Research Review on Web Service Composition Testing

Zhoujie Du[1,2(✉)] and Huaikou Miao[1,2]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
duzhoujie@163.com
[2] Shanghai Key Laboratory of Computer Software Testing and Evaluating, Shanghai, China

Abstract. Web services composition is designed to achieve a more powerful and large-grained services with organic synthesis of different Web services. In order to guarantee the quality of the Web services composition, comprehensive and adequate testing of the Web services composition is required. However, the dynamic and distributed characteristics of Web services combination make its testing technology and method have big difference with the traditional software testing and bring a large of challenges. In this paper, we summarize and analyze the definition, architecture, testing methods and testing techniques of Web service composition. In addition, we also analyze and prospect the progress of Web services combination testing.

Keywords: Web service · Web service combination · Testing methods · Testing techniques

1 Introduction

Web service is a software system that is unified by URI (unified resource identification). As a special kind of service, Web Service not only realizes the characteristics of remote access through network, but also inherits the characteristics of autonomy, openness and self-description of general services. Different organizations have different understandings and definitions of Web service. However, there was no fixed definition of Web service so far. There are descriptions and understandings of Web service by several large enterprises and institutions in the following.

The definition of the W3C organization: Web Service was a software application that used URI to unify the identification, and used XML to defined, described interfaces and binding. Web service is found and used by other users by network, and finished interacts through XML messages at last. The definition of SUN Company: Web service should include the following five characteristics. First, it provides an external interface, which exchange data in XML format. Second, the out Web service can be access by Web. Third, the services among the systems support relationship are loosely coupling. Fourth, if Web services completed registered and the services would be located. Fifth, it supports the specification of Web service protocol and implemented message communication used XML. The definition of IBM Company: Web service is the smallest

Z. Duan et al. (Eds.): SOFL+MSVL 2018, LNCS 11392, pp. 39–51, 2019.
https://doi.org/10.1007/978-3-030-13651-2_3

application module that has the characteristics of self-description, self-contained and support matched with other Web service. Web service can implement description, search, publish and be called anywhere in the network environment. Whether service users asked for simple application requests or complex composite business processes, it can accomplish tasks by calling the Web Service. When a Web service deployed successfully, any other application can be discover and invoke deployed service through a UDDI service registry to accomplish the task. The definition of HP Company: Web service is a service that solves user's problems through Internet, and transacted and processed tasks on behalf of applications and users.

Web services are platform independent, low coupled, self-contained and programmable. Web applications that describe, publish, discover, coordinate, and configure them using an open XML (a subset of the standard General Markup Language) for developed distributed interoperable application. The rise of Web service has been accompanied by the introduction of service-oriented software architecture (SOA), which provided a new paradigm of standards-based, loosely coupled, cross-platform distributed computing on the Internet. Individual Web service provides specific capabilities, and in order to meet the needs of users, more and more real projects need to integrate and combine multiple Web services to provide comprehensive and complex value-added services composite Web services. Members of the service can communicate with each other and handle user operations and requests in a logical manner. With the further development of Internet application, Web service composition is bound to get concerned and applied widely. In order to ensure the quality of composite Web services, model checking is used to verify the conformance and the related properties of the model of composited Web services with its implementation [1, 2]. Comprehensive and adequate testing for the implementation of Web services composition is required. However, because of the dynamic and distributed characteristics of the Web service composition itself, many traditional software-testing technologies have lost their original effectiveness to the Web service composition. Therefore, we need to study the new testing techniques and methods for the Web service composition, to provide a powerful support for the performance, function and reliability of composite service.

At present, Web service portfolio testing has been studied and some research results have been obtained. The purpose of this paper is to systematically summarize and analyze the existing methods and techniques for the testing of Web services composition. Although some researchers have made a definite analysis and discussion about this problem, we think that this problem is still needed further investigating. Web services testing analyzed and summarized by Hong, Bozkurt and Ebrahim [2–5], however, the research status of Web services composition testing had not been emphasized. Web services composition testing has discussed by categorized the test methods completed by Rusli et al. [6] in long before. After that, there were other study results have been published. Therefore, it is necessary to make a new and comprehensive survey summary.

The structure of this paper is as follows. The first section, introduces the definition of web services in detail; The second section, analyzes and discusses Web service architecture; The third section summarizes and describes several web service

composition testing methods; The fourth section, introduces and summarizes testing techniques about Web service composition; The last section summarizes and prospects about the research on web service composition testing.

