iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** III **Month of publication:** March 2018

DOI: http://doi.org/10.22214/ijraset.2018.3542

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue III, March 2018- Available at www.ijraset.com

Formal Web Services Description Language based Test Cases Creation for Testing Web Services

Dr. Sunil Kumar [1], Vikash Somani[2],
[1]*Assistant Professor, Dept. of Computer Science, Sangam University, Bhilwara, Rajasthan*
[2]*Assistant Professor, Dept. of Computer Science, Sangam University, Bhilwara, Rajasthan*

Abstract*: *In the most recent years a few tools automating WS testing have been introduced. However, generally the choice of which and how many test cases have to be run, and the instantiation of the input data into every test case, is still left to the individual tester. To guarantee the nature of the services that are distributed, bound, summoned and coordinated at runtime, test cases must be naturally produced and testing executed, observed and examined at runtime. This paper introduces the inspection to create Web Services test cases naturally or automatically in view of the WSDL (Web Services Description Language), which conveys the essential data of a service together with its interface procedures and the data transmitted.

In this paper we explain a proposition with computerize WSDL-based testing, which consolidates the scope of WS operations with data driven test case. The WSDL record is first parsed and changed into the organized DOM tree. At that point, test cases are produced from two viewpoints: test data generation and test task generation. At last, the created test cases are archived in XML-based test documents called Service Test Specification.

Keywords: WSDL, WSDL-based testing, Service Test Specification, Testing Web Services, test cases generation

I. INTRODUCTION

Web Services (WS) is the empowering strategy for Service-Oriented Computing (SOC), which gives a standard-based instrument and open stage for incorporating distributed autonomous service parts .The nature of services is a key issue for creating service-based software systems, and testing is vital for assessing the practical correctness, execution and reliability of individual plus composite services[1][2]. However, the dynamic highlights of WS require various new difficulties to conventional testing strategies. To start with, services are distributed, bound, summoned and incorporated at runtime. To incorporate into the framework, testing should be automated including automatic test case generation, test execution accumulation and analysis. Second, WS proposes a completely specification-based process utilizing a set of XML-based standards, e.g. SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) [8] [9], UDDI (Universal Description, Discovery and Integration) and WSFL (Web Services Flow Language).

Up-to-now, a large portion of the research of WS testing is still hypothetically based on formal strategies, for example, model checking. For illustrations, Shin proposes a model-checking procedure to confirm service flows utilizing automation-based model checker SPIN [4]. Howard et.al propose FSP (Finite State Process) [2] notation to model and check service creation. X. Yi and K. J. Kochut propose a CP-nets model, an expanded Petri Net model, for determining and confirming web service arrangement [9]. Tsai et al proposed to produce test cases based on OWL-S design during the fulfilment and consistency investigation and model checking[6][7][8]. WS testing tools likewise start to rise lately. These tools encourage monitoring and debugging of WS execution, for example, follow the process, debug WSDL and SOAP messages, catch and replay the requests and responses, and check the punctuation and semantic rightness of the details.

This paper proposes a procedure for creating test cases automatically in view of WSDL specification. The WSDL-based test case generation is a piece of the distributed service testing system that incorporates test case generation, test controller, test specialists and test evaluator. The specifications of the services are first parsed into a DOM tree portrayal in view of the WSDL schema. Test cases are created from four levels: test data generation, singular test activity generation, operation flow generation, and test specification generation. Test data are produced by examining the messages in light of XML schema data type definition including basic data type, complex data type, aggregated data type and client determined data type.

Tests for individual operations are produced by examining the related parameters (i.e. the messages). A service may include numerous operations. To test the combinational capacities, tests are created as sequences of operations. The sequences are produced based on task dependence investigation including input dependency, output dependency and input/output dependency. The created test cases are recorded into a XML-based report call Service Test Specification (STS). STS takes the fundamental components from WSDL together with operations, input, output, message and element.

