Review of the Present State-of-the-Art of Dynamic Test Reconfiguration of Composite Web Services

Sirisha K L S[1], Dr. M. Chandra Mohan[2], V. Santosh Kumar[3]

Abstract— In the wake of Service Oriented Architecture (SOA) and its applications that integrated services provided by different businesses, reliability of services play a vital role in the success of those businesses. Testing is the widely accepted approach used to find reliability of services. Testing web services that are composed to form SOA applications is non-trivial and hard to achieve. Web services with BPEL workflow drive most of the businesses that affect all stakeholders. Building comprehensive test strategies for cohesiveness of underlying components and loosely coupled nature of integrated heterogeneous pieces of software is very challenging. Moreover business processes tend to change that makes the job of testing much more complex. Many researchers contributed towards testing of web services. In this paper we review the present state-of-the-art of web service composition methods, testing web services, automatic test case generating and automatic test case reconfiguration. This paper provides insights found in the literature in terms of test methodologies, tools and techniques used for testing web services and web service compositions.

Index Terms— Web services, distributed computing , composite web services, testing , test reconfiguration

1 INTRODUCTION

Service based applications became very popular and they provide certain services to end users. These applications are made up of many similar services (businesses) that seamlessly work together so as to provide quality services to users. Such applications are based on Service Oriented Architecture (SOA) which is distributed in nature. Service oriented applications include applications in banking sector, reservations, e-commerce to mention few. These applications deliver promising services that are essential to end users. Though these applications are very complex, the complexity is transparent to end users. Users enjoy services even without having knowledge on the underlying technologies. SOA based applications need to integrate many related businesses. Supply Chain Management (SCM), ATM network are best examples for that.

Reliability of service oriented applications play vital role to the success of associated businesses. Testing is one of the approaches to find reliability of service oriented applications. The present testing approaches are not adequate to provide the required level of quality assurance in service oriented applications. Stated differently, the testing of service oriented applications needs further research as they are very complex. SOA applications are exposed as atomic web services or composite web services. Atomic web service has a single interface through which other applications can interact.

- *Sirisha K L S is currently working in Keshav Memorial Institute of Technology as an Assistant Professor in the Department of Computer Science and Engineering. E-mail: klssirisha@gmail.com*
- *Dr. M. Chandra Mohan is currently working in JNTUH as a Professor in the Department of Computer Science and Engineering. E-mail: c_miryala@jntuh.ac.in*
- V.Santosh Kumar *currently working in Sreyas Institute of Engineering and Technology as an Associate Professor in the Department of Computer Science and Engineering). E-mail: vennu.santoshkumar@gmail.com*

The composite web services on the other hand are made up of multiple web services seamlessly integrated into a business process. Such application may include many services provided by third parties. Therefore testing such applications is non trivial and needs careful selection of suitable testing techniques. There are many reasons for difficulty of testing such applications. First, the services are from different providers and deployed in different servers. Second, they are probably dynamically composed at runtime. Third, the services or the underlying components may be subjected to changes without notice. Fourth, the provider of web service may not have control over the components.

Due to the dynamic nature of the web services, it is essential to have an ongoing process for runtime testing from time to time to ensure reliability. As said earlier it is complex process and needs regular testing as there might be changes to service composition that necessitates the updating of test suits, isolation of composition changes, and notification of testers so as to reconfigure test cases in order to continue testing reliability of those web services. Having understood the full spectrum of the modern service oriented applications, the need for their reliability, in this paper, we throw light on the present state-of-the-art of testing composite web services and the test case reconfiguration to adapt changes. The remainder of the paper is structured into different sections that review the available literature to have useful insights. Our endeavour is to unearth the existing research outcomes and provide research gaps that can help in further research in this area.

2 Changes in Composite Web Services that Affect Test System

While performing reliability testing of composite web services, the test system needs to know whether there are interface changes in the underlying web services and the kind of change that has been made in order to regenerate test cases. King and Ganti [2] investigated different changes that may occur to web

services and categorized them into three types as shown in Table 1.

