A NOVEL APPROACH FOR WEB SERVICES COMPOSITION AND VERIFICATION

E. Iławarasan1, N. Danapaquiam2 & J. Sankari3

1Associate Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, (E-mail: eilavarasan@pec.edu)
2Research Scholar, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, (E-mail: n.danapaquiam@gmail.com)
3Pondicherry Engineering College, Pondicherry-605014, (E-mail: sankari68@pec.edu)

Abstract: Web services are a very appropriate communication mechanism to perform distributed business processes among several organizations. A concept of service composition, combining existing Web services easily as our needs change, is one of the key features of the technology. Many new standards have been defined to solve web service composition problem, for example BPEL4WS. BPEL4WS provides an initial work for forming an XML specification language for defining and implementing business process workflows for web services. The main problems with most practical approaches to service composition are the verification of correctness of service composition. It has to depend on formal verification method which helps to check the correctness of services composition. The proposed system concentrates on both composition and verification of web services. In this paper, an algorithm is introduced for the verification of web services composition using Muller Automata. The web services are composed using BPEL4WS and it is transformed into Promela (It is the input language of SPIN model checker). Finally, verification algorithm is designed and verified using Muller automata.

Keywords: Composition, Muller Automata, Promela, and BPEL4WS.

1. INTRODUCTION

Web services are distributed and independent computational elements that solve specific tasks, varying from simple requests to complex business processes, and that communicate using XML messages following the SOAP standard. Web services can help to solve the interoperability problem by giving different applications a way to link their data. Web service is platform and language independent. Its platform elements are SOAP and WSDL used for service interactions, and UDDI. In this the service requester will finds required services via the service broker and binds to services via service provider. The service provider provides e-business services and publishes availability of these services through a registry. The service registry provides support for publishing and locating services. By using single web service we can’t get our works done, so Web Services Composition (WSC) is brought out. Generally, end-users find to use appropriate Web services that are located somewhere in the Internet. Further, it sometimes requires combining more than one Web services to fulfill their needs. Web Services Composition is the ability to create new value added services by incorporating some existing web services together. In WSC, WS discovery and selection are applied to achieve process dynamics by finding and integrating Web services to a predefined abstract service process at run-time. Web service discovery refers to the process of finding functionally suitable Web service selection deals with comparing and choosing appropriate Web services from a known set of descriptions according to customized preferences, including non-functional constrains or requirements, e.g. quality of services (QoS). WS selection is often used after WS discovery to render more accurate and desirable results, especially when several Web services with the same functionalities are discovered.

In recent years, many Web service composition languages have been proposed.
There are two different viewpoints in the area of Web service composition (Orchestration and Choreography). Orchestration combines available services by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single sub-activities. Web service Choreography, does not assume the exploitation of a central coordinator but rather defines complex tasks via the definition of the conversation that should be undertaken by each participant. While several proposals exist for orchestration languages (e.g. BPML and BPEL) and choreography languages are still in a preliminary stage of definition. An initial proposal WS-CDL was issued few years back by the World Wide Web Consortium (W3C).

In this paper, an algorithm is introduced for the verification of web services composition using Muller Automata. The web services are composed using BPEL4WS and it is transformed into Promela (It is the input language of SPIN model checker). Finally, verification algorithm is designed and verified using Muller automata.

2. BACKGROUND CONCEPTS UNDERPINNING PROPOSED WORK

2.1. BPEL4WS

The Business Process Execution Language for Web Services (BPEL4WS), which is also referred to as BPEL [2], is currently a de facto standard for building, specifying and executing business processes for web services composition and orchestration. BPEL composes web services to get a specific result. The composition result is named a process, involved services are called partners, and message exchange is referred to as an activity. In other words, a process contains a set of activities and it invokes external partner services using a WSDL interface. A BPEL process defines the order in which involved Web services are composed, either in sequence or in parallel. BPEL allows describing conditional activities. An invocation of a Web service can for example rely on the result of another web service’s invocation. With BPEL, it is possible to create loops, declare variables, copy and assign values as well as to use fault handlers. Complex business processes can be built algorithmically by using all these constructs. It can be helpful to describe business processes graphically through UML (Unified Modelling Language) activity diagrams. BPEL supports two different ways of describing business processes that support orchestration and choreography:

1. Executable processes allow for specifying the details of business processes. They follow the orchestration paradigm and can be executed by an orchestration engine.

2. Abstract business protocols allow specification of the public message exchange between parties only. They do not include the internal details of process flows and are not executable. They follow the choreography paradigm.