©IJRASET (UGC Approved Journal): All Rights are Reserved

The rest of this manuscript is organized as follows. Section II presents the general architecture of conveyed WS testing framework. Section III delineates the system of WSDL-based test generation, including test information generation, operation flow generation and test requirement. Section IV talks about a contextual analysis and Section V outlines the paper and reaches the conclusions.

II. THE OVERALL ARCHITECTURE

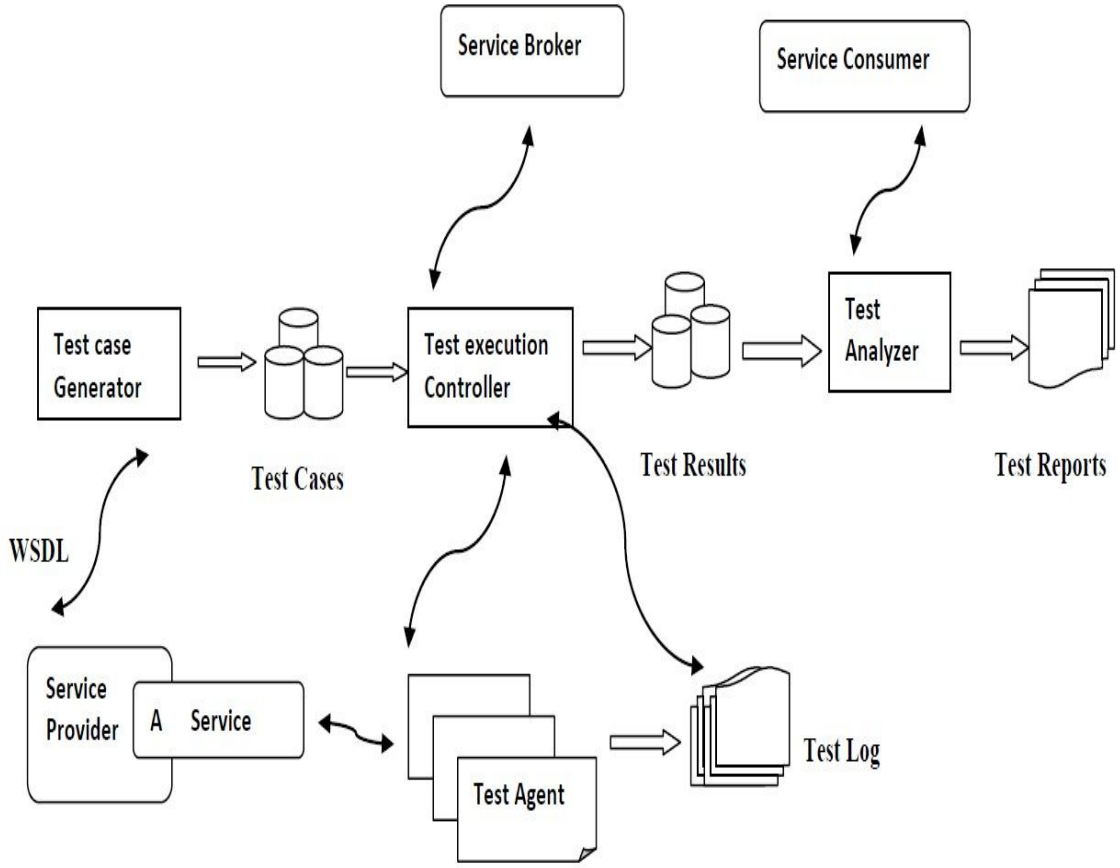


Figure 1: Overall architecture of the distributed service testing framework

A. Figure 1 demonstrates the general architecture of web service testing framework

1) Test Case Generator acknowledges the service accommodation in WSDL and programmed creates test cases. The produced tests will be put away in the focal test database. The generator can be stretched out to produce test cases in light of different WS detail languages, for example, BPEL4WS and OWL-S.