CHANGE CATEGORY	DESCRIPTION
Additive change	It is the change caused due to introduction of new component into an existing SOA application.
Reductive change	It is the change caused due to removal of a component from SOA application.
Mutative change	It is the change caused due to transformation of existing component while preserving its functionality to a greater extend. In other words, it is caused due to changes in one of the components.

Table 1 – Different change types in service composition

3. Runtime SOA Testing

Testing SOA applications need live approaches so as to test the reliability from time to time. Bai et al. [4] studied the different runtime environments for SOA applications. They proposed adaptive testing framework that makes use of continuous heuristics to improve testing strategies. Their work is based on the broker architecture explored in [6] and [5]. Their work is an extension to the UDDI test strategy. They used a web service as feedback unit in order to get response while testing and make other consecutive decisions based on the feedback. The Bai et al. improved their work in [7] using distributed test agents in order to control and coordinate activities pertaining to testing. Their work was the first attempt to address issues related dynamic testing of web services. However, their work did not focus on the service compositions. Brenner et al. [8] also investigated into web service testing strategies in similar fashion but focused more on the runtime testing of third party services. They opined that runtime contract testing is one of the strategies to test web services. They listed out different kinds of runtime strategies for testing.

4. Service Composition Testing

Service composition is dynamic and may change in future. This is not visible to the users and web services do not disclose such information. It is impractical to run black box testing on such web services. It is essential to have a look into the internal logic in order to test a functionality of web service. There are tools that came into existing to have service composition testing as explored in [10] and [9]. However, Bucchiarone et al. [11] opined that these tools and present techniques used for service composition testing are not adequate to handle complex operations that need to address composite data types. Zheng and Yan [41] explored web service composition issues and proposed an algorithm for syntactic matching of web services that are based on planning graph model which eliminates duplicates.

Lallali et al. [42] explored on web service composition problems. They proposed automatic timed test case generation for achieving web services composition into a BPEL process. Zheng et al. [43] proposed a recommender system that can provide recommendations to support selection of web services while composing services. Chandra sekaran et al. [44] proposed a tool known as Service Composition and Execution Tool (SCET) composing web services and testing them. They used Web Service Flow Language (WSFL) based specifications to achieve this. Hausmann et al. [45] proposed a model-based discovery of web services formal software models and graph transformations.

Tsai et al. [47] explored consumer centric composition of web services. This is collaboration oriented approach in which SOA based web services are identified in consumer-centric fashion. Platzer and Dustdar [49] explored semantic web services and proposed a method for discovering web services. They built a tool known as vector space search engine for this purpose which could browse existing repositories for discovery.

5. Automatic Test Case Generation for Web Service

In case of SOA applications that are made up of web services, it is essential to know interface specification that can be obtained from Web Services Description Language (WSDL). The WSDL based test data generation was explored in [13] and [12]. However, other approaches and specifications in other languages are also available for effective test case generation. Bai et al. [27] explored automatic test case generation approaches based on WSDL. Since WSDL provides useful information on web service and its underlying operations, return types and arguments they focused on WSDL based test case generation. They built Document Object Model (DOM) from WSDL before using it for test case generation. Wang et al. [28] proposed a framework for generating test cases based on ontologies. Their approach was model-driven. They made use of semantic specifications built on OWL-S. They also used Petri nets for achieving model-driven test case generation.

Siblini and Mansour [29] proposed a new method known as mutation analysis for testing web services. Their method uses WSDL in order to generate many mutant web service interfaces that are used to generate test cases automatically. Tsai et al. [30] proposed a method that extends WSDL to generate test cases with high coverage. They used four kinds of extensions known as concurrent sequence specifications, hierarchical functional description, invocation sequence, and input-output dependency. Tsai et al. [31] proposed an XML based object oriented framework for automatic test case generation. It converts WSDL specifications into different scenarios from which test cases are captured and generated. However, they approach focused on integration testing and module testing is not given importance.

