A Test Framework based on CPN Model for Functional Testing of Web Service Composition

G. Zayaraz[1] and Poonkavithai Kalamegam[2]

[1]Associate Professor, [2]Research Scholar
Department of Computer Science and Engineering
Pondicherry Engineering College,
Pondicherry, India

Abstract

Web service composition is a 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. 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. Colored Petri Nets (CPNs) is one model that combines data and control flow with behavioural aspects of a given business requirement. In this paper, a test framework based on a data-driven CPN model for functional testing of web service composition is proposed. The test framework also includes test design coverage measurement using three newly formulated metrics: Decision Coverage, Input Output Coverage and Implementation Coverage. The framework was applied to airline reservation system and the generated test sequences were evaluated against the coverage criteria.

Keywords: *CPN, MBT, web service composition testing, Test framework, test case generation*

1. Introduction

SOA (Service-Oriented Architecture) is a design pattern composed of loosely coupled, discoverable, reusable, inter-operable platform agnostic services that follow a well defined standard [1]. Service Composability is the most important and fundamental design principle of SOA. Web service is an encapsulated and self-contained functionality; that often represent the foundation stones of larger applications. A business process in SOA is a composition of service invocations that follow a certain set of execution rules to handle transformations of data, dependencies, and correlations to accomplish the business process. Such a procedure is called Web Service composition. As the organization grows, the complexity of business application software that are built using web services will increase. To yield the benefits of SOA, Web services and their compositions need to behave correctly. The correctness of any implementation is accessed by executing a test suite with a significant number of test cases. As the complexity and size of the web service compositions increases, the complexity of testing and the types of testing increases. Black-box testing, white-box testing and gray-box testing are the three main types of tests which are considered when designing test cases. In SOA only the interface of the implemented service is known and only the output of the service under test can be observed, hence black-box testing technique is best suited. The tests are also categorized into different levels: Unit, Integration, System, Acceptance and Regression testing. As a consequence, several test strategies have been proposed in the past

for every level of testing [2]. In this paper, the major focus would be on creating a test framework for function testing of web service composition.

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. UML (Unified Modeling Language), FSM (finite state machines) and Colored Petri Nets (CPN) are widely used modelling mechanisms to specify, analyse and simulate requirements of the software, for test automation and for model-based software testing. CPN have been used to formalize service compositions invariable of the composition style. Moreover, CPN offers a large formal background that facilitates its usage to model service compositions and generate test cases. CPN is one model that combines data and control flow with behavioural aspects the requirements. The analysis of transitions fired will lead to control flow coverage and the data flow coverage is defined in terms of tokens used. 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, Model Based Testing (MBT) technique is applied on the CPN model to generate test sequences and a test framework for functional testing of web service compositions is proposed. The paper is organized as follows. In next section, the existing work that motivated the proposed framework is discussed. In Section 3, the test framework for functional testing of web service composition is presented and in Section 4 the case study and the usage of the test framework are presented. In Section 5 presents the experimental results, followed by conclusions in the last section.

2. Motivation

The proposed test framework for web service composition is based on CPN model. The motivation for the proposal is derived from automatic generation of test sequences from the CPN model and few limitations in the existing test framework for web service composition testing.

The CPN model provides a formal description of the web composition a system. There are several approaches to verify the design of web service composition using CPN model. 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. 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 [3]. 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 [4]. U. Farooq, C. P. Lam and H. Lin [5] 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 [6] 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. To the best of our knowledge CPN models are not used efficiently in test case generation for integration testing of service oriented solutions.