The Structure of a Business Process

The role of BPEL is to define a new web service by composing a set of existing services through a process-integration type mechanism with control language constructs. To build a BPEL process, the following elements are required:

1. Business partners which will interact with the process.

2. Information about the data exchange type between the process and the business partners.

3. A workflow that defines the order of process execution (receive and invoke web services, map data and reply to business partners).

4. A BPEL process needs a WSDL file in order to create an executable BPEL definition. The WSDL file consists of the namespace, partner link types, operations, and messages which are required to define process activities.

5. A BPEL must have namespaces which point to associated WSDL schema locations and other resources such as XSL style sheets and xml files used in resource catalogue custom functions.

A process consists of a set of activities. It interacts with external partner services through a WSDL interface. To define a BPEL process, we use:
1. A BPEL source file (.bpel) which describes the execution order, activities and conditional behaviours.

2. A process interface (.wsdl) that defines the ports, the namespace, partner link types, operations, and messages which are required to determine process activities, and WSDL files are needed in order to create a valid, executable BPEL definition. For the specification of a business process, BPEL4WS provides activities and distinguishes between basic activities and structured activities.

**Basic Activities**

The basic activities are:

- `<invoke>`: invoking other Web services’ operations.
- `<receive>`: waiting for the client to invoke the business process by sending a message.
- `<reply>`: generating a response for synchronous operations.
- `<assign>`: manipulating data variables.
- `<throw>`: indicating faults and exceptions.
- `<wait>`: waiting for some time.
- `<terminate>`: terminating the entire process.

**Structured Activities**

The main structure activities are the following:

- `<sequence>`: defining a set of activities that will be executed in an ordered sequence.
- `<flow>`: defining a group of activities which will be invoked in parallel.
- `<switch>`: Case-switch construct for implementing branches
- `<while>`: defining loops.
- `<pick>`: selecting one of several alternative paths.

2.2 Muller Automata

In automata theory, Muller automaton is a type of a \(\omega\)-automaton (\(\omega\)-automata are finite automata on infinite words). The acceptance condition separates a Muller automaton from other \(\omega\)-automata. The Muller automata is defined using Muller acceptance condition, i.e. the set of all states visited infinitely often must be an element of the acceptance set. Muller-automaton is a tuple \(M = (Q, \Sigma, \delta, q_0, F)\)

\[Q\] is a finite set. The elements of \(Q\) are called the \textit{states} of \(Q\).

\[\Sigma\] is a finite set called the \textit{alphabet} of \(A\).

\[\delta: Q \times \Sigma \rightarrow \text{ is a function, called the transition function of } A.\]

\[q_0\] is an element of \(Q\), called the initial state.

\[F\] is the \textit{accept component}, \(F \subseteq \text{pow}(Q)\) if it is used with the following acceptance condition:

**Muller Acceptance:** We say, \(M\) accept a \(\omega\)-word \(\alpha \in \Sigma\) if and only if there exist a run \(r\) of \(M\) on \(\alpha\) satisfying

\[\text{Inf}(r) \in F\]

i.e the set of infinitely often visited states are exactly one of the set in \(F\).

2.3. Promela and Spin

PROMELA is the language used in SPIN to represent concurrent systems with abstraction. PROMELA programs consist of processes, message channels, and variables. Processes are global objects that represent the concurrent entities of the distributed system. Message channels and variables can be declared either globally or locally in a process. PROMELA supports rendezvous and asynchronous communication between processes via channels. Processes specify behavior, while channels and global variables define the environment in which the processes run.

SPIN (Simple ProMeLa INterpreter) is a verification tool for models of distributed software systems. SPIN takes a model of the system design and a requirement as input and the model checking algorithm specifies whether the system design meets the requirement or not. SPIN verification is focused on proving the correctness of process interactions; not much importance is given to internal computations of the processes.
3. RELATED WORK
Several works have been done to verify web service composition. In recent years, many researchers focus on this problem. Huang [4] classified the formal models describing the Web services composition into two main categories: models based on state transition and models based on workflow. But specific verification method for BPEL4WS was not presented. In [5], Jia Mei et al, Proposed a framework to compose lots of specialized services flexibly, BPEL4WS is proposed to describe Web services composition. In this approach, BPEL4WS is mapped into interface automata, which is then transformed into Promela (It is the input language of SPIN model checker.), and the correctness properties of Web services composition can be verified by SPIN. In [6], YanPing Yang et al, proposed paper is to provide a transformation from BPEL to hierarchical CP-nets. the hierarchical modeling technique is by means of substitution transitions, and therefore if a transition represents a sub activity, there always remains the possibility of decomposing it into various actions. Next, we will translate activity of BPEL to hierarchical CP-nets constructs. In [7], Srini Narayanan et al, proposed DAML-S DAML+OIL ontology for describing the capabilities of Web services. With the semantics in hand, we encode our service descriptions in Petri Net formalism and provide decision procedures for Web service simulation, verification and composition. In [8], Nakashiro et al, translated BPEL to Promela and verified using SPIN model checker.

