Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.6, 2012

Test Sequences for Web Service Composition using CPN model

Poonkavithai Kalamegam* and Dr. Zayaraz Godandapani
Dept of CSE, Pondicherry Engineering College, Pondicherry-605014, India
* E-mail of the corresponding author: poonks2012@gmail.com

Abstract

Web service composition is most mature and effective way to realize the rapidly changing requirements of business in service-oriented solutions. Testing the compositions of web services is complex, due to their distributed nature and asynchronous behaviour. Colored Petri Nets (CPNs) provide a framework for the design, specification, validation and verification of systems. In this paper the CPN model used for composition design verification is reused for test design purpose. We propose an on-the-fly algorithm that generates a test suite that covers all possible paths without redundancy. The prioritization of test sequences, test suite size and redundancy reduction are also focused. The proposed technique was applied to air line reservation system and the generated test sequences were evaluated against three coverage criteria; Decision Coverage, Input Output Coverage and Transition Coverage.

Keywords— CPN, MBT, web service composition testing, test case generation

1. Introduction

Web Services is a set of distributed message oriented interacting components. It is the most active and widely adopted implementation of SOA which is a design pattern composed of loosely coupled, discoverable, reusable, inter-operable platform agnostic services that follow a well defined standard [1]. Success of any deployment in an enterprise depends on the quality assurance process undertaken. Web Service composition testing is quite different from traditional testing. It requires its own type of test architecture and tester skills. To test any composition, all the web services needs to be tested in isolation along with the common use cases where the services are interdependent. The services can be composed by following two complementary views; choreography and orchestration. In Orchestration a central element controls the business logic and execution order of the interactions. In Choreography interactions may involve multiple parties and multiple sources, but each element of the process is autonomous and controls its own agenda. Hence service composition testing can be classified into choreography-based and orchestration-based testing.

Software testing is one of the most crucial phases in any SDLC to assure the quality of software. Creation of test cases consumes major effort allocated for testing. Model Based Testing (MBT) is a form of black-box testing technique that uses behavioural models of the system to automate the test case design process. Colored Petri nets (CPN) provide a framework for the construction and analysis of distributed and concurrent systems. A CPN model of a system describes the states which the system may be in and the transitions between these states. CPN have been applied in a wide range of application areas particularly in software system designs and business process re-engineering. Extending the usage of Colored Petri Nets beyond the system design phase, particularly in test design phase proves to be very effective in terms of greater test coverage and reduced test effort.

In this paper a novel idea of applying MBT technique on the CPN model used for verifying the web service composition is presented. The paper is organized as follows. In next section, we discuss the motivation for our approach. In section III, we introduce usage of CPN model in web service composition. In section IV, the approach to automate test case generation for testing web service composition is described in detail. We present results and discussions in section V followed by conclusions in the last section.

2. Motivation

MBT provides a solid foundation for automating the test design process by generating the test cases from the business requirements that are formally represented by a model. UML (Unified Modeling Language), FSM (finite state machines) and CPN are widely used modeling mechanisms to specify, analyze and simulate requirements of the software, for test automation and for model-based software testing. CPN is a mathematical model and hence has

better formal capabilities to specify and analyze even complicated behaviours of a system. Moreover CPN model could be simulated dynamically, directed by the data-dependent control flow of system behaviours. It is better in modeling, analysis and validation of the accuracy of the system functional models. There exist few approaches to derive test cases from the CPN model. A simple approach to generate test cases using the state space is proposed in[2].The advantage of this approach is that the correctness of the specification based on CPN can be validated by simulation tools and the state space can be also generated by state space tools. An efficient approach for building conformance test suite using the PN-ioco relation is proposed in [3]. U. Farooq, C. P. Lam and H. Lin [4] proposed a method to convert the AD activities to a CPN model and apply the Random Walk Algorithm to create test sequences. Harumi Watanabe and Tomohiro Kudoh [5] proposed two techniques that can be applied for concurrent systems. One uses CPN-Tree and the other uses Colored Petri net Graph (CP-graph). CPN-Graph is considered as FSM and existing test case generation methods based on FSM is applied. In CPN-Tree method the reachability trees reduced by the equivalent marking technique are used to achieve the practical test suite length.