2) Test Execution Controller controls test execution in an appropriated domain. It recovers test cases from the test database, assigns them to circulated test agents, screens test runs, and gathers test comes about.

3) Test Agents are scattered in a LAN or WAN area. An agent is an intermediary that performs remote testing on target services with particular utilization profiles and test information.

4) Test analyzer examines test comes about, assess the nature of services and create test reports. All the gatherings including service supplier, service specialist and service purchaser can get to the databases with various benefits.

B. This general framework can be connected to test Services at Various Levels, for Example,

1) L1: Test the individual tasks of an atomic service;

2) L2: Test the blend tasks of atomic services;

3) L4: Test the combination tasks of composite services.

This paper centres on testing the atomic services, that is, L1 and L2 testing. It exhibits the examination of programmed test age in light of the determination of a service interface and the WSDL detail.

©IJRASET (UGC Approved Journal): All Rights are Reserved

III. WSDL-BASED TEST CASE GENERATION

In view of the WSDL schema, test cases are produced from four levels as appeared in Figure 2

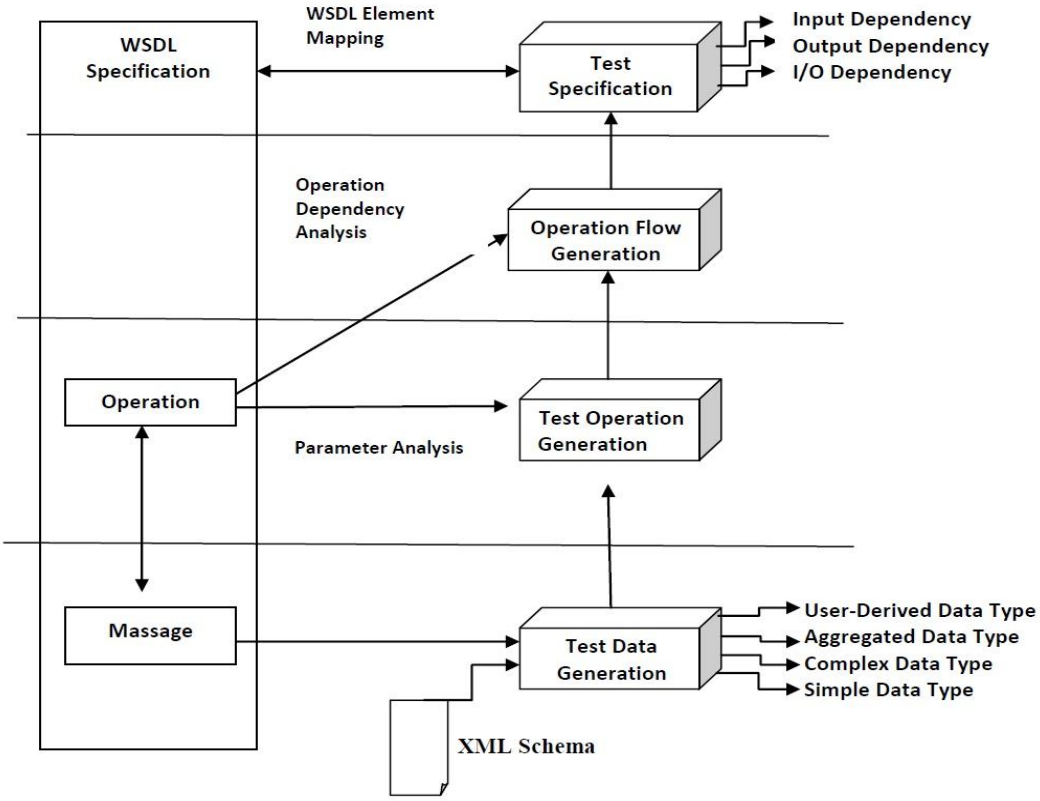


Figure 2: WSDL-Based Test Case Generation

A. Test data are created from WSDL message definition in view of XML schema data types. Four types of data are talked about including straightforward data compose, complex data write, aggregated data write and client determined data compose.