Sneed and Huang [32] proposed a tool for testing web services. The tool was named WSDL Test which is used to generate and validate test data. Thus it can help in accomplishing the task. The tool is finding with simple WSDL files but does not give guaranteed performance with very complex WSDLs. Heckel and Lohmann [33] proposed a method known as contract-based web service testing which exploits matching of service descriptions of provider and requirements of consumer and visualizes contracts. Then the operational interpretation

of rules will help the method to generate test cases.
Bai et al. [34] proposed ontology based web services testing method which makes use of a test ontology model which represents test related concepts and their relationships to help in test case generation. The method also has provision for consistency checking and generates less number of test cases. Zhu [35] proposed a framework for testing web services that are in SOA applications. In [36] perturbation based testing was proposed. Actually XML messages are modified and thus test cases are explored and generated. This model proved to be efficient but lacks in reconfiguration abilities. Chou and Guo [37] proposed control flow analysis based testing of web services that provide efficient and accurate means for testing besides providing high test coverage. However they do not generate test cases automatically.
Al-Masri and Mahmoud [38] proposed a tool for searching web services over Internet. Web Service Crawler Engine (WSCE) proposed by them generates metadata information pertaining to deployed web services. Morales et al. [39] explored passive testing of web services with timed extended invariants that represent time and data constraints. The method analyzes execution traces for finding formal extended invariants. Brenner et al. [40] performed runtime analysis based testing of web services provided by third parties. They could identify different types of tests that are possible based on the runtime analysis.
Conroy et al. [48] proposed an approach to generate test cases automatically based on the GUI of web service clients. This exploits visualization and accessibility mechanisms to generate unit test cases. Drag and drop and point and click operations were used as basis to identify test case requirements.

6. Testing Web Services

Many researchers contributed to the testing strategies of web services. Zhu et al. [3] proposed a framework for collaborative testing of web services. The collaborative testing web services are provided by third parties. They are discovered and used at runtime using ontology of software testing known as STOWS. Test brokers are used to realize the runtime composition of test services. Ontology is used to represent the services available and the relationship among them. Bultan et al. [14] focused on composite web services and the tracking and analyzing of conversations among such web services. They could identify the differences in synchronous and asynchronous conversation behaviours. They could gain knowledge of that for both bottom-up and top-down web service specifications.
Lin et al. [15] threw light on application of safe regression testing on web services built in Java platform. They worked on the white-box testing methods for Java web services using Apache Axis toolkit. Belli and Linschulte [17] proposed an event-based approach for testing web services that contain specific functionality. They made use of sequence graphs and achieve fault management. Tsai et al. [18] focused on group testing of web services using voting algorithm. They used multi-dimensional test data besides clustering them for effective testing of web services. Neisse et al. [19] explored the bandwidth consumption of web services so as to find their feasibility in different conditions. Ciupa et al. [20] focused on a tool named ARTOO fro adaptive testing of applications built using Object Oriented (OO) languages. Chen et al. [21] explored adaptive random testing as test selection strategy based on runtime heuristics. Testing non-testable applications [22], testing semantic web services [23], specification based testing of web services that are semantic in nature [24], test case prioritization based on quota [25] and ontology usage for representing composite web services and help in testing [26] are other significant researches on web service testing.
Penetration testing was explored in [46] for securing web services from intrusion attacks. Especially they focused on the testing of SQL injection attacks into web services by using interface monitoring and enhancing attack signatures. They explored SOAP based web services to detect injection vulnerabilities in web services. Yu et al. [50] explored Testing as a Service (TaaS) that runs in cloud for providing testing capabilities to cloud users. This could help could users to have consistent test platform without investing time and money on proprietary testing mechanisms. Kavalli et al. [51] proposed a framework named WebMov for testing composite web services. They employed passive testing techniques to know the robustness and conformance of composite web services. They validated their framework with travel reservation case study.
Location based web services and prioritization of test cases [52], runtime behaviour analysis for conversational web services [53], TGSE tool for testing composite web services [54], a framework for scalable web services [55], investigation of broker role in web service discovery [56], JOpera for testing web services in Agile methodology [57], web services composition based on timed modelling towards automated testing [58], and exploration of web services presence in Internet [59] are other researches focused on web services and the testing of them in SOA environment.