2 Analyzed and Studied the Web Service Architecture

In general, Web Service used to invoke remotely. The traditional software testing technology could not be simply apply to measure tested work of Web service application system; In view of this, Zhang and Zhang [7] proposed a criteria contain J attribute indicators such as accuracy, fault tolerance and testability, which can be used to evaluate the reliability of Web services application systems. According to this reliability criterion, we can effectively eliminate inappropriate Web services.

Testing framework [8] based on JUnit used in unit testing of application system widely. Therefore, Zhang et al. [9] had attempted to propose a suitable unit test framework for BPEL, which included Composition Model, test architecture, life cycle management and so on. In this framework, the test function divided into several test process (TP) and control TP process (CP), with the life cycle of TP be controlled by TP provided beginTest and endTest.

Dong et al. [10] put forward an automated testing framework based on WSDL. Given that the message contained in WSDL didn't fully assist the test work, so the WSDL extension specification was referenced in this framework, it included four other extensions such as into/output (I/O) dependency. According to these extensibility, this test framework could deduce test data and operation flow, formed a complete test case at last.

Akehurst et al. [11] defined constraint for each object in BPEL based on the Object Constraint Language (OCL), and implement the Java classes of verification based on these restrictions. At the same time, Akehurst established a Meta-model based on specification of BPEL, and the associations in objects defined by BPEL were represented as UML diagrams.

Looker et al. [12] put forwarded a test method based on Fault-Injection. Because of the SOAP packet format used by Web services to exchange messages was based on XML, Looker was able to add an injector server between the service provider and the service requester by modified the container of the Web service, to monitor all messages exchange between the service provider and the service requester, and according to the setting of test cases insert a message that might cause an error into a normal message, and observed whether the Web service under tested could correctly correspond to messages with exceptions, such as error content, missing content, and so on.

Offutt et al. [13] proposed to use data disruption to generated different SOAP parameter data, and analyzed the messages in response to verified correctness of the peer-to-peer Web service. Offutt et al. [13] proposed three methods of disruption message: Data Value Perturbation (DVP), RPC Communication Perturbation (RCP), and Data Communication Perturbation (DCP). DVP was mainly based on the parameter message format defined by WSDL, such as string or numeric value, through the method of boundary value analysis to generate different parameter messages. RCP used mutation operators to calculated parameters to generate different new parameter

messages. DCP is a Web services testing that used XML messages to send messages. Offutt et al. [11] modified the contents of XML messages in SOAP by some rules, These XML messages are used to test the access to a Web service's database whether correct or not.

Chen, Li and Zhang [14] proposed a development and test environment that could flexibly define the process – WSCE, which enables the combination of Web services to carry out in a very convenient way. In this architecture, Yu proposed two mechanisms such as virtual partner and inspector service, to help developers verify the correctness of the process or not.

Tsai and Paul et al. [15] put forwarded a test framework of WSTF (Web Services Testing Framework). This framework was based on agent technology and could be applied to SOA architecture system.

3 Web Service Composition Testing Methods

Web services testing methods had many similarities to traditional software testing and there are differences of them. Web service testing required service requesters, service providers, and UDDI accomplished together. The comparative results between Web service testing and traditional software testing were shown in Table 1.

Compared with the traditional software testing and the characteristics of the Web service composition itself, the tester could not have all the test information, because of the component service is black box test, the tester could not have the source code of the component service, so it was unable to get all the features of a component service and build a rich test model. Therefore, in the test of Web service composition, some scholars have studied how to expand document parsing or build model technology to obtain sufficient test information. Furthermore, the dynamic binding of Web service composition makes it difficult to predict the operating environment and behavior of the combined service, the generation of test prediction was difficult, however, traditional software testing techniques target and software behavior were predictable, static, and non-distributed, so it could not be applied to Web service composition testing. As with traditional software, the compositional Web service also has a software evolution process, but the evolution process was dynamic, the changes didn't limited to internal structure or variable of the program. Instead, component services are upgraded or replaced, business processes are replaced, and interface information for component services is changed, and these evolutionary processes exist throughout the operation of the system, so additional information is needed to support the regression test. As we could see from the comparison above, the difference between Web service comparison testing and traditional software testing exists in the whole process of testing. According to the testing process, this paper summarized and analyzed the technologies and methods of Web composite service testing in different stages.