B. Test operations are created in light of analysis of its related parameters (messages).

C. Operation flows are produced to test an arrangement of operations by operation dependency analysis. Three types of conditions are recognized: input dependency, yield dependency and information/yield dependency. The paper talks about how these conditions can be utilized as a part of test age.

D. Test specification age is to at last encode the test cases in XML documents. Experiencing the levels, numerous test cases can be created to test the administration including singular operations and also composite operations from both positive and negative points of view.

IV. TEST DATA GENERATION BASED ON XML SCHEMA DATA TYPE

As indicated by the service portrayal, messages are extricated, bound to the activities and broke down to generate test cases. Test cases are generated based on data compose investigation. The XML schema [1] punctuation characterizes two message types: straightforward compose and complex writes [7] [8]. Distinctive types can be joined to amassed data types. Clients can likewise characterize their own particular types got from the work in data types.

The XML standard characterizes an arrangement of work in basic data types, including crude types and determined types. Every basic kind is related with an arrangement of features which portray the specific parts of a data compose. Features are of two types: central aspects semantically dynamic the data write, for example, equivalent, requested, limited, cardinality and numeric; and compelling or non-major aspects are utilized to confine the data values, for example, length, design, counts, and so forth.

This paper examines how to programmed generate test data based on data types and their related obliging aspects. An information base is built up where each form in basic data compose is related with default aspects definition and sets of candidate values based on the test procedures, for example, random values and boundary values. That is, SDT (Simple Data Type) is characterized as

©IJRASET (UGC Approved Journal): All Rights are Reserved

<FACET, DV> where FACET is the arrangement of named requirements, {<namei, constraints i>}; DV is the arrangement of default values comparing to various testing strategies, {<strategyi, {valuek}>}. For example, type "string" is recorded with its facets and default values as {Length of data=any, minimum length=0, maximum length=MAX, patter=any type, enumeration= any type, whitespace= any type, ordered=false/true, bounded=false/true, cardinality=accountably infinite, numeric=false/true}, where 'any' is a pre-characterized value sense no limitations. The type is also assigned a default set of random values {"abce", "=_+/~", … }. Once the WSDL text is parsed, the analyzer will take out the data, generate instances of their comparing data type, obtain and verification the facet values with the data instances. Every facet is associated to a facet generator, for examples, for a data of type "string", lenGen () to generate the length of a string, patten Gen() to generate a string according to the patterns. Test generator coordinates the facets generators and creates the outcome data.
The basic algorithm is as per the following:

A. Retrieve the database for data type definition.
B. For each facet of the data type, if it is not characterized in the instance, take its default value from the database; otherwise, take the instance value.
C. Invoke the facet generators in arrangement to generate test data based on the facet definition of the instance.

Complex data type characterizes a composition of straightforward and/or complex data types. It characterizes the "arrangement", "decision", and "all" relationship among part data types. To generate test data of complex data types, the generator recursively analyses the structure of the data type until the point when it reaches the basic type.
The basic algorithm is as per the following:
Analyse the hierarchical tree structure of a complex data type.

D. Traverse the tree, and at each tree node
1) If it is a straightforward data type, perform basic data type analysis.
2) If it is a "grouping" structure node, generate an arrangement of test cases with each test case comparing to a requested combination of the child nodes.
3) If it is a "decision" structure node, generate an arrangement of test cases with each test case comparing to one child node.
4) If it is an "all" structure node, generate an arrangement of test cases with each test case Compare to a random combination of the child nodes.

Data types can also be consolidated to shape aggregate or gathering data types, for example, List or Union. Similar to the complex data type analysis, different algorithms are produced to generate sets of test cases based on the constraint relationships among the part data types. Specialist co-ops can also characterize their own benefit particular data types as user-derived data types which are derived from worked in data types. A user-derived data compose is first dissected taking the learning from the inherent kind that it is derived from. Particular induction highlights are broke down at runtime case by case.

