International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 05 | May -2017 www.irjet.net p-ISSN: 2395-0072

A Technique for Testing Composed Web Services including Footprint

Chetan Phalke[1] , Prof. Archana Jadhav[2].

[1] Department of Computer Engineering Alard Collage of Engineering Pune, Maharashtra, India
[2]Professor, Department of Computer Engineering Alard Collage of Engineering, Pune, Maharashtra, India

---

Abstract - *Testing a service based application with more suitable approach is introduced in this paper. Paper has an architecture that responds to changes of service operation , service composition changes and allow all Footprints by Tree. Our test system is able to identify changes that occur in service operations and operational arguments in a service description of a test candidate. Automated reconfiguration is used to support the continuous operation of the testing systems during a test candidate change. Our proof-of-concept test system performs runtime testing on our model atomic and composite web services using a random testing technique with previous footprints records.*

***Key Words*: ATTCWSIF ,Footprints, Runtime testing, SOA, Software Testing, Hash Table**

1.INTRODUCTION *(Size 11 , cambria font)*

Recent Days service-based applications become for trendy and useful for verification and validation. The popularity of service-based applications makes it increasingly important that effective means for verification and validation of them are available. Due to only interface is available for web services it ts difficult to test service oriented application because there are many chance of third party services. Due to such difficulty Testing technics become more important . The dynamic execution environment of Web services means that runtime testing is necessary for verifying their ongoing reliability during deployment and production use.

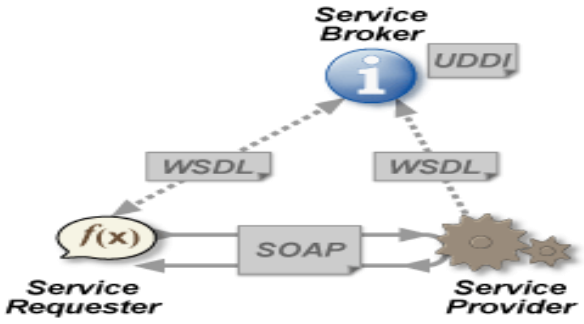


Fig -1: Web services Arch

Despite the very large number of available Web services (WS), each one, taken alone, has a relatively limited functionality. Generally, user requests cannot be satisfied by a single web service. Thus, WS should be combined to fulfill user requirements. The combination process of WS is called WS composition. Current testing approaches involve several manual tasks and, hence, are error-prone and costly. In fact, the analysis approach is accomplished by detecting, manually, critical code locations in order to verify its vulnerability. Software engineers have to investigate these program places manually to decide for themselves whether this part of the program may or may not violate the property at hand. However, making this kind of analysis is almost very subtle and complex. Furthermore, composite WS impose additional analysis complexity as they are consumed by a plethora of clients having different desires. This provides frequent changes within composite services

In this paper, we extend previous work [1] on runtime testing of atomic Web services. Our methods allow: Identification of service composition changes, including addition, deletion, and modification of a component, interface changes at runtime , and leave footprints . We have implemented a system to demonstrate these methods, and apply them to a small composite Web service.

2. REVIEW OF LITERATURE

Project research focus is on testing service oriented applications, which use atomic and composite Web services. Such web services maybe composed by thirdparty . Testing must not only take place during development, but at runtime. According to King and Ganti [3] test System will affect by Additive change: An introduction of a new component interface or an implementation. Reductive change: A removal of an existing implementation or an interface from a cloud application. Mutative change: A transformation of an existing component while retaining some of the existing functionality of that component. For any runtime testing system required artifacts related that web service. Hong and Yufeng [4]Identify by collaborating multiple web services test service partners invoked at runtime so it has Lack of software artifacts due to implementation being hidden from the users. Lack of control over test execution due to distributed component interactions.

Runtime SOA Testing Based on Hong and Yufeng's work, we suggest that runtime or live testing strategies should be used for testing SOA applications. To make runtime testing of SOA practical, it is necessary for testing methodologies to adapt as the tested system changes. SOA applications and introduced an adaptive testing framework which can continuously learn and improve the built-in test strategies as Bai et al. [5]

Mark B. Cooray, James H. Hamlyn-Harris, and Robert G. Merkel[1] used service composition details to identify deviations in composition and used WSDL service descriptions as a framework for generating test cases[1][2]. We also used service composition details to identify deviations in composition.