7. Test Reconfiguration for Service Oriented Applications

Once test cases are generated for SOA applications, the reconfiguration of the test cases is essential as there might be different changes in services as explored in [2]. Cooray et al. [1] proposed architecture for test system. They generated test data based on the information available in WSDL content. Their test system architecture includes many components such as test manager, coordinator, WSDL tracker, WSDL resolver, test generator, database coordinator, service client, and change manager. The test manager is responsible for managing entire test process. Coordinator is responsible for coordinating other components in the test system. Change manager is responsible for analyzing changes in the service composition to initiate changes for the test case reconfiguration. Test generator component is used to generate test cases automatically. WSDL tracker is used to ensure that WSDL is available for use. WSDL resolver is used to decompose WSDL and gain knowledge from that. Database controller is used to handle database related queries. Service client is the program built in Apache Axis [16] for testing web services.
Cooray et al. [60] proposed a framework that can be used to reconfigure test cases that have been generated when test compositions are changed. The framework has provision for generating test cases automatically. It makes use of the test composition changes explored in [2] for automatic reconfigu-

ration of test cases. However, the framework supports test cases of operations with simple data types. It does not support complex and composite data types in the web service operations.

8. SUMMARY

This section provides the summary of important research as shown in Table 2. It includes the methods used, their merits and demerits.

Table 2- Summary of important research on testing web services

Ref	Method	Advantages	Drawbacks/ Limitations	Remarks
[3]	Collaborative testing	Reuse of test services	-	Uses broker architecture
[14]	Conversation analysis	Runtime behaviour is known	-	Differentiation of synchrochronous and asynchronous calls
[15]	Safe regression testing	Testing internal logic	-	Tested only Java web services
[17]	Event-based testing approach	Event sequence graphs help explore	-	Fault management
[18]	Group testing	Voting multidimensional data and clustering	-	Voting algorithm is used
[27]	WSDL based test case generation	Specification based approach	Dependency on WSDL	DOM is used
[28]	Ontology based test case generation	Good representation of knowledge	-	OWL-S was used
[29]	Mutation analysis	Test accuracy	Generates many mutated web service interfaces.	WSDL based test cases
[30]	Extending WSDL	High test coverage	-	Four kinds of extensions are used
[31]	XML based OO testing framework	Integration testing	Module testing is not suitable	Converts WSDL specifications into test scenarios
[32]	WSDLTest tool	Generates and validates test data	Does not work for complex WSDLs	Uses pre and post conditions
[33]	Contract based testing	Useful for unit testing of web services	Works in simulated environment only	Contract rules are automatically interpreted
[34]	Ontology based testing	Consistency checking and reducing number of test cases	-	OWL-S is used
[36]	Perturbation based testing	Efficient approach	Reconfiguration is not done	Supports document or message-passing style
[37]	Control flow analysis	Efficient, accurate and high test coverage	Automatic test case generation is not done	SPARQL queries are used
[38]	Web Service Crawler Engine (WSCE)	Discovers web services	Discovery process is not controllable	UDDI business registries are exploited
[39]	Passive testing of web services	Invariants representing data and time constraints	Time complexity	Case study based approach

[48]	Test case generation from GUI	Accessibility technologies and visualization	Locating distributed GAPs is not yet realized	Explored unit test cases
[50]	Testing as a Service (TaaS)	Testing capabilities to cloud users	Limited testing services	Unit testing services were explored
[51]	WebMov tool	Testing composite web services	Evolution of services was not yet explored	Passive testing

Table 2 – Summary of important research

9. CONCLUSIONS AND FUTURE WORK

In this paper we reviewed literature on the present state-of-the-art of web services testing. SOA applications are composed with proprietary and third party web services. Testing such applications is a challenging task. Different test strategies are required that are specific to web services. The rationale behind the difficulty lies in the fact that web services are in distributed environment and their location is not known priori and discovered at run time. WSDL, SOAP and UDDP are to be understood and WSDL needs to be explored to perform operations and test web services. To reiterate the fact, testing web services is a complex job and that needs systematic approach to ensure reliability of web services. BPEL processes are composed with multiple web services and testing them is hard to achieve. In this paper we reviewed the literature and presented in this paper the insights pertaining to web services composition, testing web services, automatic test case generation, and test case reconfiguration. This research can be extended further to implement automated test case generation for composite web services besides support for automatic test case reconfiguration in adaptive fashion.