Table 1. Traditional software testing and web service composition testing

Item	Traditional software testing	Web services composition testing
Testers	Dedicated test team or software developer	Service integrator
Regression testing	Offline, static evolution; can sufficient understand software changes and regression testing timely	Online; dynamic evolution; difficult to grasp the evolve situation of component services, and there will also be evolution in the process of regression testing; additional information is needed to support regression testing
Software evolution	Static evolution, changed internal structure or variables of program	Dynamic evolution, component services upgraded or replaced, business processes changed, and component service interface information changed
Test client	Software itself	Built component service, such as proxy, etc.
Test coverage	White-box testing and black-box testing for software	Black-box testing for component services; white-box testing for BPEL documents
Test distribution	Centralized, multi-stage testing	Distribute, remote, multi-stage testing
Test execution	Off-line test	Runtime test
Test model	Have software code and could build rich test model according to software characteristic	Do not have source code of component services, testers could build controlled and observable test models only
Test prediction	The behavior of software is predictable, and it is easier to generate test predictions	It is difficult to predict state and behavior of composite services and generate test predictions difficultly
Test type	Unit test, integration test, system test, acceptance test, regression test, etc.	Unit test, integration test, system test, regression test

4 Web Service Composition Testing Techniques

One of the major features of Web services testing is that most testers do not gain the source code. Therefore, all white-box testing relevant techniques are not used. How to test Web services effectively is becoming a hot issue in Web services research.

4.1 Web Service Composition Testing Based on EH-CPN

The Web composite service testing technology based on EH-CPN mainly have the following steps: First of all, we analyzed the data flow in the extended colored Petri net through OWL-S document transformation, and find the used pairs for all of the variable definitions and the used chains for the definitions of corresponding input and output. Next, the definition that all input and output used chain extended to get an executed test sequence. The above test sequences meet the full definition of use coverage criteria. The test data was generated by the test case generated method that combined the

equivalence class partition and condition constraint, then combined test data and test sequences to generated test cases. Input the OWL-S document in the developed test tool of TWCS, and TWCS would generated a colored Petri net that extended levels corresponding to OWL-S, then find out used chains that all the input and output of all variables and extend it to an executable test sequence. Input the number of test cases corresponding to each test sequence, and completed the generated the number of test cases was same. Using the proxy occupancy program in .NET Framework SDK to generated proxy for each Web service; finally, the proxy service be used to completed the call of the corresponding Web service, executed the test case and completed the test of the Web service.

4.2 Web Service Composition Testing Based on Mutation

The idea of mutation testing was to detect the effectiveness of test cases by embedded errors in the program and guided the generation, selection and reduction of test cases, and to achieve the purpose of tested at last. It was a test method based error. The idea of mutation testing was proposed based on white box testing first, and the object of mutation testing was program code. With the development of mutation testing, the idea of mutation testing could be using for Web services testing, and guided test cases generation, selection and reduction of test cases [16].

There were a lot of research in Web service mutation test [17]; this section introduced the mutation-based Web service composition testing, which takes following steps to test the Web composite service workflow: Firstly, parsed OWL-S document and extracted information such as the type, format, and etc. input format accepted by the composite Web service to be tested and workflow. Generated the initial test data set based on the type, format, and other information of the input format, and test data could be generated randomly or by the usual methods of boundary value analysis or equivalence class division. At the same time, the workflow information of Web composite service is analyzed and found nodes that meet the variation conditions in the workflow. According to the corresponding change rules, changed the OWL-S document information to generate new mutant that injected the wrong to the original composite service to be wrong version of service composition.

After completed the above work, entered the same test data should be returned different test results executed the original service composition and the variant Web service composition, because of the workflow of the service composition was changed. If the output was different, the variant was killed and the test data could identify the wrong Web service combination and it was an effective test data. The test data should be retained and added an effective test data set for later test data selection. It indicated that the test data is invalid if the output results were same, the new test data should be redesigned or repeated above tests and expanded the effective test data set [18].