There are several methods for automatic test cases generation using the BPEL structure of the composite web service. MBT [6] can be used along with Symbolic Execution, Model Checking and Petri Nets for testing and verification of the web service composition. In this paper we will limit to testing case generation. Model Checking technique is used in [7, 8]. The Timed Extended Finite State Machines (TEFSM) is used for widely due existing tool support [9-13]. The BPEL specification is transformed into the TEFSM model and then used for test case generation. The Control Flow Graph(s) (CFG) are used to represent the BPEL processes. In [14] the test cases are generated using Graph Search Algorithm(s) and test data using Path Analysis (using constraint solving) (PA). In [15], an automated test data generation framework that extends CFG to represent the BPEL activities as the edges is proposed. Another extended CFG [16] called BPEL Flow Graph (BFG) that contains both control and data flow of a BPEL process is used for test data generation and semantic information such as dead paths. Hou et al. [17] suggests that the BPEL processes can be modelled as a Message Sequence Graph (MSG) from which test cases are generated. Guangquan et al. [18] propose the use of UML 2.0 Activity Diagram to model the BPEL process and a depth first search method combined with the test coverage criteria is used to generate test cases. Tarhini et al. [19, 20] proposed two abstract models; the Task Precedence Graph (TPG) and the Timed Labelled Transition System (TLTS) to represent system under test. The TPG models the interaction between services and the TLTS models the internal behaviour of the participating web services.

Most of the existing approaches take into account the BPEL specifications alone for creating the test cases and the rich information on data available in the WSDL schema is not utilized effectively. On one hand CPN models have been used in test case generation but not in web service composition testing. On the other hand there are various approaches to verify web composition using CPN model. We aim at bringing in CPN models into MBT techniques to create cost effective and efficient test sequences for validating any composition. In this paper we propose an algorithm to generate of test sequences from the CPN model used for verifying the design of web service composition.

3. Need for Using CPN

The CPN model provides a formal description of the web composition a system. However it is still possible that more than one implementation is derived from the same specification and is not compatible. This is mainly due to incorrect implementation of the composition. Hence there is a need for testing every implementation for conformance to the business requirements. Testing can be carried out using test sequences generated from the business requirements. Control flow testing focuses on the transfer of control, while data flow testing focuses on the definitions of data and their subsequent use. The execution of a service is driven by the received and manipulated data; hence validation of data flow is very critical in composition. The data flow pertains to service messages namely the request and response messages through which data is exchanged among the participating services to accomplish a business goal. Moreover semantics of data structures in Web service composition is complex; for e.g. service messages are XML documents and service functions are XPath expressions.. CPN is one model that combines data and control flow with behavioural aspects of a given business requirement. Control flow coverage is defined in terms of transitions fired while the data flow coverage is defined in terms of tokens used. The web service composition is modelled and analysed for the accuracy of the functional design. Hence generating test sequences from such a model

is bound to produce completely feasible, effective and efficient test sequences for practical test executions. Also gains the highest effectiveness by combining the two complementary flows.

CPN has been widely used for verification of BPEL composition. Many deals with converting the BPEL constructs into CPN model. A data driven approach to compose the web services using CPNs is given in [21]. When a new business requirement given with input/output details needs to be implemented, the existing service portfolio is checked for reusability. The data relations in business and service domain are utilized to create a complete and coherent data model. A service net is created with all the possible composition candidates from given service portfolio. Then it is reduced with respect to the given business requirement. In [22] a method to create a mapping between WS-BPEL process and CPN model is proposed. The web service composition is validated by analyzing its reachability tree. CPN [23] provides an effective means to simulate, analyze and verify the correctness of web service composition. In this paper the input to the test suite generation algorithm is the data driven model that considers both business and implementation domains.