Existing Testing approaches involve several manual tasks, Specifically Testing a web services. Due to this manual task overall testing process becomes error-prone and costly. And we notice that Existing Architecture is lengthy without Effective traceability of web services.

3. SYSTEM OVERVIEW

A Technique for Testing Composed Web Services including Footprint test system architecture has three main layers, as shown in Fig. 2.

3.1 Services Interface layer

The Service Interface Layer provides coordination with stakeholders external to the test system. The test system administration component accepts new test requests from service providers and the interface. This Layer also responsible to make Hash Table of web services.

3.2 Services Operation layer

Services operation Layer has to apply Change capture algorithm[1] on incoming web services. Make new details about web services if a new service is there else update old details. Select test cases according it and Collect footprint Tree Details.

3.3 Services Management layer

Services Management layer Performs Testing According Services changes and perform Reports with Footprint Tree.

3.4 System Architecture

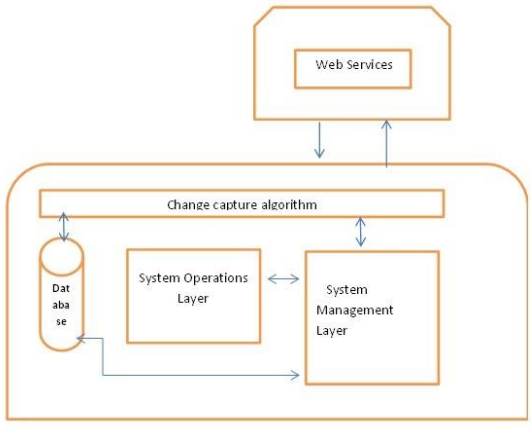


Fig -2: A Technique for Testing Composed Web Services including Footprint Architecture.

3.5 Mathematical Model

- Let 'W' be set of whole System :- W={input , process, output}
- The central outline of the proposed algorithm is as follows.
- 1)Input: Input=U,D,N Where:
  - U=URL of Web Service D= Description Of Web Service N=Name of service.
- 2) Process:
  - step 1: System Accept new URL of web Service including Description and Service Name
  - step 2: System, Check received service Existed in Database and then if it is then apply change capture algorithm by extracting there details like number of line, parameter ,operations.
  - step 3: System generate Test case Suit current testing approach and Apply test cases.
- 3) Output:
  - Report For Reliable Testing

4. ATTCWSIF DEMONSTRATION

This Demonstration was evaluate our ATTCWSIF approach met aims of work.

4.1 Input Web Services

Evaluation was conducted using simple Arithmetic services. Service function is to perform Arithmetic operations. This web service were Deployed using Axis2.

4.2 Services Interface layer

Service provider was registered first and then he was submitting his web service details.

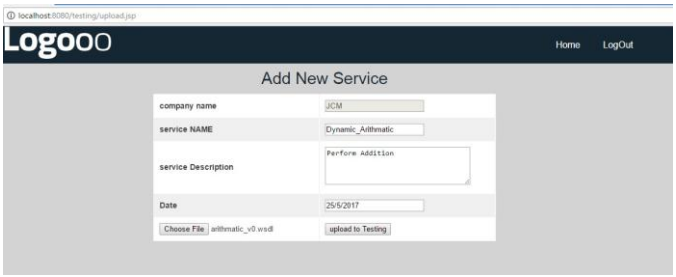


Fig -3: Uploading Web Service

4.3 Services Operation layer

Apply change capture Algorithm[1] and store result in Database.

Fig -4: Basic version of web service.

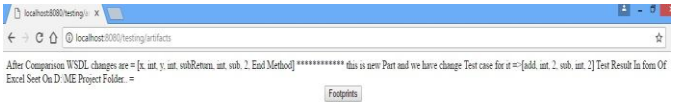


Fig -5: new version of web service.