REFERENCES

[1] M. B. Cooray, J. H. Hamlyn-Harris, and R. G. Merkel, "Test recon- figuration for service oriented applications," in Proc. IEEE Int. Conf. Utility Cloud Comput., 2011, pp. 300–305.

[2] T. M. King and A. S. Ganti, "Migrating autonomic self-testing to the cloud," in Proc. Int. Conf. Softw. Testing, Verification, Validation Workshops, Paris, France, 2010, pp. 438–443.

[3] Z. Hong and Z. Yufeng, "Collaborative testing of web services," IEEE Trans. Service Comput., vol. 5, no. 1, pp. 116–130, Jan.–Mar. 2012.

[4] X. Bai, C. Yinong, and S. Zhongkui, "Adaptive web service es testing," in Proc. 31 Annu. Comput. Softw. Appl. Conf., 2007, pp. 233–236.

[5] X. Bai, Z. Cao, and Y. Chen, "Design of a trustworthy ser vice broker and dependence-based progressive group test ing," Int. J. Simul. Process Modell., vol. 3, pp. 66–79, 2007.

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

[7] X. Bai, X. Dezheng, D. Guilan, T. Wei-Tek, and C. Yinong, "Dynamic reconfigurable testing of service-oriented archi tecture," in Proc. 31st Annu. Int. Comput. Softw. Appl. Conf., 2007, pp. 368– 378.

[8] D. Brenner, C. Atkinson, O. Hummel, and D. Stoll, "Strat egies for the run-time testing of third party web service es," in Proc. IEEE Int. Conf. Service-Oriented Comput. Appl., 2007, pp. 114–121.

[9] C. Barbara G. Ryder,Ana Milanova,David Wonnacott. (2004). Testing of Java Web Services for Robust ness. ACM, p271-280.

[10] C.-H. Liu, S.-L. Chen, and X.-Y. Li, "A WS-BPEL based structural testing approach for web service composi tions," in Proc. IEEE Int. Symp. Service-Oriented Syst. Eng., 2008, pp. 135–141.

[11] A. Bucchiarone, H. Melgratti, and F. Severoni, "Testing service composition," in Proc. 8th Argentine Symp. Softw. Eng., Mar del Plata, Argentina, 2007, pp. 1–16.

[12] X. Bai, D. Wenli, T. Wei-Tek, and C. Yinong, "WSDL-based automatic test case generation for web services testing," in Proc. IEEE Int. Workshop Service-Oriented Syst. Eng., 2005, pp. 207–212.

[13] M. Chunyan, D. Chenglie, Z. Tao, H. Fei, and C. Xiaobin, "WSDLbased automated test data generation for web service," in Proc. Int. Conf. Comput. Sci. Softw. Eng., 2008, pp. 731–737.

[14] T. Bultan and Jianwen Su,Xiang Fu. (2006). Analyzing Conversations of Web Services. *IEEE*, p.20-30.

[15] F. Lin, Michael Ruth, Shengru Tu. (2006). Applying Safe Regression Test Selection Techniques to Java Web Servic es. *IEEE*, p1-10.

[16] S. Chan Oh, Dongwon Lee,Soundar R.T. Kumara. (2008). Effective Web Service Composition in Diverse and Large-Scale Service Networks. *IEEE*. 1 (1), p.20-30.

[17] Fevzi Belli, Michael Linschulte. (2008). Event-Driven Modeling and Testing of Web Services. *IEEE*, p1370-1381.

[18] W. T Tsai, Yinong Chen, Dawei Zhang, Hai Huang. (2005). Voting Multi-Dimensional Data with Deviations for Web Services under Group Testing. *IEEE*, p.90-101.