There are several frameworks proposed for web service composition testing. A unit testing framework [7] was created for testing the BPEL compositions. The framework comprises four components. The Web Services Business Process Execution Language (WS-BPEL) Processes under testing, it's Partner Processes and a test architecture that simulates the Partner processes using a Control Process that start and end the tests. The fourth component is a User Interface (UI) that starts-up or stops Control Process and Test Processes. An abstract workflow-based framework was proposed by Karam [8] to test web service composition. Test cases are generated from open nets in [9]. The Timed Extended Finite State Machines (TEFSM) is used widely due to the available tool support. The BPEL specification is transformed into the TEFSM model and then used for test case generation. Based on Time Extended Finite State Machine Cavalli [10] developed a testing framework that can be used in all phases of Web services composition. An approach to specify, model, verify and validate web services choreography has been presented in [11]. Finite State Processes (FSP) is used along with the Labeled Transition System Analyzer (LTSA) tool to produce and analyze Labeled Transition Systems (LTS). The existing approaches use the BPEL code for generating the model and that model is used for test case generation. The key feature of the test framework proposed in this paper, is that the test sequences are generated from CPN model that is a formal specification of the system, rather than being abstracted from the implementation domain/code.

The existing frameworks use BPEL code for creating the test models that are used to generate the test cases. The rich information on data available in the WSDL schema is not utilized effectively. Moreover the test models are not used to verify the correctness of web service composition design. 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. The drawbacks of the existing frameworks are addressed by the following features of the proposed test framework:

(i) Generate test sequences for web service composition using the same CPN model used for verifying design

(ii) Link the business requirements to the CPN model and the test sequences

(iii) Make the best use the WSDL schema for test data generation

(iv) Build an effective and efficient test framework to achieve highest test design coverage in lesser test suite size.

In this paper, an algorithm to generate of test sequences from the CPN model is presented. Then a test framework is setup for functional testing of web service compositions.

3. Proposed Test Framework for Web Service Compositions

The steps involved in the Test Framework are:

- CPN Model Initialization
- Unique Input Output Pair generation
- Prioritization
- Test Data generation
- CTS-G algorithm for Automated Test Sequence Generation

- Traceability
- Test Coverage measurement

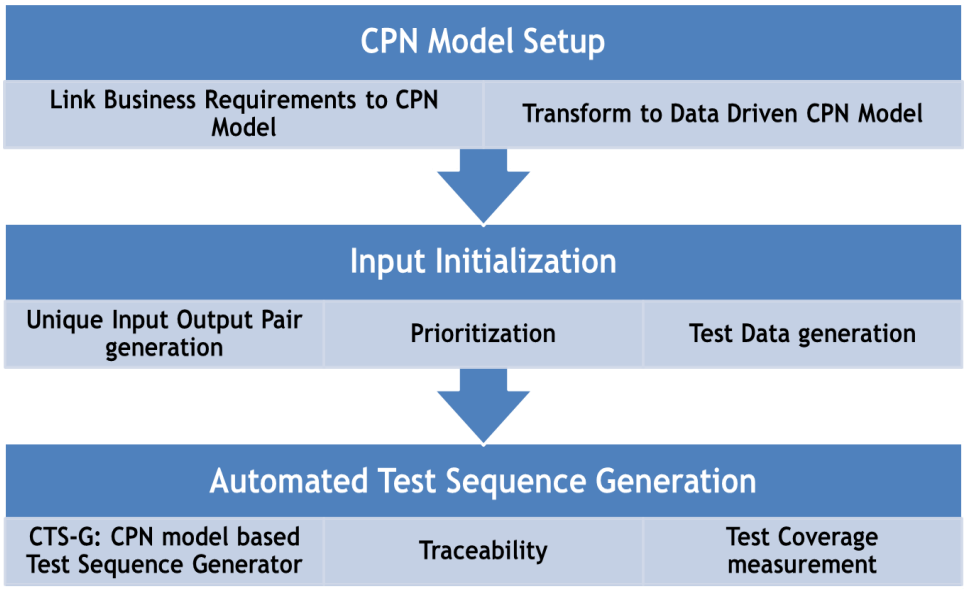

Figure 1. Test framework for functional testing of web service compositions

The aforementioned steps are used as shown in Figure 1. The test framework generates the test sequences for functional testing of web service compositions from the CPN model. The test sequences generated by the CTS-G algorithm can be converted into XML request files and executed using the SOAPUI tool. In the following sub sections each step of the test framework for web service composition is elaborated.