V. OPERATION FLOW GENERATION BASED ON DEPENDENCY ANALYSIS

A service may contain various operations. Test cases are produced for testing singular operations and also a grouping of operations called operation flow. In view of the test data age, numerous test cases can be produced for an operation by consolidating its parameter data sets. Besides, sets of test cases can be produced to test complex operation flows of an administration by operation dependency examination and operation blend.
The set of operations inside an administration definition may rely upon each other. For instance, for the Stock Quote benefit, the user can get to the "Get Last Trade Price" operation if and just in the event that it has performed "Login" operation and a verified recognizable proof is returned. Thus, the "Get Last Trade Price" operation is subject to the "Login" operation. Tsai et. al talked about the augmentation of WDSL to encourage test case generation[8]. Especially, they propose four expansions to current WDSL specification: input-output dependency, summon successions, concurrent arrangements, and progressive functional depiction. Input-output dependency recognizes the relationship between interfaces, which can decrease the quantity of test cases for regression testing. Conjuring arrangements catch data flow and control flow between administrations while concurrent groupings catch the concurrent conduct of multi-benefit exchanges. The grouping specifications can be utilized for way testing. Various levelled functional portrayal displays the interface functional specifications in a progressive structure, which can encourage programmed functional testing.

©IJRASET (UGC Approved Journal): All Rights are Reserved

This paper depends on current WSDL specification and dissects the data conditions among operations. Three types of conditions are characterized: input dependency (ID), input/output dependency (IOD), and output dependency (OD). An operation op1 is said to be input/output reliant on anther operation op2 if no less than one of the input messages of op1 is the output message of op2. Two operations are said to be input subordinate on the off chance that they both offer a set of messages as their input message. So also, two operations are said to be output subordinate on the off chance that they both offer a set of messages as their output message. We utilize Input (operation) to speak to the set of input messages of an operation op and Output (op) as the set of output notes, the descriptions are given as follows:

A. Portrayal 1

Operation Input Dependency mean If [Input(op1) ∩ Input(op2) ≠ ϕ], at that point op1 and op2 are input dependent, that is, ID(op1,op2) .

B. Portrayal 2

Operation Output Dependency: If [Output (op1) ∩ Output (op2) ≠ ϕ], at that point op1 and op2 are output dependent, that is, OD (op1, op2).

C. Portrayal 3

Operation Input/output Dependency: If [Input (op1) ∩ Onput (op2) ≠ ϕ], at that point op1 is input/output dependent on op2, that is, IOD (op1, op2). Op2 is said to be the point of reference of op1, and op1 is said to be the dependent of op2.

IOD fulfils the transitive properties. That is, for any three operations op1 , op2 and op3 , if IOD(op1,op2) and IOD(op2 ,op3) , then IOD(op1,op3) . IOD forces succession requirements on the operations. An operation is relied upon to be practiced before its dependent operations and after all its point of reference operations. In this manner, a recursive IOD investigation process is first performed and a reliance graph is made. The operations are orchestrated all together as indicated by their IOD connections.

An operation may have various points of reference recorded as PRE(op). A total test requires a full mix of operations in PRE set. At the point when the quantity of operations is extensive, this mix can be dangerous. Output dependency can be utilized to classify the point of reference operations into various groups. Two operations that are output dependent are isolated into independent sets. The operations in various groups are not consolidated, and each will be additionally formed into various test cases. Along these lines, it can significantly lessen the quantity of test cases produced.

The accompanying calculation depicts the general procedure of dependency-based operation flow approach:

1) For every operation, perform IOD analysis and distinguish the arrangement of its points of reference PRE and the arrangement of its dependents DEP.
2) For every operation, perform OD analysis on its PRE and ID analysis on its DEP. As a result, an arrangement of sub-groups are made, that is, PRE={prei} and DEP={depi}.
3) Select an operation whose PRE is unfilled.
4) IF DEP (op) isn't void, make an arrangement of test cases as TC [].
5) For each depi(op) in DEP(op), make a test case OP[] as a vector of operations, and include operation and every operation in depi(op) to OP[].
6) Repeat 4-5 until all DEP (op) are navigated.
7) Repeat 3-6 until the point that all operations with PRE discharge are crossed.