4.4 Services Management layer

On basis of changes Test case was suggested in for of test suit that has performed and generate report of test cases and shows the Footprints.

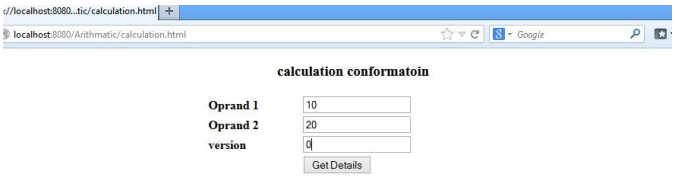
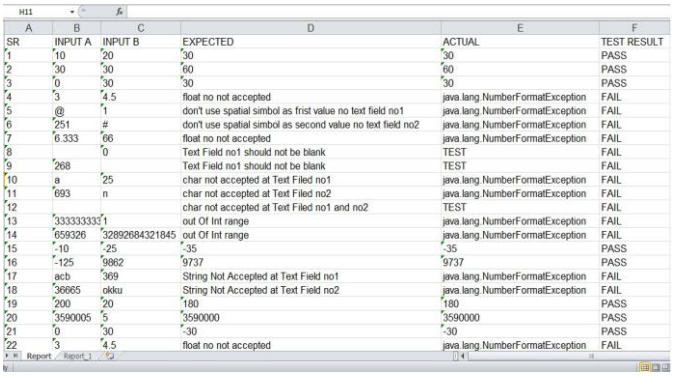


Fig -6: Testing Performed.



SR	INPUT A	INPUT B	EXPECTED	ACTUAL	TEST RESULT
1	10	20	30	30	PASS
2	30	30	60	60	PASS
3	0	30	30	30	PASS
4	3	4.5	float no not accepted	java.lang.NumberFormatException	FAIL
5	@	1	don't use spatial simbol as frist value no text field no1	java.lang.NumberFormatException	FAIL
6	251	#	don't use spatial simbol as second value no text field no2	java.lang.NumberFormatException	FAIL
7	6.333	66	float no not accepted	java.lang.NumberFormatException	FAIL
8		0	Text Field no1 should not be blank	TEST	FAIL
9	268		Text Field no1 should not be blank	TEST	FAIL
10	a	25	char not accepted at Text Filed no1	java.lang.NumberFormatException	FAIL
11	693	n	char not accepted at Text Filed no2	java.lang.NumberFormatException	FAIL
12			char not accepted at Text Filed no1 and no2	TEST	FAIL
13	333333333	1	out Of Int range	java.lang.NumberFormatException	FAIL
14	659326	32892684321845	out Of Int range	java.lang.NumberFormatException	FAIL
15	-10	-25	-35	-35	PASS
16	-125	9862	9737	9737	PASS
17	acb	369	String Not Accepted at Text Field no1	java.lang.NumberFormatException	FAIL
18	36665	okku	String Not Accepted at Text Field no2	java.lang.NumberFormatException	FAIL
19	200	20	180	180	PASS
20	3590005	5	3590000	3590000	PASS
21	0	30	-30	-30	PASS
22	3	4.5	float no not accepted	java.lang.NumberFormatException	FAIL

Fig -7: Test Report.

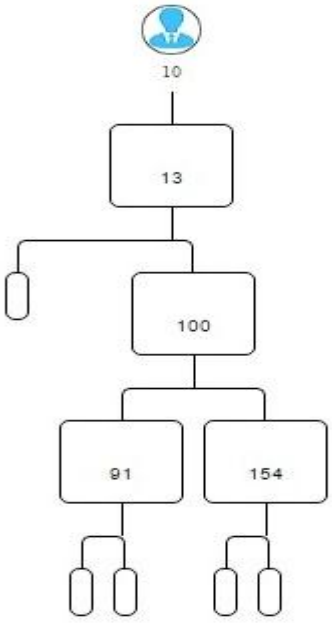


Fig -8: Footprints.