4.3 Web Services Testing Based on Interactive Behavior Specifications

This section introduces test problem of service interaction from the service requester's point of view. A Web service tested method that leverages interaction behavior specification. The main steps were as follows:

Firstly, the behavior specification of Web service should be described correctly. UML is widely used in system modeling in industry and academia because its simplicity and standardization, using sequence diagrams of UML and the behavior rules of Web services described by OCL. It is provided to the service requestor in the form of an XML file (XML metadata interchange) together with WSDL document, and XML documents could be generated automatically by existed UML modeling tools.

Secondly, used an extended state machine model of ELTS described the behavior rules of the service. Added semantic information based on traditional LTS generated extended ELTS to strengthen the function of LTS described data flows. ELTS was based on the implementation relationship in a certain formal defined, and introduced a new implementation relationship through generated algorithm of traditional LTS. Through this new relationship given corresponding test data generated algorithm and test cases with test coverage. It used to test whether the interaction behavior specification was consistent with service implementation or not [19].

4.4 Testing Techniques Based on Formal Methods

Test case generated based on formal method divided into model detector and formal analysis technology. The test case generation method based on model detector was an input model, which converts the service combination described of BPEL into a certain detector, and used formal method to describe the demand model that the composite service should satisfy, used them as input of the model detector to produces test cases. Most of the detectors used in the study included Nu SMV, SPIN and BLAST [20–23]. Test case generated method based on formalized analysis technology, which described by BPEL through a formal method or other formalization methods, such as Petri nets, automata or process algebra, then use existed analytical techniques of the formalized method to generated test cases, such as references [24–26].

Petri net was a modeling and analysis tool for distributed systems. It was a directed graph composed of repository, change, and directed arc. It was easy to described sequence, concurrency, conflict and synchronization of the processes and components in the system. Compared with other system models, true concurrency was a unique advantage of Petri network. The modeling method based on Petri net could described all kinds of control flow in the combinatorial process, but it could not reflect overall state of composited service directly.

Automaton was a mathematical model with clear semantics. It was suitable for describing discrete input and output systems. The system has a limited state, different states represent different meanings. In actual needs, the system could complete prescribed tasks in different states and transfer to another state. The automaton modeling method could described internal state of Web service composition directly, but it could not described interaction behavior between two services. The ability to described

concurrent activities in composite process was limited, and there was a space explosion problem. Process algebra was a formal modeling language based on algebraic methods. A group of operators was used as a process component in grammar. The semantics of the operator was defined by a structured operating semantic method. In this way, a process could be regarded as label transition system (LTS). A significant feature of process algebra was attributed concurrency to non-deterministic, that is, considered the behavior of concurrent processes as all possible interlace behavior of each process. The behavior of concurrent execution was suitable for described concurrent interactive systems. The modeling method of process algebra had strong description ability and rigorous computational reasoning ability, but its expression was more complicated, not intuitive and difficult to understand.

Miao and Chen et al. [27] proposed a testing approach to model-based testing for Web applications, which designed and implemented a web application testing system based on this model. Taking the UML state diagram of Web application as system test model, used UML sequence diagram described test target, and the FSM test model is constructed by transformation and combination, automate generate test case; test model visualization and automation of test execution were come true. It mainly focused on functional testing in article: Model-Based Testing for Web Applications. The performance tests, load testing, usability testing, compatibility testing and security testing not verified.

Qian and Miao et al. [28] proposed a test path generate approach, which illustrated by SWLS (Simple Web Login System) as an example and presented an effective Web testing model for Web software testing. One of the main advantages of this approach is that you did not need to access back-end source code. In order to get PTT from PFD, they proposed a transformation algorithm by this method. They obtained test path from PTT by constructing path expression, and gave a possible way to describe test path in XML. Qian and Miao, which were full link coverage and full-page coverage, also proposed two important concepts. It is possible that a particular link will appear on a page only if provided a specific input in tested. But this web test method is not necessarily adaptable in new case, so it needs to further improve test path generation method, and develop new prototype to re-validate this web test model proposed.

Above several kinds of formal model could described the behavior of the Web service combination well and have relevant technical and tools support, there were some differences only in computational complexity. However, these methods required staff with relevant professional background knowledge and ignored data flow information modeling in combination process. Therefore, the non-formal test case generated method was discussed below.

4.5 Testing Techniques Based on Informal Methods

The informal method [29–51] that it converted control flow, data flow, message flow, behavior, etc. in a composite service described by BPEL and others into a graph model and used search technology and constraint analysis technology generated test cases.