3.1. CPN Model Setup

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. The web service composition has to be validated for both Control Flow and Data Flow aspects. 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. A data driven approach to compose the web services using CPNs is given in [12]. 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 [13] 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.

In this paper, the data-driven CPN model that considers both business and implementation domains is used for generating the test sequences. In the CPN model Setup process of the test

framework two activities are done (i) the CPN model is updated manually such that the model links all the business requirements (ii) if the CPN model is generated for BPEL code then it is updated with functional and data abstraction using the business requirement specifications and WSDLs. The web service composition is modeled and analyzed for the accuracy of the functional design using the CPN. 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. Thus same CPN model is used for generation of test sequences and for verifying design correctness for web service composition.

3.2. Input Initialization

After setting up the CPN model for web service composition, the next process in the test framework is to gather the input that enable automatic generation of test sequences. The results of the following sub tasks in this process are tabulated in the 'Experimental Results' section.

3.2.1. Unique Input Output Pair generation

The UIO-Pairs are created from the business domain. Business domain is also known as the requirement specification domain, in which requirements are represented by business processes consisting of 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. 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 of test sequences depends on the number of branches too.

3.2.2. Prioritization

The test sequences can also be prioritized by prioritizing the UIO-Pairs. The test suite size can be abridged by generating sequences for the UIO-Pairs that are most business critical. In the case of ticket booking business process the most critical scenario is the end-user receiving the tickets for the proposed itinerary. The scenario such as reservation cancellation due to over time that happens very rarely has the lowest priority. Therefore the test sequence generated using that UIO-Pair is also low. In some cases the input is observable but the output triggers an action that eventually merges into a decision path. Such UIO-Pairs can be merged with one of the other test sequences that take similar input. Moreover the UIO-Pairs are sorted in such a way that the special cases are prioritized after the normal pairs. This enables effective merging of the test steps for the special cases into the already available test sequences according to priority.

3.2.3. Test Data generation

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 test data is a set of inputs that would be used in the test step during execution. Test data for system testing is of two types; (i) the data that initiate the system under test and (ii) test data that change the system to a particular state for some test cases to start execution. Every Transition in the CPN model is considered as an operation of a web service and the in-ward Places to that transition are the input parameters for that operation. Thus the test data applied to the test sequences are derived from the business requirements and the WSDL documents.

3.3. Automated Test Sequence Generation

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. A collection of test sequences constitutes a test suite. The proposed framework aims at validating the behavioral correctness of the system using the automatically generated test sequences and achieving 100% test coverage in lesser test suite size.

3.3.1. CTS-G: CPN model based Test Sequence Generator

The test framework utilizes the CTS-G algorithm proposed by the authors in [14] to generate test sequences. The data-driven CPN model is used to generate Test Sequences for validating the web service composition. The proposed algorithm takes the most influencing pair first and starts creating test sequences. The test data for the test sequences are derived from the WSDLs are combined together to generate XML test cases. The XML test cases can be used in the SOAP UI tool for validating the web service composition. The below are the key steps followed to generate test sequences.

Algorithm – CTS-G

[1] Initialize the CPN model

[2] For every UIO-Pair parse the CPN model do steps 3 to 8

[3] Start traversal of CPN model from transition that satisfies the input set

[4] If input set already used → Reuse the already existing path to the Decision Points (DP)

[5] If a DP is encountered during the traversal → Record the common path to the DP along with the pre-conditions

[6] Update the traceability matrix to calculate the Implementation Coverage

[7] Calculate transition and decision coverage of each test sequence

[8] Merge the test sequence for unobservable states with already existing sequences