[19] Ricard. (2004). Implementation and Bandwidth Con sumption Evaluation of SNMP to Web Services Gate ways. *IEEE*, p271-280.

[20] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "ARTOO: Adaptive random testing for object-oriented software," in Proc. 30th Int. Conf. Softw. Eng., Leipzig, Germany, 2008, pp. 71–80.

[21] T. Y. Chen, H. Leung, and I. K. Mak, "Adaptive random testing," in Proc. 9th Asian Comput. Sci. Conf., 2004, vol. 3321, pp. 320–329.

[22] E. J. Weyuker, "On testing non-testable programs," Computer J., vol. 25, pp. 465–470, 1982.
[23] M. Kluscha, Benedikt Fries , Katia Sycarac. (2009). OWLS-MX: A hybrid SemanticWeb service matchmaker for OWL-S services.*Elsevier*. 7 , p1-10.
[24] M. Shaban Jokhio, Gillian Dobbie and Jing Sun. (2009). Towards Specification Based Testing for Semantic Web Services. *IEEE*, p.20-30.
[25] S. S Hour, Lu Zhang. (2008). Quota-Constrained Test-Case Prioritization for Regression Testing of Service-Centric Systems. *IEEE*, p1-10.
[26] J. Nitzsche, D. Wutke, and T. vanLessen, "An ontology for executable business processes," in Proc. Semantic Business Process Product Lifecycle Manage. in conjunct tion 3rd Eur. Semantic Web Conf., Innsbruck, Austria, 2007, pp. 52–63.
[27] Xiaoying Bai, Wenli Dong. (2005). WSDL-Based Auto matic Test Case Generation for Web Services Test ing. *IEEE*, p1-10.
[28] Yongbo Wang, Xiaoying Bai, Juanzi Li, Ruobo Huang. (2007). Ontology-Based Test Case Generation for Testing Web Services. *IEEE*, p271-280.
[29] Reda Siblini, Nashat Mansour. (2005). Testing Web Ser vices. *IEEE*, p1370-1381.
[30] W. T. Tsai, Ray Paul, Yamin Wang, Chun Fan, and Dong Wang. (2002). Extending WSDL to Facilitate Web Servic es Testing. *IEEE*, p.20-30.
[31] W. T. Tsai, Ray Paul, Weiwei Song, Zhibin Cao. (2002). Coyote: An XML-Based Framework for Web Services Testing. *IEEE*, p1-10.
[32] Harry M. Sneed. (2006). WSDLTest – A Tool for Testing Web Services.*IEEE*, p.20-30.
[33] Reiko Heckel,Marc Lohmann. (2005). Towards Contract-based Testing of Web Services. *Elsevier*. 116, p1-10.
[34] Xiaoying Bai and Shufang Lee,Wei-Tek Tsai and Yinong Chen. (2008). Ontology-Based Test Modeling and Parti tion Testing of Web Services.*IEEE*, p271-280.
[35] Hong Zhu. (2006). A Framework for Service-Oriented Testing of Web Services. *IEEE*, p1-10.
[36] Lourival F. de Almeida Júnior and Silvia R. Vergilio. (2006). Exploring Perturbation Based Testing for Web Services. *IEEE*, p1-9.
[37] Li Li, Wu Chou, Weiping Guo. (2008). Control Flow Analysis and Coverage Driven Testing for Web Servic es. *IEEE*, p271-280.
[38] Eyhab Al-Masri and Qusay H. Mahmoud. (2008). Inves tigating Web Services on the World Wide Web. *WWW 2008 / Refereed Track: Web Engineering - Web Service Dep loyment*, p.20-30.
[39] Gerardo Morales, Stephane Maag, Ana Cavalli. (2010 Services . *IEEE*, p.20-30.
[40] Daniel Brenner, Colin Atkinson, Oliver Hummel, Diet mar Stoll. (2007). Strategies for the Run-Time Testing of Third Party Web Services. *IEEE*, p271-280.
[41] Xianrong Zheng,Yuhong Yan. (2008). An Efficient Syn tacticWeb Service Composition Algorithm Based on the Planning Graph Model. *IEEE*, p.1370-1381.