The steps of test case generated method as follows: Firstly, build a model based on some features of Web service composition. Secondly, generated test paths based on model traversal. Thirdly, generated test cases based on constraint condition in above path. The following research work falls into this category.

On studied of test case generated problem in Web service combination, some researchers focus on control flow characteristics of Web service composition. Yuan et al. [29] proposed a BPEL test case generation method based on graph search, which used to deal with concurrent semantics of BPEL. This method described the WS-BPEL program by defined control flow graph (BPEL flow graph, BFG). BFG contained control flow and data stream of BPEL program. Generated test paths by traversed the BFG model, the constraints in test path as the input of constraint parser generated abstract test data and converted it into executable test cases automatically. However, the process of converted BPEL documents to BFG and test paths search process were done manually.

The method proposed by Yan et al. [30] was similar to Yuan's method [29]. They converted the WS-BPEL program into an extended control flow graph (XCFG) and generated test paths based on XCFG, and then used a constraint parser generated test data from test path. However, it is different from the Yuan et al. method that Yan and others used symbolic execution methods to obtain a series of constraints from the test path by invoked component services, but this method produced abstract test case that it needed to be converted to an executable test case manually.

Mei and others proposed a test method based on XPath Rewrite Graph (XRG) [31, 32] that it combined the control flow graph (CFG) and XRG to solved possible integration problem caused by XPath in BPEL process. With the gradual deepening research, some scholars believed that the model based test cases produce techniques same as the method based on path generated test case. This method represents the test data by generated message parameters, but the generated test cases were high redundancy and low error rate. Hou [33] and Ni et al. [34] applied test technology based on message flow in Object Oriented Program (OOP) to Web service combination test first time. Wu and Huang [35] thought that binding internal state of single service, execution sequence among services and behavior of service closely related in runtime. Therefore, references [36–38] proposed an EDSM sequence test model (EFSM-SeTM) for Web service composition. They studied runtime test from the point of workflow view and proposed a scenario-based testing framework for Web service composition.

We summarized and analyzed informal testing method based on model. As shown in Table 2. It is shown that most of above test case generation techniques are semi-automated, even include the technology proposed in running test. Therefore, firstly, How to achieve full automation is a problem that can be further studied. Secondly, no test case generated technology involved Web services and the quality of Web services determined the correctness of entire Web service composition, so it is necessary to test the Web service.

Table 2. Classification of informal methods

References	Model	Focus	Quality of test case	Type	Automated testing
Reference [34]	Message sequence graph	Message flow	Test case accurate, error detect capability low; high redundant	Runtime test	Semi-automatization
				Runtime test	
Reference [35]	State transition diagram; message exchange sequence diagram	Service interchange and dynamic behavior	Without considering the constraint conditions in the path; test case with practical significance can not be obtained	Runtime test	Semi-automatization
Reference [36]	BPEL model	Scene-based	Testing is only based on path; test cases inaccurate; high redundant; error detect capability low	Runtime test	Automation
Reference [37]				Runtime test	Automation
Reference [38]				Runtime test	Automation
Reference [30]	Extended control flow graph	Control flow	Abstract test cases	Static test	Semi-automatization
Reference [31]	Rewrite the graph of Xpath	Data flow	Abstract test cases	Static test	Semi-automatization
Reference [32]			Abstract test cases	Static test	Semi-automatization
Reference [33]	Message sequence graph	Message flow	Higher detection rate than RAND and GS; redundancy is higher	Static test	Semi-automatization

5 Summary and Outlook

It can be known from above analysis that some problem of Web service combination need to be research, although some results have been achieved in this area, which were mainly reflected in the following aspects.

Formal modeling technology of Web services combination need to be developed and researched deeply, such as research on the correlation, fairness and applicability of formal model, further research on the property analysis technology of Web services formal model. It is necessary to research Web services technologies made formal technology provide services and support better.

The quality of test cases was fundamental condition for effective Web service combination testing, which shown that high error rate, low redundancy and high coverage. How to obtain more constraints that can enrich test information generating test cases is a problem needed to be solved in academia and industry today.

Some researchers have been concerned about the runtime binding problem of Web service composition, Ni [34], Wu [35] and Sun et al. [38] designed test automation prototype tools, but they did not elaborated automation level of test technology, or did they verify the relationship between automation and runtime binding issues.

Acknowledgement. This paper is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61572306.