[9] Repeat from step [2] until all UIO-pairs has test sequences

In the CPN model the Decision Point is represented by a Place with multiple out-going arcs. Any Decision Point has more than one possible path. However the path to the Decision Point is common to all the branching out paths. Hence saving the common path reduces redundancy. In the case of ticket booking business process to cancel reservation of ticket, the

itinerary must be proposed, seat be reserved and then cancelled. Proposal of itinerary is a common action for any test sequence. By reusing the already existing steps, the time is saved and redundancy is avoided. The algorithm is simple as it takes up only the unique input and output sequences. The test data for each step in the Test Sequence is generated from the WSDL documents that are used to create the CPN model. The proposed takes the most influencing pair first and starts creating test sequences. In this section a generic algorithm to generate test sequences for validation web service composition was presented.

3.3.2. Traceability

The requirements tracing is the process of documenting the links between the business requirements and the work products developed to implement and verify those requirements. All the work products, starting from the software requirements to project closure documents are to be linked properly. In testing phase a requirements traceability matrix document links the business requirements with the test cases created, to make sure that all the requirements are covered in test cases and hence all functionality is validated. The proposed framework automatically creates a bi-directional that contains both Forward and Backward Traceability. In forward traceability each test sequence is linked to business requirement and backward traceability ensures that every test sequence is linked to a business requirement. In the CPN model the transitions are linked to the requirements, to ensure bi-directional traceability between the requirements and the model. Thus all the test sequences generated from the model are linked to the business requirements. Moreover the test sequences are also linked to the implementation domain by linking the test sequences to the web services involved.

3.3.3. Test Coverage measurement

The proposed algorithm will result in high coverage with minimal effort. 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, Several CPN coverage criteria can be defined to determine the design coverage of test sequences generated. In Zhu and He [15] a number of coverage criteria have been introduced. A methodology of testing high-level Petri nets is proposed based on general theory of testing concurrent software systems. They classify the testing strategy as: transition-oriented, state oriented, flow-oriented and specification-oriented. For each strategy a set of coverage criteria to measure test adequacy are defined. In this paper, three new metrics appropriate to the test framework proposed is defined.

Implementation Coverage (IMCov)

The CPN model is generated using the WSDLs of the web services that participate in the web service composition. The implement coverage is defined as the ratio of web services covered by the test suite and total number of web services (WS) participating in the implementation of the web service composition.

$$IMCov_{Total} = \frac{No.\ of\ WS\ covered\ by\ test\ suite}{Total\ No.WS\ part\ of\ implementation}$$

$$IMCov_{TS} = \frac{No.\ of\ operations\ of\ a\ WS\ covered\ in\ test\ sequence\ TS}{Total\ No.WS\ operations\ in\ a\ web\ service}$$

$$IMCov_{Impact} = \frac{No.of\ test\ sequences\ that\ use\ operation\ x\ of\ y\ WS}{Test\ Suite\ Size}$$

The metric IMCov % is used to (i) analyze the impact of every web service on every test sequence (ii) identify test sequences that cover most of the web service operations and (iii) analyze the impact of changes in the web service operations on the test suite. The value of this metric is a direct measure of impact on the test suite when there is a change in the involved web services. In case of ticket booking business process, Proposed Itinerary operation has more impact than of Cancel Reservation operation. In this paper, the result of this metric is tabulated in the Experimental Results and Case Study sections.

Decision Coverage (DCov):

In CPN model a Place with multiple outputs corresponds to a Decision Point. For 100% Decision Point coverage, a test suite needs to have at least one test sequence for each condition of the decision.

$$DCov = \frac{No.of\ decision\ points\ used\ in\ the\ test\ sequence}{Total\ No.of\ decision\ points}$$

This metric is similar to branch coverage. This metrics gives an assurance that every Decision Point is visited at least once. Considering every output of a Decision Point will lead to an exploratory and huge test suite. This coverage can also be used during regression test suite selection. The test sequences that have 100% decision coverage are selected to ensure that the defect fixes has not affected the main functionalities of the system.