3.1 CPN Model

In this paper the CPN model for the composite web service (Service-Net) is defined as a tuple < Σ, I, P, T, A, C, O >, where

- Σ is a non-empty color set and represents the data types of the participating web services;
- I is a set of input places derived from the business requirement and represents the initial input to the composite web service;
- P is a limited set of place, P is inclusive of I and O, and represents the state of atomic web service;
- T is a limited set of transition, and represents the operation of atomic web service;
- A is a finite set of arcs.
- C is a color function defined from P into Σ. C is injective, i.e., C (p1) = C (p2) Σ p1 = p2.
- O is a set of output place derived from the business requirement and represents the final output of the composite web service

3.2 Case Study

A composite service implements a business process which accomplishes a specific organizational goal by using a coordinated set of tasks performed by humans or software. We take a simple case study; 'Plan for Travel' process which consists of three web services: a Traveller, a Travel Agent and an Airline Reservation System. A person, who wants to travel via air, first proposes an itinerary and orders for the trip. He can change or cancel the itinerary. He reserves, books and then receives the ticket. Web service Traveller helps the person get the tickets for the proposed itinerary. The Travel Agent web service receives order, checks availability of seats, reserves and books the seat, and sends statement to the person. It also acts upon the timeout scenarios, change and cancellation requests. The Web service 'Airline Reservation System' verifies the seat availability, books the tickets and sends to the person. It also handles the cancellation requests. This case study has been used in both WSCI and BPEL4WS composition languages [22]. The figure 1 gives the composition model of the three web services and this model is used for creating test sequences.

4. Algorithm

The unique input output pairs (UIO-Pairs) and the CPN model is the input to the algorithm. The business requirements are represented by business processes consisting abstract activities with input/output data. The input and output data are mapped to form UIO-Pairs. The UIO-Pairs at the highest level for the ticket booking business process would be, input: proposed itinerary and outputs: booking succeeded or failed. The unique pairs can be easily derived from the pre conditions and post conditions specified in the business requirements. Moreover the pairs would be a limited and finite set. Hence the time taken to create the UIO-Pairs would be minimal. Table 1 gives the exhaustive set for the case study taken into consideration.

The CPN model is generated as a service net using the WSDL in the service portfolio and the business requirements. The approach to build the service nets is presented in [21]. Well-designed test data helps identify critical flaws in the functionality. The test data for each step in the sequence can be generated from the WSDL documents that are used to

create the CPN model. The valid input space of a web service is the subset of the input space satisfying the precondition of the web service. The proposed algorithm takes the most influencing pair first and starts creating test sequences. The table of unique input and output can also be used to roughly estimate the test suite size. The number of test sequences is directly related to the UIO-Pairs. However if there are conditional branches based on computation and not on user inputs, then the number test sequences depends on the number of branches too. The test sequences can also be prioritised by prioritising the UIO-Pairs. The test suite size can be reduced by generating sequences for the UIO-Pairs that are business critical. In table 1 the first row is the most critical scenario where the end-user of the system is satisfied by receiving the tickets for the proposed itinerary. The fourth row is pertaining to reservation cancellation due to over time. Such situations are very rare and hence the priority for this row is low. Therefore the test sequence generated using that UIO-Pair is also low. Definition of a test sequence and the design coverage evaluation relates to the generation technique used. Thus, it is important to define the coverage criteria and concepts followed in this paper. The test data is a set of inputs that would be used in the test step during execution. A test sequence is a set of test steps that trigger a sequence of tasks or operations to accomplish a logical flow of events in the business process. The focus of the proposed technique is to validate the behavioral correctness of the system using the generated test sequences. The test sequence will validate the functional correctness and dependencies of the operations. A test suite is a collection of test sequences.

Figure 2 gives the test sequence generation process for the web service composition. The proposed algorithm will result in high coverage with minimal effort. The algorithm is analysed using the following coverage criteria.