5. RESULT ANALYSIS

Here are some statistics which prove the System performance

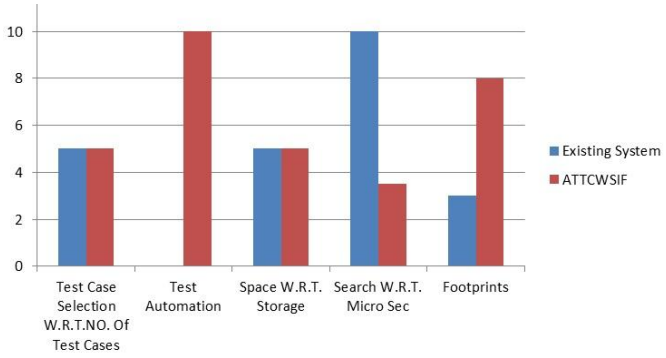


Chart -1:- Result Analysis

By Using ATTCWSIF Performance of testing a web services become fast and more effective with respect to Existing System and this ATTCWSIF perform equal, Test case Execution automation is in ATTCWSIF , Required Storage is equal in both cases , Search operation more faster in ATTCWSIF, Footprints is more effectively present for further Use.

3. CONCLUSIONS

The work presented in this paper able to provide an automated testing system that is sensitive to changes and performs reconfiguration of test artifacts and build footprints according to the behavior of the service under test. Work can be used as a starting point of further investigations of the technical and commercial barriers preventing the introduction of effective evaluation processes for service-based applications.

ACKNOWLEDGEMENT

I like to thanks all people for helping me for giving importance guidance about project work. I am very thankful to those who help me during publishing my paper as well as project dissertation stage. I am thankful to journal for allowing me the best opportunity to publish my paper. I am sincerely thankful to my (HOD) head of department, my project guide and other staff members from department for supporting me.

REFERENCES

[1] Mark B. Cooray, James H. Hamlyn-Harris, and Robert G. Merkel, Dynamic Test Reconguration for Composite Web Services,IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 8, NO. 4, JULY/AUGUST 2015.

[2] M. B. Cooray, J. H. Hamlyn-Harris, and R. G. Merke,Test reconguration for service oriented applications, in Proc. IEEE Int. Conf. Utility Cloud Comput., 2011, pp. 300305

[3] T. M. King and A. S. Ganti,Migrating autonomic self-testing to the cloud,in Proc. Int. Conf. Softw. Testing, Verication, Validation Workshops, Paris, France, 2010, pp. 438443.

[4] Z. Hong and Z. Yufeng,Collaborative testing of web services, IEEE Trans. Service Comput., vol. 5, no. 1, pp. 116130, Jan.Mar. 2012.

[5] X. Bai, C. Yinong, and S. Zhongkui, Adaptive web services testing, in Proc. 31 Annu. Comput. Softw. Appl. Conf., 2007, pp. 233236.

[6] Bai, Z. Cao, and Y. Chen,Design of a trustworthy service bro- ker and dependence-based progressive group testing, Int. J. Simul. Process Modell., vol. 3, pp. 6679, 2007.

[7] W. T. Tsai, R. Paul, Z. Cao, L. Yu, and A. Saimi,Verication of web services using an enhanced UDDI server, in Proc. 8th Int. Workshop Object-Oriented Real-Time Dependable Syst., 2003, pp. 131 138.

[8] X. Bai, X. Dezheng, D. Guilan, T. Wei-Tek, and C. Yinong,Dynamic recongurable testing of serviceoriented architecture,in Proc. 31st Annu. Int. Comput. Softw. Appl. Conf., 2007, pp. 368 378.

[9] D. Brenner, C. Atkinson, O. Hummel, and D. Stoll,Strategies for the run-time testing of third party web services, in Proc. IEEE Int. Conf. ServiceOriented Comput. Appl., 2007, pp. 114121.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

BIOGRAPHIES

Chetan Nandkumar Phalke.
Is Student of Computer Engineering at Savitribai Phule Phule Pune University , Pune . His work interest include Cloud computing, Software Testing and web services.

Archana Jadhav
Is Professor with Savitribai Phule Pune University , Pune. Her work interest include Software Testing , Software Security and web services.