*100=60. Where 6 is the number of states moved using the diagram 7, divvied by total number of states .Finally the output will be the 60%./

Mean $\bar{x} = \sum x / n$ =76.75

Variance $\sigma^2 = \sum (xi - \overline{x^2}) / n$ =99

S.D= $\sqrt{\sigma^{\ 2}}$ =9.94

Coefficient of variance CV=Standard Deviation/Mean

= 9.94/76.75 =0.129

Here RBA value is lesser than the TA value. So TA is riskier than RBA model.

9.1.4. T. Test

It can be used to determine if two sets of data are significantly different from each other, and is most commonly applied when the test statistic would follow a normal distribution if the value of a scaling term in the test statistic were known. When the scaling term is unknown and is replaced by an estimate based on the data, the test statistic (under certain conditions) follows a Student's *t* distribution.

$\overline{x_1} = \sum x_1 / n$ =89.25, this value is obtained by adding the RBA values and divided by number of terms.

$\bar{x}_2 = \sum x_2 / n$ =76.75, similarly for x2 also. n1=4 and n2=4

$\sigma^2 d = \sigma_1^{\ 2} / n1 + \sigma_2^{\ 2} / n2$, where $\sigma_1^{\ 2}$ is the value of standard deviation for HA and $\sigma_2^{\ 2}$ is the value of standard deviation for TA.

= 31/4+99/4

= 42.2

Standard Deviation (S.D) $\sigma d = \sqrt{\sigma^2} d$ = 6.49

T= $\bar{x}_1 - \overline{x_2} / \sigma d$ = 1.92. Where T is obtained by subtracting the value first mean and second means which is divided by standard deviation.

Enter T-table at (n1+n2-2) degrees of freedom *i. e.*, 4+4-2=6
The calculated value is 1.92 and Tabulated value for 6 degrees of freedom in p=0.1 is=1.94 in Table [24].
So, concluding that the calculated value is greater than the tabulated value. So there is a difference between these two..ie p=0.1, 90% difference with the model TA than the RBA.

10. Related Works

We now discuss related work on WSC verification. Recent survey [1] which can be roughly divided into (i) approaches that are based on Timed Automata [1, 4, 10, 11, 12]. These papers based on Timed Automata and Computation Orchestration. WSCDL used for composition of web workflow net models are used and define a fully-automatic translation of

this formalism into conceptual BPEL by means of the tools 4BPEL framework. The automatic generation of BPEL code by using the workflow net formalism as design model, which can be squared of being free of deadlocks and live locks.

This proposal then takes the orchestration viewpoint. Whereas, we take choreographic one. Timed Automata applicable only for deterministic system rather than non-deterministic System.

We discuss the (ii) approaches that are based on Interface Automata [2, 7 and 13]. It is based on the verification of the composed web services using interface automata. Initially composition done by using BPEL4WS which can be converted into Interface automata mapped into Promela and then verified using the tool called SPIN. This approach is applicable for Deterministic system rather than Non-Deterministic system.

We now discuss (iii) approaches that are based on Petrinet [3, 5, 6, 14, 15, 17, 18, 19, 20, 23] used Petri-nets presented a methodology for the design , verification and validation of composite web services using WS-CDL as the language for describing web service interactions., and petrinets as a formalism (that allows us to analyze the described systems). In this work they have considered timed automata and prioritized collaboration in composite web services, so the considered model of Petri nets is a prioritized version of Timed Petri nets. Different WS-CDL which was one of the first proposals for choreography descriptions. However, WS-CDL has a more affluent expressivity than WSCI, as stated in the comparative work of [7].

Composition of web services described using [8] orchestration and WS-BPEL, used Label Transition Systems (LTS), which is obtained by using Finite State Process (FSP) as an intermediate language. The main difference to our work is that, this work is more generalized rather than automata.

WS-CDL can be mapped into BPEL4WS converted into CSP [16] with verifying if they obtained orchestrations behave as specified in the corresponding choreography. This is an informal mapping in the form of conversion tables, and also the considered subsets of WS-CDL and WS-BPEL are quite basic.

The existing models are suitable deterministic system but our proposed model is applicable for non-deterministic system. So it avoids the sparseness and reachability problem by 95% than existing system.

11. Conclusions and Future Work

Web service is a self-describing, self-contained and interoperable modular application that can be published, discovered and invoked over a network, usually the Internet. Since the web has evolved as a service **provider** in all areas, there are problems to be addressed. Some challenges faced by web services are related to Security, Quality of Service and Composition. Among these challenges Composition turns out to be an area of major research. In this paper, we have presented a novel approach for the verification of WSC. We have created the web services and composition done by using WS-CDL. In WSC the interaction between the services are specified in WS-CDL. Then, we present a technique for the translation of WS-CDL into RBA. Finally, the WSCVT tool is used to simulate, validate and verify some properties of the composed web service. The above mentioned properties verified by the WSCVT.

The proposed tool is compared with UPPAAL tool. The problems with the UPPAAL tool are identified and solved with WSCVT tool.

As future work we plan to use the WSCVT with orchestration. Also we plan to enhance the WSCVT tool.