Input Output Coverage (IOCov):

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

$$IOCov = \frac{No.of\ UIO-Pairs\ used\ in\ the\ test\ suite}{Total\ No.of\ UIO-Pairs}$$

This metric is used to (i) analyze impact of the UIO-Pairs on the test suite and (ii) to help in regression test suite setup. For example, when a business requirement changes, the CTS-G algorithm can be re-run for UIO-Pairs that are linked to the changed business requirements alone. This saves lots of effort by avoiding test sequence generation for the whole system under test.

4. 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. A simple case study; 'Plan for Travel' process which consists of three web services: a Traveller, a Travel Agent and an Airline Reservation System is considered as system under test. 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.

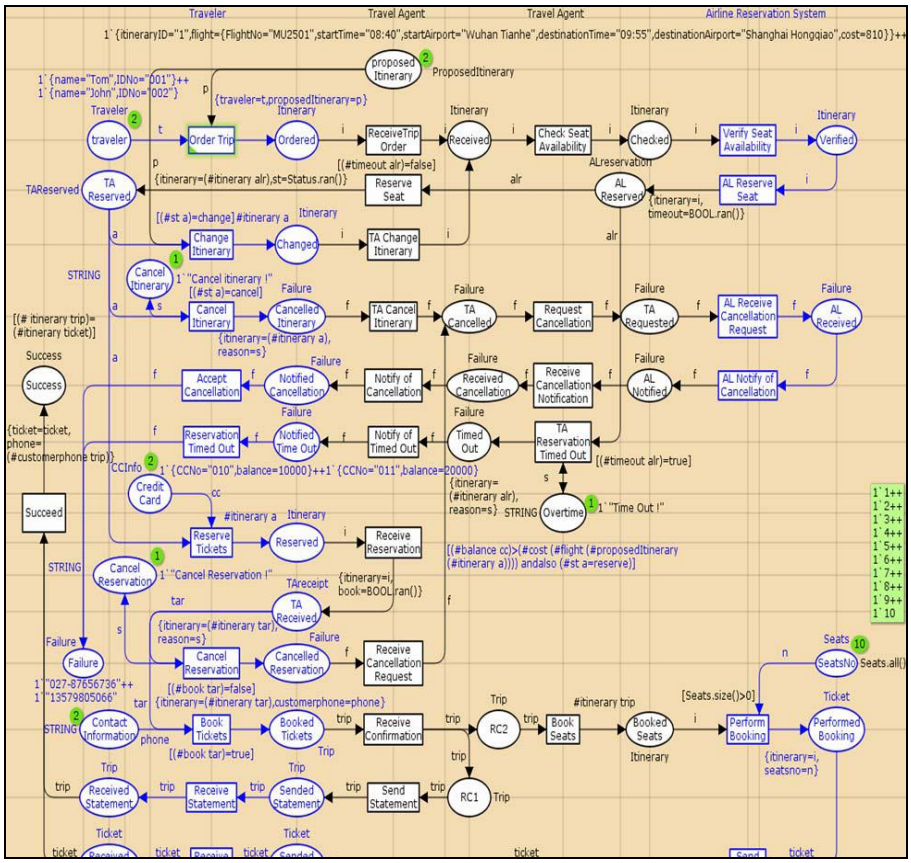

Figure 2. CPN Model for Airline reservation system

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 [12]. The Figure 2 gives the composition model of the three web services and this model is used for creating test sequences. The case study taken is a simple on hence the UIO-Pairs can be manually derived from the general description. The business process under consideration is ticket booking process, so the for the proposed itinerary there are only two outputs; one the tickets are booked and sent to traveller, two failure notice is sent to traveller. 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 and 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.