4.1 Complete Sequence Coverage (CSCov)

The meaningful sequences obtained by traversing the CPN model for all the UIO-Pairs and all the decision points. In the CPN model the decision point is represented by a place with multiple out-going arcs.

$$CSCov = \frac{\text{No. of UIO-Pairs and decision points used}}{\text{Total No. of UIO-Pairs and decision points}} \quad \text{--- (1)}$$

4.2 Decision Coverage (DCov):

The test suite TS fulfils 100% decision coverage if there is at least one test sequence of every decision point in the CPN model.

$$DCov = \frac{\text{No. of decision points used in sequence}}{\text{Total No. of decision points}} \quad \text{--- (2)}$$

4.3 Input Output Coverage (IOCov):

The test suite TS satisfies Input Output coverage if there is at least one test sequence of every unique input output pair of the business requirement/process.

$$IOCov = \frac{\text{No. of UIO-Pairs used in sequence}}{\text{Total No. of UIO-Pairs}} \quad \text{--- (3)}$$

4.4 Transition Coverage (TCov):

The test suite TS satisfies this coverage if there test sequences such that every transition is traversed at once in any of the sequences.

$$TCov = \frac{\text{No. of transitions used in the test sequence}}{\text{Total No. of transitions in the CPN Model}} \quad \text{--- (4)}$$

Algorithm CTS-G (CPN based Test Sequence Generation)

Begin
  Initialize the CPN Model
  For each UIO-Pairs
    Let IP= input set of top UIO-Pair


```
        Let OP= output set of top UIO-Pair
        /*Reduce redundant traversal to decision point*/
        For each DP
          If (IP part of DP) Then
                Copy test steps till DP
                T = Transitions after DP
             Exit For
          End If
        Next For
        Enable T that satisfy IP and pre-conditions
        While (P Not OP) Do
          /*Places will specify the inputs need for firing a transition*/
          Choose an enabled transition (T) that influence OP
          Fire (T)
          /*Create test step for the test sequence relevant to UIO-Pair*/
          Record the Places connected to T as input
          Record traversal in the test sequence
          Record resulting Places and Arc expressions as output
          If (T is web service operation) Then
                /*Automatically update traceability matrix (TM) */
            Update TM
          End If
          Analyse resulting place P
          If (P has multiple arcs) Then
            /*Decision points are Place with multiple arcs */
            Save decision points; DP=DP+P
            If (this is first decision point) Then
              Path_Start = Initial Transition
            Else
              Path_Start = Previous DP
            End If
            Path_End = Current DP
            Save test steps to Place along with pre-conditions
          End If
        End Do
        Remove UIO-Pair
        Calculate TCov, DCov
        Update test sequence with post-conditions and coverage
        Initialize CPN to decision point that is yet to be covered
      Next For
End
```

In the CPN model for web service composition, a test sequence is any path from one of the initial state to one of the final state. In the case study taken up the 'Change Itinerary' flow takes up input as Proposed Itinerary but the output depends on the 'Check Seat Availability' operation. In this paper external input and output pairs are focused and the sequences that are missed are merged when the transition coverage is calculated. In the algorithm each Place and Transition considered is the data type of web service operation and web service operation respectively. Intermediate transitions and places are not recorded as part of test sequences. The test sequences generated can be used for black box testing of web service composition. Moreover the traceability is created to business requirements and the web services. To best of our knowledge traceability has not be considered in existing approaches. Moreover the gap between business domain and implementation domain is bridged in this approach.