4. PROPOSED WORK

4.1. Overview
Orchestration methods have already gained rich experience due to the development of process oriented information systems. Many enterprises adopt BPEL for integrating and assembling Web services. As choreography methods use WS-CDL for composition, it does not yet satisfy the correctness requirement but composition using BPEL4WS is used more by researchers. However, BPEL4WS lacks formal semantics and BPEL models can therefore not be verified directly, we can use formal verification methods for checking the correctness properties of web services composition. The motive is to address the problems during composition and verifying the composed web services to avoid deadlock, reachability problem and incompatible behaviours by using formal methods.

The proposed system concentrates on both composition and verification of web services. In the proposed work, an algorithm is introduced for the verification of web services composition using Muller Automata. The web services are composed using BPEL4WS and it is transformed into Promela (It is the input language of SPIN model checker). Finally, an algorithm is designed, implemented and verified using Muller Automata. The advantage of the proposed system is using formal methods (i.e., Automata) for verification that avoids deadlock, incompatible behaviours and reachability problem during composition and checks the correctness properties of the composed web services.

4.2. Proposed Framework
The Figure 1 depicts the proposed framework and illustrates how the Web services are composed, verified using verification tool. Our proposed architecture constitutes two parts. First one is Web services composition and another one is verification of composite Web services. 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 BPEL4WS. The output of the composition part will be given as the input to the verification part. During verification, BPEL4WS is translated into Promela and a state transition diagram will be generated by Promela files when it is given as input to the SPIN Model checker. Finally, it is verified using Muller verification algorithm for checking the correctness, reachability and safety properties of web services composition.

4.3. Steps for Composition
1. Receive the user request
2. Get the required services
3. Declare input and output variable
4. Invoke the operation
5. Reply for the received message
6. Establish partner links for the services
7. Create or import BPEL process files into the BPEL Module project.
8. Create or import WSDL resources to act as partner services in your business process.
9. Import XML schema resources.
10. Add to the source code of the BPEL, WSDL, and XSD files.
11. Build a BPEL Module project.
12. Deploy the Composite Application project to the BPEL Service Engine runtime.
13. Test-run BPEL processes by sending sample messages to the deployed process.

Because the BPEL process communicates with other Web services, it depends on the WSDL descriptions of the Web services invoked by the composite Web service. A simple example shows a global view to how a BPEL process is designed.

**Example:**

Travel Reservation application, explains the BPEL constructs in this section. In this example, we show a client making a request in order to buy a flight ticket, rent and reserve a room for his travel. The Travel Agent (central process) will communicate with a hotel service; a car rent service and an airline service to fulfill the client’s wishes. Once a reservation is made, it is sent back to the client.

To understand how business processes are described with BPEL, we will define a simplified business process for a travel reservation agent: The client invokes the business process, specifying his name, the destination, the departure date and the return date. The BPEL process checks if it is possible to book the flight ticket with Airline A. We assume that Airline Company A provides a Web service through which such bookings can be made. Then the BPEL checks if it is possible to rent a car from CAR-Rent Company. We assume as well that the CAR rent Company provides a web service through which such renting can be made. Finally, the BPEL process reserves a room from a hotel and returns the travel reservation summary to the client. We build a synchronous BPEL process. We assume that the Web services for booking the airplane ticket and the hotel as well as renting the car are synchronous because such data can be obtained immediately and returned to the requestor.

4.5. Web Services Verification

In verification process the following properties are verified [3]. They are

- **Safety** assurance that the composition is deadlock free and is checked against partial correctness of transitions
- **Correctness** assurance that the composed services are correct
- **Reachability** assurance that whether it is possible for a process to achieve the desired result.

---

4.4. Web Services Composition

In this paper, BPEL4WS is used for composing web services. Consider a typical scenario, the BPEL business process receives a request. To fulfill it, the process invokes the involved Web services and then responds to the original caller.
• **Liveness** assurance against starvation of progress (that the service process eventually terminates) and that messages received are served on a first-come-first-served basis

• **Deadlock** assurance that the composed services are deadlock free

5. **CONCLUSION**

In this paper, we present introduced an approach to verify and analyze Web Services composition based on transformation BPEL4WS to Promela. We construct the mapping in the semantic from BPEL4WS to Promela, design an algorithm for Muller automata, and then verify Web applications by Muller verification algorithm. The experiment shows that our approach can reduce reachability, deadlock and emptiness problem effectively in verification process.

**References**