Table 1. UIO-Pairs

SNo	Input	Output
1	Proposed Itinerary Traveler, TA Reserved Credit Card No Contact No	Success (Tickets Booked)
2	Proposed Itinerary Traveler, TA Reserved Cancel Itinerary	Failure (Cancel Itinerary notification)
3	Proposed Itinerary Traveler, TA Reserved Credit Card No Cancel Reservation	Failure (Cancel Reservation notification)
4	Proposed Itinerary Traveler OverTime	Failure (Timeout notification)
5	Proposed Itinerary TA Reserved	Received (TA changed details)

In the CPN model the Decision Point is represented by a Place with multiple out-going arcs. Table 2 lists the Decision Points in the case study taken. Any Decision Point has more than one possible path. However the path to the Decision Point is common to all the branching out paths. Hence saving the common path reduces redundancy. For example the test sequence for row 3 of Table 1; for cancel reservation of ticket the seat must have done reservation in the airline and travel agent services, the steps for which is available in test sequence generated for row 1 of Table 1. By reusing the already existing steps, the time is saved and redundancy is avoided.

Table 2. Decision Points

Sno	Decision Point	Input	Next Transition
1	AL Reserved	Proposed Itinerary Traveler	Reserve Seat TA Reservation TO
2	TA Reserved	Proposed Itinerary Traveler	Change Itinerary Cancel Itinerary Reserve Tickets
3	TA Received	Proposed Itinerary Traveler Credit Card No	Cancel Reservation Book Tickets

Table 3. Implementation Coverage and Change Impact analysis for Traveller Service

Operation	Test Sequences				IMCov$_{Impact}$%
	Ticket Booked	Cancel Itinerary	Cancel Reservation	Reservation Timeout	
Order Trip	X	X	X	X	**100%**
Change Itinerary	X				**25%**
Cancel Itinerary		X			**25%**
Accept Cancel		X			**25%**
Reservation Time Out				X	**25%**
Reserve Tickets	X		X		**50%**
Cancel Reservation			X		**25%**
Book Tickets	X				**25%**
Receive Statement	X				**25%**
Receive Tickets	X				**25%**
IMCov$_{TS}$ %	**60%**	**30%**	**30%**	**20%**	

The Test Suite, Coverage Measurements and Traceability Matrix are the outputs of the proposed test framework. The Traceability Matrix is a document which links the business requirement ids with the Transitions in the CPN model and then the Test Sequence ids with the requirements via the Transitions. Here Table 3 gives the analysis of change impact of every operation in the Traveller web Service. There are ten operations in this service. The IMCov$_{TS}$% gives the measure of design coverage of each test sequence over the operations in the web services. During regression test case selection the test sequence with highest IMCov$_{TS}$% can be chosen instead of the entire test suite. The IMCov$_{Impact}$% is the direct measure of usage of a web service operation in the entire test suite. More the IMCov$_{Impact}$% value, higher will be the impact of changes in that web service operation. Moreover the coverage results illustrate that the prioritization of the UIO-Pairs are precise.

5. Experimental Results and Analysis

The Table 4 gives the test design coverage of the test sequences generated for the case study considered. The implementation and decision coverage of the test sequences generated are presented. In general business critical test sequences would have more Design Coverage. The business critical test sequences are linked to high priority UIO-Pairs. The maximum decision and implementation coverage is obtained from the first test sequence generated using the highest priority UIO-Pair. This again illustrates that the priority assigned to the UIO-Pairs are correct. In the case study taken into consideration the Transitions in the CPN model directly correlate to operations in the web services. Hence here 100% Implementation Coverage means that every Transition in the CPN model is part of the test suite. The proposed

test frame work gives a flexibility of stopping test design when optimal test coverage is achieved. Thus saving effort spend on test design and test execution.

Table 4. Test Design Coverage Analysis

Test Sequence	No. of Transitions		$IMCov_{total}$	DCov
	Unique	Re-Used		
Ticket Booked	19	4	63%	100%
Cancel Itinerary	6	8	20%	67%
Cancel Reservation	2	14	7%	100%
Reservation Timeout	3	5	10%	33%

Table 5 presents an analysis by comparing the approaches that exist for automatic generation of test sequences from the CPN Model. Here Traceability column refers to traceability of the test sequences and Usage column refers to the domain or the applications for which the test sequences are generated. Traceability is one of the key attributes every test suite should maintain. In the proposed approach every web service operation and every business requirement is linked to the appropriate test sequence in the test suite. Thus a complete bi-directional traceability between the business and implementation domain through the test suite is enabled. Moreover the test prioritization is done even before the test sequences are generated.