VI. SERVICE TEST SPECIFICATION & TEST COVERAGE ANALYSIS

The created test cases are encoded in XML called Service Test Specification (STS). STS can be effectively traded between circulated testing parts or bound to network gathering, for example, SOAP for test execution. STS takes the ideas of WSDL including activities, part, message, input /output, reflecting a nature mapping between WSDL elements and test elements.

The Service Test Specification schema definition is appeared bellow.

<operation-name name=”operation”> <input-name name=”input”>
<message-name name=”message”> <part-name name=”part”>part-value</part-name>
</message-name>
</input-name>
<output-name name=”output”>

©IJRASET (UGC Approved Journal): All Rights are Reserved

<message-name name=”message”> <part-name name=”part”>part-value</part-name>
</message-name>
</output-name>
</operation-name>

A. Test Coverage Analysis

WSDL gives the premise to work based test cases generation and programmed service functional testing. The test case generation calculation parses through the whole detail and produces test cases for each interface work characterized in WSDL. In light of the WSDL constructions, different scope criteria can be connected to control and assess create comes about.

An analysis was directed on 356 service WSDL determinations looked from the Internet, including address finder, airplane terminal data, and so forth, with through and through 2050 operations. 8200 test cases are produced covering every one of the operations and messages. Around 7500 test cases are fruitful practiced on 1800 operations. The fizzled test cases are for the most part because of blunders in WSDL documents.