[42] Mounir Lallali , Fatiha Zaidi , Ana Cavalli , Iksoon Hwang. (2008). Automatic Timed Test Case Generation forWeb Services Composition.*IEEE*, p1-10.
[43] Zibin Zheng, Hao Ma, Michael R. Lyu, Irwin King,Shatin, N.T.,and Hong Kong . (2009). WSRec: A Colla borative Filtering Based Web Service Recom mender System. IEEE, p1-8.
[44] Senthilanand Chandrasekaran , John A. Miller , Gregory S. Silver , Budak Arpinar & Amit P. Sheth. (2010). Performance Analysis and Simulation of Compo site Web Services. IEEE, p1-14.
[45] Jan Hendrik Hausmann, Reiko Heckel,and Marc Loh mann . (2004). Model-based Discovery of Web Services. IEEE, p1-8.
[46] Nuno Antunes,and Marco Vieira . (2011). Enhancing Pe netration Testing with Attack Signatures and Interface Monitoring for the Detection of Injection Vulnerabilities in Web Services . IEEE, p1-8.
[47] W.T. Tsai, Bingnan Xiao, Raymond A. Paul ,and Yinong Chen . (2006). Consumer-Centric Service-Oriented Archi tecture: A New Approach . IEEE, p1-6.
[48] Kevin M. Conroy, Mark Grechanik, Matthew Hellige, Edy S. Liongosari, and Qing Xie . (2010). AutomaticTest GenerationFromGUIApplicationsForTestingWebServic es.IEEE, p1-10.
[49] Christian Platzer and Schahram Dustdar . (2005). A Vec tor Space Search Engine for Web Services. IEEE, p1-9.
[50] Lian Yu, Wei-Tek Tsai1, Xiangji Chen, Linqing Liu, Yan Zhao, Liangjie Tang,and Wei Zhao2 . (2010). Testing as a Service over Cloud . IEEE, p1-8.
[51] Ana Cavalli1, Tien-Dung Cao2, Wissam Mallouli3, Eliane Martins4, Andrey Sadovykh5, Sebastien Salva6,and Fati AdedicatedframeworkforthemodellingandtestingofWeb Services composition. IEEE, p1-8.
[52] Ke Zhai, Bo Jiang,W. K. Chan,and T. H. Tse. (2010). Tak ing Advantage of Service Selection: A Study on the Test ing of Location-Based Web Services through Test Case Prioritization . IEEE, p1-8.
[53] Dimitris Dranidis,Ervin Ramollari,and Dimitrios Kourte sis .(2009). Run-time Verification of Behavioural Con formance for Conversational Web Services . IEEE, p1-9.
[54] Tien-Dung Cao1, Patrick F´elix1, Richard Castanet1 and Ismail Berrada2 . (2009). TestingWebServicesComposi tionusingtheTGSETool. IEEE, p1-8.
[55] Satish Srirama, Eero Vainikko, Vladimir Šor,and Mat thias Jarke . (2010). Scalable Mobile Web Services Media tion Framework . IEEE, p1-6.
[56] Colin Atkinson, Philipp Bostan, Oliver Hummel and Di etmar Stoll . (2007). A Practical Approach to Web Service Discovery and Retrieval . IEEE, p1-8.
[57] Cesare Pautasso . (2005). JOpera: an Agile Environment for Web Service Composition with Visual Unit Testing and Refactoring. IEEE, p1-3.
[58] Mounir Lallali, Fatiha Zaidi,and Ana Cavalli . (2008) TimedModelingofWebServicesCompositionforAutoma ticTesting. IEEE, p1-10.
[59] Yan Li, Yao Liu, Liangjie Zhang, Ge Li, Bing Xie+, and Jiasu Sun . (2007). An Exploratory Study of Web Services on the Internet . IEEE, p1-8.
[60] Mark B. Cooray, James H. Hamlyn-Harris and Robert G. Merkel . (2015). Dynamic Test Reconfiguration for Com

posite Web Services . IEEE, p1-13.