5. Results and discussions

In traditional MBT, the models are created using requirements or specifications only. However, the existing model-based test case generation approaches generate models from executable code: the BPEL code. Hence the model created reflects the behaviour of the executable code rather than the expected behaviour of the system. Using such a model for test case generation will lead to validation of BPEL code and not business requirements which specifies the actual system. In the proposed approach the CPN model used is a data driven model that bridges the gap between the service and business domain. Such a CPN model is already validated for accuracy of the system's functional design. Generating test cases for such a model is bound to produce effective and efficient test cases. The algorithm also inherits another default benefit by using CPN model for deriving test sequences. The CPN tools have been exhaustively used in verification of web service compositions; be it choreography or orchestration. BPEL code constructs and WSCI code constructs have been transformed to CPN models to check reachability and soundness of the composition. Table 2 presents an analysis by comparing the approaches that exist for automatic generation of test sequences from the CPN Model. Traceability refers to traceability of the test sequences. Usage refers to the domain or the applications that uses test sequences generated. In future test suite length and the algorithm complexity will also be analyzed. One of the other advantages of MBT is that the test sequences generated also aims at providing maximum testing coverage of the system under test. Generally the coverage based adequacy metric usually relates to the model. The data driven CPN model is created from the business requirements and WSDLs, hence the requirement coverage and web service coverage are inherited by default if the whole model is covered. Table 3 represents the test sequence for Reservation time out scenario. The traveller enters the proposed itinerary and his personal details to order tickets for the trip. The airline reservation system checks and verifies the seat availability for the itinerary received from the travel agent. The reserved seat details will be sent to traveller or there might happen a time out situation where the system enters the failure state after notifying that time is the reason for failure. Here the UIO-Pair is row 3 from Table 1.

6. Conclusion

Testing is the most critical and expensive phase of the software development life cycle. Generation of test sequences or cases is most challenging part of testing phase as an efficient test design can detect greater number of faults. Moreover around 40% of software testing cost is spent on test design. In this paper we have present an approach to reduce that cost by means of automating the generation of test sequences for web service composition. We first analysed the existing approaches to generate test cases from the CPN model. Then we analysed the usage of CPN in web service composition. Then finally we consider the data driven CPN model used for verifying the design of composition and UIO-Pairs created from the business process requirements as input to create test sequences. As opportunities for future work, on-the-fly test sequence prioritization and automatic tracing back to business requirements can be taken up. Moreover the decision points which are places that have multiple arcs can be used to reduce redundancy in test sequences.

References

[1] Thomas Erl (2005), Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall PTR

[2] Lizhi Cai, Juan Zhang, Zhenyu Liu, (2011), A CPN-based Software Testing Approach, JOURNAL OF SOFTWARE, VOL. 6, NO. 3, pp.468-474

[3] Jing LIU, Xinming YE, Jun LI, (2011), Colored Petri Nets Model based Conformance Test Generation, IEEE Xplorer, pp: 967-970.

[4] U. Farooq, C.P. Lam and H. Li, (2008), Towards Automated Test Sequence Generation. 19th Australian Conference on Software Engineering, IEEE Computer Society.

[5] H. Watanabe and T. Kudoh. (1995), Test Suite Generation Methods for Concurrent Systems based on Coloured Petri Nets.2nd Asia-Pacific Software Engineering Conference (APSEC 1995), Brisbane, Australia, pp. 242-251.

[6] Mustafa Bozkurt, Mark Harman and Youssef Hassoun, (2009), Testing & Verification In Service-Oriented Architecture: A Survey, SOFTWARE TESTING, VERIFICATION AND RELIABILITY, Wiley InterScience, pp.1-67.

[7] Jose Garcia-Fanjul, Javier Tuya, Claudio de la Riva, (2006), Generating Test Cases Specifications for BPEL Compositions of Web Services Using SPIN, International Workshop on Web Services Modeling and Testing, pp. 83-94.

[8] Y. Zheng, J. Zhou, P. Krause, (2007), A Model Checking based Test Case Generation Framework for Web Services, (2007), International Conference on Information Technology.

[9] Y. Zheng, P. Krause, Automata Semantics and Analysis of BPEL, International Conference on Digital Ecosystems and technologies.

[10] X. Fu T. Bultan J. Su,(2004), Analysis of Interacting BPEL Web Services, International Conference on World Wide Web. May 17 - 22New York, USA.

[11] M. Lallali, F. Zaidi, A. Cavalli, (2008), Transforming BPEL into Intermediate Format Language for Web Services Composition Testing, The 4th IEEE International Conference on Next Generation Web Services Practices.

[12] M. Lallali, F. Zaidi, A. Cavalli, Iksoon Hwang, (2008), Automatic Timed Test Case Generation for Web Services Composition, Sixth European Conference on Web Services. Dublin, Ireland, Nov 12 - 14.