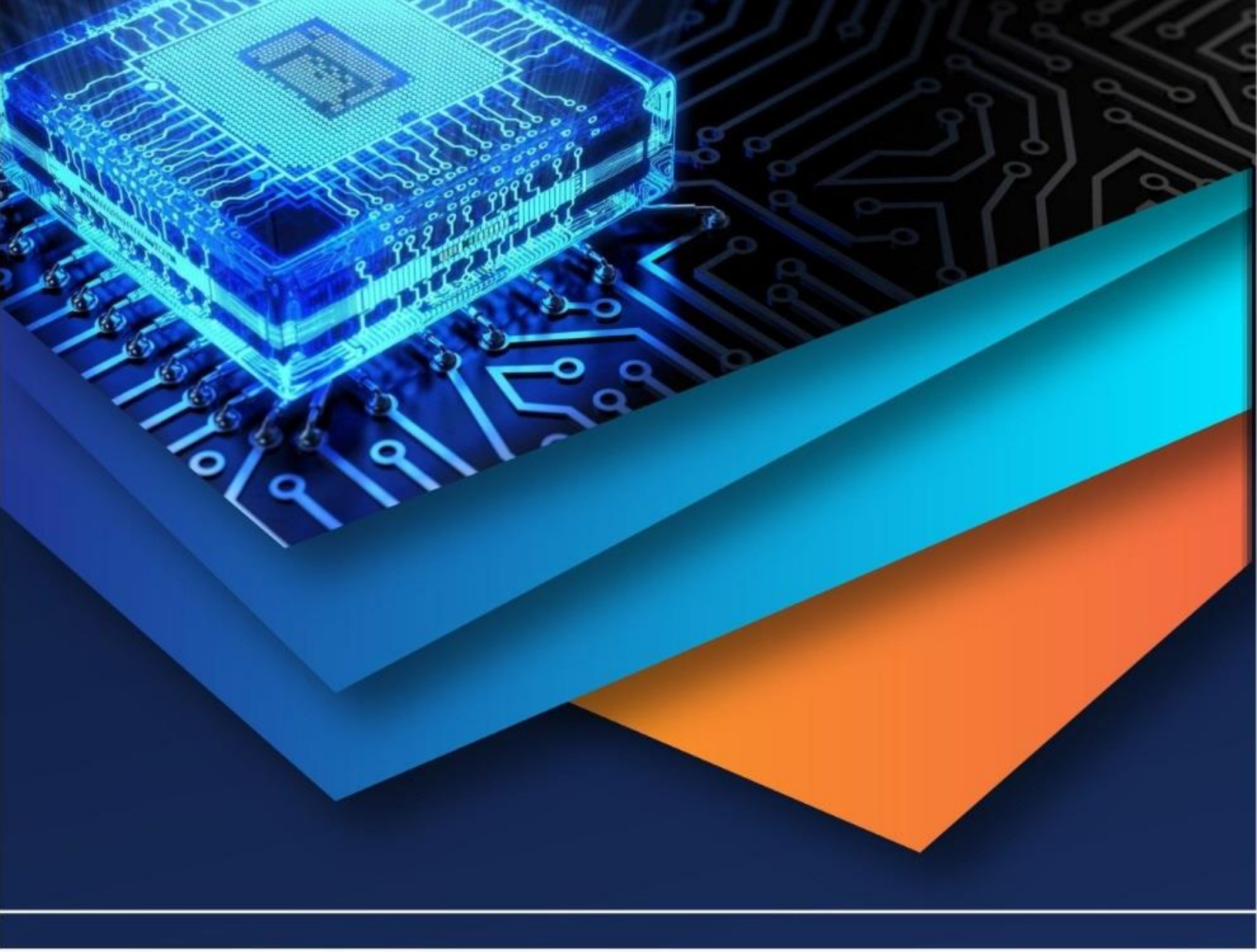
VII. CONCLUSIONS AND FUTURE WORK

This paper displayed the procedure and algorithms of automatic test case generation depend on WSDL specification for conveyed service testing. The case study demonstrated this proposed approach could viably encourage test automation and incredibly increase test productivity. In any case, with WSDL, just test cases for individual services could be created, either single operations or mix of operations. In addition, with current WSDL specification, test data and operations were produced for the most part by syntactical analysis, for example, data type analysis and activity dependency analysis. Future research will broaden extend current work from following points of view: 1) Service flow specification analysis and automatic test generation for composite services; 2) Semantic service specification analysis to investigate more semantic data for more smart test case generation.

REFERENCES

[1] J. Bloomberg, Web Services Testing: Beyond SOAP, ZapThink LLC, Sep 2002.
[2] H. Foster, S. Uchitel, J. Magee, and J. Kramer, “Model-based verification of web service compositions”, In Proc. ASE'03, 2003.
[3] N. Looker and J. Xu, “Assessing the Dependability of SOAP RPC-Based Web Services by Fault Injection”, Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS’03), pp. 163-163, Oct. 2003.
[4] S. Nakajima, “Model-checking verification for reliable web service”, In Proc.OOPSLA'02 Workshop on OOWeb Services, 2002.
[5] S. Narayanan and S. Mcllraith, “Simulation, verification and automated composition of web services”, In Proc. WWW'02. ACM, 2002.
[6] H. Huang, W.T. Tsai, R. Paul, Y. Chen, "Automated Model Checking and Testing for Composite Web Services", 8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC), Seattle, May 2005, pp, 300 - 307.
[7] W. T. Tsai, R. Paul, Y. Wang, C. Fan, and D. Wang, “Extending WSDL to Facilitate Web Services Testing”, Proc. of IEEE HASE, 2002, pp. 171-172.
[8] W. T. Tsai, Y. Chen, and R. Paul, “Specification-Based Verification and Validation of Web Services and Service-Oriented Operating Systems”, 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 05), Sedona, February 2005.
[9] X. Yi and K.J. Kochut, “A CP-nets-based Design and Verification Framework for Web Services Composition”, Proceedings of the IEEE International Conference on Web Services, pp. 756-760, March, 2004.

©IJRASET (UGC Approved Journal): All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)