References

1. Duan, Z., Yang, X., Koutny, M.: Framed temporal logic programming. Sci. Comput. Program. **70**(1), 31–61 (2008)
2. Tian, C., Duan, Z., Duan, Z.: Making CEGAR more efficient in software model checking. IEEE Trans. Softw. Eng. **40**(12), 1206–1223 (2014)
3. Hong, Z., Feng, Z.Y.: Collaborative testing of web services. IEEE Trans. Serv. Comput. **5**(1), 116–130 (2012)
4. Bozkurt, M., Harman, M., Hassoun, Y.: Testing and verification in service-oriented architecture: a survey. Softw. Test. Verif. Reliab. **23**(4), 261–313 (2013)
5. Ebrahim, S.M.: A survey of service-oriented architecture systems testing. J. Softw. Eng. Appl. (IJSEA) **3**(6), 19–27 (2012)
6. Rusli, H.M., Puteg, M., Ibrahim, S., Tabatabaei, S.: A comparative evaluation of state-of-the-art web service composition testing approaches. In: Proceedings of the 6th International Workshop on Automation of Software Test (AST), pp. 29–35 (2011)
7. Zhang, J., Zhang, L.-J.: Criteria analysis and validation of the reliability of web services-oriented systems. In: Proceedings of the IEEE International Conference on Web Services, pp. 11–15 (2005)
8. Toure, F., Badri, M., Lamontagne, L.: A metrics suite for JUnit test code: a multiple case study on open source software. J. Softw. Eng. Res. Dev. **2**(1), 1–32 (2014)
9. Zhang, X., Sun, W., Jiang, Z.B.: BPEIAWS unit testing: framework and implementation. In: Proceedings of the IEEE International Conference on Web Services, pp. 103–110 (2005)
10. Dong, W., Tasi, W.T., Chen, Y.: WSDL-based automatic test case generation for web services testing. In: IEEE International Workshop, pp. 2l5–220 (2005)
11. Akehurst, D.H.: Validating BPEL specifications using OCL. Technical report, University of Kent at Canterbury (2004)
12. Looker, N., Xu, J.: Assessing the dependability of SOAP RPC-based web services by fault injection. In: The Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 165–172 (2003)
13. Offutt, J., Xu, W.: Generating test cases for web services using data perturbation. In: ACM SIGSOFT Software Engineering Notes, vol. 29, pp. 1–10 (2004)
14. Chen, Y., Li, Y., Zhang, L.: WSCE: a flexible web service composition environment. In: Proceedings of the IEEE International Conference on Web Services, pp. 428–435 (2004)
15. Tsai, W.T., Paul, R., Yu, L., Saimi, A.: Scenario-based web service testing with distributed agents. IEICE Trans. Inf. Syst. **86**, 2130–2144 (2003)
16. Jiang, Y.: Research on web service workflow variation test technology. Southeast University (2011)
17. Wang, R., Huang, N.: Requirement model-based mutation testing for web service. In: Proceedings of the 4th International Conference on Next Generation Web Services Practices, pp. 71–76 (2008)