[13] Tien-Dung Cao, Patrick Felix, Richard Castanet and Ismail Berrada, (2009), Testing Web Services Composition using the TGSE Tool, 2009 Congress on Service-I, IEEE Computer Society, pp. 187-194

[14] A. T. Endo, A. S. Sim˜ao, S. R. S. Souza, and P. S. L. Souza, (2008), Web services composition testing: A strategy based on structural testing of parallel programs, TAIC-PART ’08: Proceedings of the Testing: Academic & Industrial Conference - Practice and Research Techniques. Windsor, UK: IEEE Computer Society, pp. 3–12.

[15] J. Yan, Z. Li, Y. Yuan, W. Sun, and J. Zhang, (2006), BPEL4WS unit testing: Test case generation using a concurrent path analysis approach, ISSRE ’06: Proceedings of the 17th International Symposium on Software Reliability Engineering. Raleigh, NC, USA: IEEE Computer Society, pp. 75–84.

[16] Y. Yuan, Z. Li, and W. Sun, (2006), A graph-search based approach to BPEL4WS test generation, ICSEA ’06: Proceedings of the International Conference on Software Engineering Advances. Tahiti, French Polynesia: IEEE Computer Society, p. 14.

[17] S. S. Hou, L. Zhang, Q. Lan, H. Mei, and J. S. Sun, (2009), Generating e_ective test sequences for BPEL testing, QSIC 2009: Proceedings of the 9th International Conference on Quality Software. Jeju, Korea: IEEE Computer Society Press.

[18] Z. Guangquan, R. Mei, and Z. Jun, (2007), A business process of web services testing method based on uml2.0 activity diagram, IITA’07: Proceedings of the Workshop on Intelligent Information Technology Application. Nanchang, China: IEEE Computer Society, pp. 59–65.

[19] A. Tarhini, H. Fouchal, and N. Mansour, (2006), Regression testing web services-based applications, Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications. Sharjah, UAE: IEEE Computer Society, pp. 163–170.

[20] A. Tarhini, H. Fouchal, and N. Mansour, (2005), A simple approach for testing web service based applications, Proceedings of the 5th International Workshop on Innovative Internet Community Systems (IICS 2005), ser. Lecture Notes in Computer Science, vol. 3908. Paris, France: Springer, pp. 134–146.

[21] Wei Tan, Yushun Fan, MengChu Zhou and Zhong Tian, (2010), Data-Driven Service Composition in Enterprise SOA Solutions: A Petri Net Approach, IEEE TRANSACTIONS, pp. 686-695

[22] Xinguo Deng, Ziyu Lin, Weiqing Cheng, Ruliang Xiao,Ling Li,Lina Fang, (2007), Modeling and Verifying Web Service Composition Using Colored Petri Nets Based On WSCI, IEEE, pp:1863-1867

Poonkavithai Kalamegam received B.Tech degree in Computer Science and Engineering from Pondicherry University and M.E. degree in Computer Science from Anna University. Her research interest includes Service Oriented Architecture, Model Based Testing, and Web Service Composition Testing. She has around 10 years of industry experience in functional testing of banking domain applications. She has been involved in all phases of testing, starting from estimation for testing phase to closure with test summary report. She has worked in Cognizant Technology Solutions for 6 years. JP Morgan Chase Bank, Dutsche Bank and Boeing Financials are some of the clients she has worked for. She is currently pursuing PhD degree in the area Web Service Composition Testing in Pondicherry Engineering College.

Dr. G. Zayaraz is currently working as Associate Professor in Computer Science & Engineering at Pondicherry Engineering College, Puducherry, India. He received his Bachelor`s, Masters and Doctorate degree in Computer Science & Engineering from Pondicherry University. He has published more than Thirty five research papers in reputed International Journals and Conferences. His areas of specialization include Software Architecture and Information Security. He is a reviewer/editorial member for several reputed International Journals and Conferences and Life Member of CSE, and ISTE.