Table 5. The comparison between approaches for Test Sequence generation

Reference	Traceability	Usage	Redundancy	Test Priority
Cai et al [3]	To model	Generic	X	X
LIU et al [4]	To model	GUI	X	X
Farooq et al [5]	To AD and model	SOA	Limited	X
Watanabe et al[6]	To model	Concurrency	X	X
The proposed work	Business and implementation	SOA	Handled	Yes

The test framework provides a flexibility of generating test sequences for the select UIO-Pairs, thus saving time in design and execution phases of testing. Each test sequence generated by the proposed framework is unique in the aspect of both coverage and business logic. This enables better test management and test suite selection during regression testing. The redundancy in the test sequences are addressed by using the Decision Points in the CPN model. Farooq, *et al.*, [5] has proposed an approach similar to ours. The test suite length for the airline reservation system case study was analyzed. The test sequences generated by the proposed system provide the same measure of coverage with lesser test suite size. The TSG technique creates separate test sequences for non-observable outputs also. As the number of such outputs increase the test suite size increases. In the proposed test framework, such test sequences are merged to already existing test sequences, thus decreasing the test suite size.

The case study has only one non-observable output; hence the test suite size differs by 1. Figure 3 gives the comparison of the size of the test suite generated using the CTS-G algorithm of the proposed test framework and the random test sequence generator presented in [5]. Different business processes were modeled in CPN. Each model had different number of Transitions and number of unobservable outputs. In Case 4, the test suit size has a large difference due to presence of more number of unobservable outputs. As the test suite size is directly correlated to effort for test execution, the proposed framework saves test execution effort without any compromise on test design coverage. Moreover as the test suite size is lesser the maintenance cost and the rework on the test suite during changes in the business requirement is also lesser.

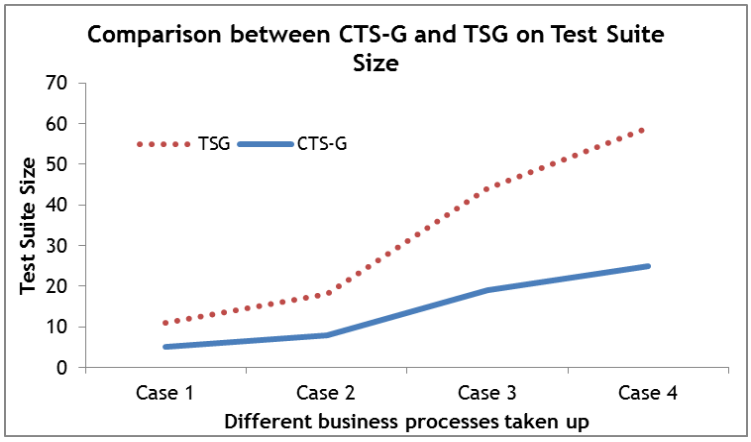

Figure 3. Comparison of proposed approach with TSG approach

6. Conclusion

Web service compositions have been widely adopted as a convenient mechanism to integrate heterogeneous systems and to model business workflows. 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. The test design phase is one the time consuming and hence costly phases in software testing life cycle. In this paper, a test framework to reduce that cost by means of automating the generation of test sequences for web service composition was presented. The existing approaches to generate test cases from the CPN model and frameworks available for web service composition were analyzed. Then a novel test framework for functional testing of web service composition was presented. A data driven CPN model and UIO-Pairs created from the business process requirements were used as input to create test sequences. The common path to the Decision Points was reused to reduced redundancy in test design. Moreover the test suite size is reduced by merging test sequences of unobservable outputs with already existing test sequences. The test suite size is directly proportional to the test execution time and hence the cost. The test framework proves to be cost effective saving effort in both test design and execution phases. Moreover efficient in terms of design coverage, achieving 100% test coverage in lesser test suite size. As opportunities for future work, the test framework would be extended to validate different layers of service oriented architecture.