References

[1] M. Emilia Cambronero, G. Diaz, V. Valero and E. Martinez, "Validation and Verification of Web servicesChoreographies by using timed autroata", The Journal of Logic and Algebraic Programming, doi:10.1016/j.jlap.2010, vol. 80 **(2011)**, pp. 25-49.
[2] J. Mei, H. Miao, Y. Chen and H. Gao, "Verifying Web Service composition based on Interface Automata using SPIN", AICIT International Journal of Digital Content Technology and its applications, vol. 4, no. 8, **(2011)**.
[3] V. Valero, M. Cambronero, G. Díaz and H. Macià, "A petri net approach for the design and analysis of Web serviceschoreographies," The Journal of Logic Algebraic Programming, **(2009)**, pp. 359-380.
[4] J. S. Dong, Y. Liu, J. Sun and X. Zhang, "Verification of computation orchestration via timed automata", International Conference on Formal Engineering Methods (ICFEM), pp. 226-245, **(2006)**.
[5] H. Kang, X. Yang and S. Yuan, "Modeling and Verification of Web servicesComposition based on CPN", Proceedings of IFIP International Conference on Network and Parallel Computing – Workshops, 0-7695-2943-7/07 $25.00 © 2007 IEEE.
[6] J-J. Le and F. He, "Automatic Web services Composition Based on Reasoning Petri Net", International Conference on Advanced Language and Web Information Technology, DOI 10.1109/ALPIT.2008.83, IEEE.
[7] L. Alfaro and T. Henzinger, "Interface automata", Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering, **(2001)**, pp. 109-120.
[8] M. Bravetti and G. Zavattaro, "A theory for strong service compliance, Coordination Models and Languages", 9th International Conference, Coordination 2007, Paphos, Cyprus, Proceedings, Lecture Notes in Computer Science, vol. 4467, Springer, **(2007)** June 6-8, pp. 96-112.
[9] A. Brogi, C. Canal, E. Pimentel and A. Vallecillo, "Formalizing Web Service choreographies", Electron, Notes Theor. Comput. Sci., vol. 105.
[10] M. E. Cambronero, G. Díaz, E. Martínez, V. Valero and L. Tobarra, "WST: a tool supporting timed composite Web services model transformation", Simulation: Transactions of the Society for Modeling and Simulation International.
[11] G. Díaz, J. J. Pardo, M. E. Cambronero, V. Valero and F. Cuartero, "Verification of Web serviceswith timed automata", Proceedings of First International Workshop on Automated Specification and Verification of Web Sites, Electronic Notes in Theoretical Computer Science, vol. 1125, 157/2, **(2005)**.
[12] J. S. Dong, Y. Liu, J. Sun and X. Zhang, "Verification of computation orchestration via timed automata", International Conference on Formal Engineering Methods (ICFEM), **(2006)**, pp. 226-245.
[13] X. Du, C. Xing, L. Zhou and Z. Li, "Nested Web Service interface control flow automata", Proc. 4th International Conference on Next Generation Web servicesPractices, **(2008)**, pp. 129-136.
[14] H. Foster, W. Emmerich, J. Kramer, J. Magee, D. S. Rosenblum and S. Uchitel, "Model checking service compositions under resource constraints", ESEC/SIGSOFT FSE, **(2007)**, pp. 225-234.
[15] R. Hamadi and B. Benatallah, "A petri net-based model for Web Service composition", Proc. 14th Australasian Database Conference, vol. 17, **(2003)**, pp. 191-200.
[16] N. Lohmann and J. Kleine, (Fully-automatic translation of openworkflownet models into simple abstract bpel processes), Modellierung, **(2008)**, pp. 57-72.
[17] N. Sharygina and D. Kröning, "Model checking with abstraction for Web Services, Test and Analysis of Web Services", Springer, **(2007)**, pp. 121-145.
[18] V. Valero, M. Cambronero, G. Díaz and H. Macià, "A petri net approach for the design and analysis of Web serviceschoreographies", J. Logic Algebraic Program, vol. 78, no. 5, **(2009)**, pp. 359-380.
[19] W. M. P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl and K. Wolf, "From public views to private views – correctness-by-design for services", WS-FM, **(2007)**, pp. 139-153.
[20] H. Yang, X. Zhao, C. Cai and Z. Qiu, "Model-checking of Web serviceschoreography", IEEE International Workshop on Service-Oriented System Engineering, **(2008)**, pp. 79-84.
[21] W. L. Yeung, "Mapping WS-CDL and BPEL into CSP for behavioral specification and verification of Web Services", European Conference on Web services(ECOWS), **(2006)**, pp. 297-305.
[22] X. Zhou,W. T. Tsai, X. Wei, Y. Chen and B. Xiao, "Pi4SOA: a policy infrastructure for verification and control of service collaboration", ICEBE'06: Proceedings of the IEEE International Conference on e-Business Engineering, IEEE Computer Society, Washington, DC, USA, **(2006)**, pp. 307-314.
[23] J. Zhu Kan Zhang Guangquan Zhang, "Verifying Web servicesComposition based on LTL and Colored Petrinet", The 6th International Conference on Computer Science & Education (ICCSE 2011), Super Star Virgo, Singapore, **(2011)** August 3-5.
[24] Statistical analysis for model checking referred in http://archive.bio.ed.ac.uk/jdeacon/statistics/table1.html.