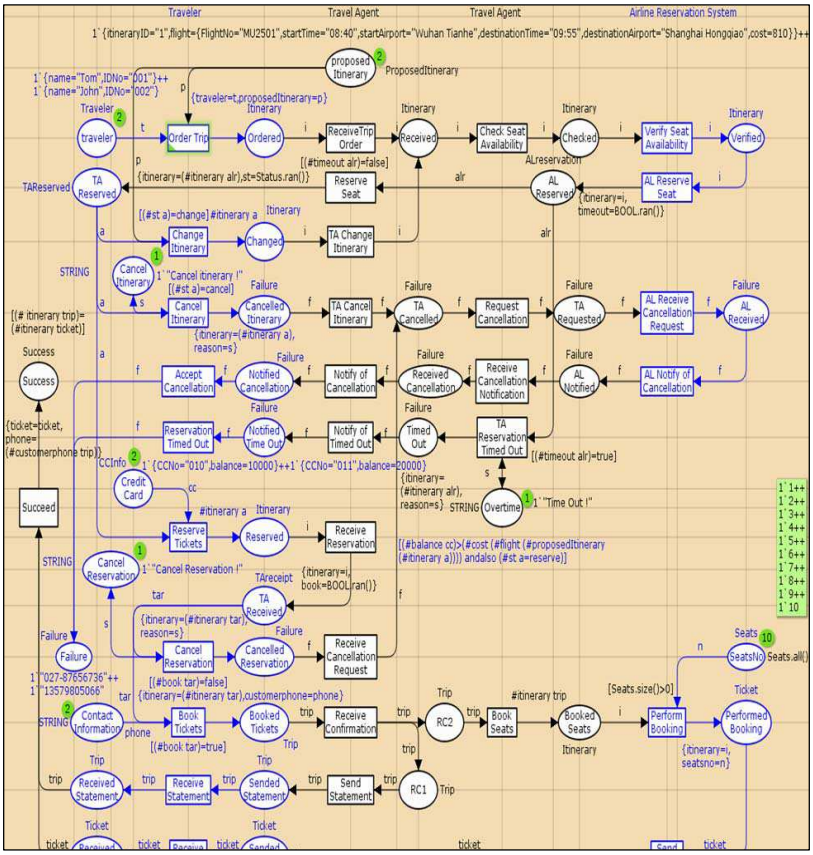

Figure 1. CPN Model for web service composition

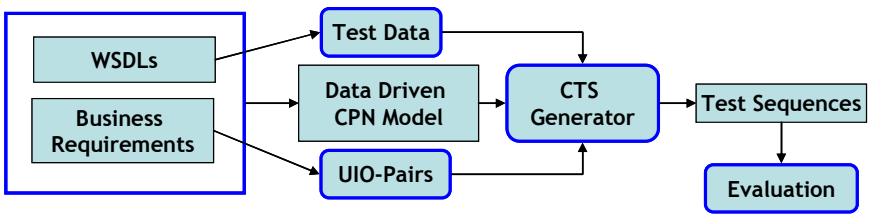

Figure 2. Test Suite generation process

Table 1. UIO-Pairs

SNo	Input	Output
1	Traveller details Proposed Itinerary	Success (Tickets Booked)
2	Cancel Itinerary Reserved Seat Details	Failure notification (Cancel Itinerary)
3	Cancel Reservation Reserved Itinerary	Failure notification (Cancel Reservation)
4	AirLine Reservation Failure OverTime	Failure notification (Timeout nofication)

Table 2. The comparison between some existing approaches of CPN to Test Sequence generation

Reference	Traceability	Usage	Redundancy	Test Priority
Lizhi Cai,J Zhang, Zhenyu Liu[2]	To model	Generic	Not Handled	Not Handled
Jing LIU, Xinming YE, Jun LI[3]	To model	GUI Apps	Not Handled	Not Handled
U. Farooq, C.P. Lam and H. Li[4]	To AD and model	SOA	Limited	Not Handled
H. Watanabe and T. Kudoh[5]	To model	Concurrency	Not Handled	Not Handled
Our Work	To Bus. Req and Web Services	SOA	Handled	Handled