References

[1] T. Erl, "Service-Oriented Architecture (SOA): Concepts, Technology, and Design", Prentice Hall PTR, **(2005)**.
[2] G. Canfora and M. Di Penta, "Service Oriented Architectures Testing: A Survey", LNCS 5413, Springer-Verlag Berlin Heidelberg, **(2009)**, pp. 78-105.
[3] L. Cai, J. Zhang and Z. Liu, "A CPN-based Software Testing Approach", Journal of Software, vol. 6, no. 3, **(2011)** March, pp. 468-474.
[4] J. Liu, X. Ye and J. Li, "Colored Petri Nets Model based Conformance Test Generation", IEEE xplorer, **(2011)**, pp. 967-970.
[5] U. Farooq, C.P. Lam and H. Li, "Towards Automated Test Sequence Generation", 19th Australian Conference on Software Engineering, IEEE Computer Society, **(2008)**.
[6] H. Watanabe and T. Kudoh, "Test Suite Generation Methods for Concurrent Systems based on Coloured Petri Nets", 2nd Asia-Pacific Software Engineering Conference (APSEC 1995), Brisbane, Australia, **(1995)**, pp. 242-251.
[7] Z. Li, W. Sun, Z. B. Jiang and X. Zhang, "BPEL4WS Unit testing: Framework and implementation", Proceedings of the 2005 IEEE International Conference on Web Services, ISBN: 0-7695-2409-5, **(2005)** July 11-15, pp. 103-110.
[8] M. Karam, H. Safa and H. Artail, "An abstract workflow-based framework for testing composed Web services", Proceedings of the 2007 IEEE/ACS International Conference on Computer Systems and Applications, ISBN: 1-4244-1030-4, Amman, **(2007)** May 13-16, pp. 901-908.
[9] K. Kaschner and N. Lohmann, "Automatic Test Case Generation for Services", In Monika Solanki, Barry Norton, and Stephan Reiff-Marganiec, editors, 3rd Young Researchers Workshop on Service Oriented Computing (YR-SOC 2008), **(2008)**.
[10] A. Cavalli, T. D. Cao, W. Mallouli, E. Martins, A. Sadovykh, S. Salva and F. Zaidi, "WebMov: A dedicated framework for the modeling and testing of Web Services composition", Proceedings of the 2010 IEEE International Conference on Web Services, ISBN: 978-1-4244-8146-0, Miami, **(2010)**, pp. 377-384.
[11] H. Foster, S. Uchitel, J. Magee and J. Kramer, "LTSA-WS: a tool for model-based verification of web service compositions and choreography", ICSE '06: Proceedings of the 28th international conference on Software engineering, New York, NY, USA, ACM, **(2006)**, pp. 771-774.
[12] W. Tan, Y. Fan, M. C. Zhou and Z. Tian, "Data-Driven Service Composition in Enterprise SOA Solutions: A Petri Net Approach", IEEE TRANSACTIONS, **(2010)**, pp. 686-695.
[13] X. Deng, Z. Lin, W. Cheng, R. Xiao, L. Li and L. Fang, "Modeling and Verifying Web Service Composition Using Colored Petri Nets Based On WSCI", IEEE, **(2007)**, pp. 1863-1867.
[14] P. Kalamegam and Z. Godandapani, "Test Sequences for Web Service Composition using CPN model", Computer Engineering and Intelligent Systems, vol. 3, no. 6, **(2012)**, pp. 32-41.
[15] H. Zhu and X. He, "A Methodology of Testing High-Level Petri Nets", Information and Software Technology, Elsevier, vol. 44, no. 8, **(2002)**, pp. 473-489.

Authors

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, Dutche 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.

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.