18. Bruno, M., Canfora, G., Di Penta, M., Esosito, G., Mazza, V.: Using test cases as contract to ensure service compliance across releases. In: The 3rd IEEE International Conference on Service-Oriented Computing, Amsterdam, Netherlands (2005)
19. Li, B., Zhang, P.: Modeling. Testing and Verification of Combined Services. Science Press, Henderson (2013)
20. Huang, H., Tsai, W.T., Paul, R.: Automated model checking and testing for composite web services. In: Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 300–307 (2005)
21. Garcia-Fanjul, J., de La Riva, C., Tuya, J.: Generating test cases specifications for BPEL compositions of web services using SPIN. In: Proceedings of WS-MaTe 2006, pp. 83–94 (2006)
22. Garcia-Fanjul, J., de La Riva, C., Tuya, J.: Generation of conformance test suites for compositions of web services using model checking. In: Testing: Academic and Industrial Conference - Practice and Research Techniques, pp. 127–130 (2006)
23. Zheng, Y.Y., Zhou, J., Krause, P.: A model checking based test case generation framework for web services. In: Proceedings of the International Conference on Information Technology, pp. 715–722 (2007)
24. Li, B., Zhang, W.S., Zhang, X.G.: Describing and verifying web service using CCS. In: Proceedings of the International Conference on Parallel and Distributed Computing, pp. 1571–1576 (2006)
25. Long, H.Y., Ma, D.: Checking compatibility of BPEL4WS based on CCS. In: Proceedings of the International Conference on System Science, Engineering Design and Manufacturing Informatization, pp. 255–258 (2011)
26. Du, Y.H., Tan, W., Zhou, M.C.: Timed compatibility analysis of web service composition a modular approach based on Petri nets. IEEE Trans. Autom. Sci. Eng. **11**(2), 594–606 (2014)
27. Miao, H.-K., Chen, S.-B., Zeng, H.-W.: Model-based testing for web applications. Chin. J. Comput. **34**(06), 1012–1028 (2011)
28. Qian, Z., Miao, H.: Efficient web software testing method. Comput. Sci. **38**(02), 152–155+159 (2011)
29. Yuan, Y., Li, Z., Sun, W.: A graph-search based approach to BPEL4WS test generation. In: Proceedings of the International Conference on Software Engineering Advances (ICSEA), pp. 1–14 (2006)
30. Yan, J., Li, Z., Yuan, Y., Sun, W., Zhang, J.: BPEL4WS unit testing: test case generation using a concurrent path analysis approach. In: Proceedings of the 17th International Symposium on Software Reliability Engineering (ISSRE), pp. 75–84 (2006)
31. Mei, L., Chan, W.K., Tse, T.H.: Data flow testing of service choreography. In: Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC), pp. 151–160 (2009)
32. Mei, L., Chan, W.K., Tse, T.H.: Data flow testing of service oriented workflow applications. In: Proceedings of the 30th International Conference on Software Engineering (ICSE), pp. 371–380 (2008)
33. Hou, S.S., Zhang, L., Lan, Q., Mei, H.J., Sun, S.: Generating effective test sequences for BPEL testing. In: Proceedings of the 5th International Conference on Quality Software, pp. 331–340 (2009)
34. Ni, Y., Hou, S.S., Zhang, L., Zhu, J., Li, Z.J., Lan, Q.: Effective message-sequence generation or testing BPEL programs. IEEE Trans. Serv. Comput. **6**(1), 7–19 (2013). https://doi.org/10.1109/TSC.2011.22

35. Wu, C.S., Huang, C.H.: The web services composition testing based on extended finite state machine and UML model. In: Proceedings of the IEEE International Conference on Service Science and Innovation, pp. 215–222 (2013)
36. Sun, C.A., Shang, Y., Zhao, Y., Chen, T.Y.: Scenario-oriented testing for web service compositions using BPEL. In: Proceedings of the 12th International Conference on Quality Software (QSIC), pp. 171–174 (2012)
37. Zhang, P.C., Leung, H., Li, W.R., Li, X.D.: Web services property sequence chart monitor: a tool chain for monitoring BPEL-based web service composition with scenario-based specifications. IET Softw. **7**(4), 222–248 (2013)
38. Sun, C., Zhao, Y., Pan, L., Liu, H., Chen, T.Y.: Automated testing of WS-BPEL service compositions: a scenario-oriented approach. IEEE Trans. Serv. Comput. **11**, 616–629 (2015)
39. Li, Q., et al.: Service composition and interaction in a SOC middleware supporting separation of concerns with flows and views. J. Database Manag. (JDM) **22**(2), 32–63 (2011)
40. Belli, F., Endo, A.T., Linschulte, M., Simao, A.: A holistic approach to model-based testing of web service compositions. Softw.: Pract. Exp. **44**(2), 201–234 (2014)
41. Herbold, S., Harms, P., Grabowski, J.: Combining usage-based and model-based testing for service-oriented architectures in the industrial practice. Int. J. Softw. Tools Technol. Transfer **19**(3), 309–324 (2017)
42. Chen, C., Zaidman, A., Gross, H.G.: A framework-based runtime monitoring approach for service-oriented software systems. In: Proceedings of the International Workshop on Quality Assurance for Service-Based Applications, QASBA 2011, pp. 17–20. ACM, New York (2011)
43. Gao, H., Li, Y.: Generating quantitative test cases for probabilistic timed web service composition. In: Proceedings of the APSCC, pp. 275–283 (2011)
44. Hallé, S., La Chance, E., Gaboury, S.: Graph methods for generating test cases with universal and existential constraints. In: El-Fakih, K., Barlas, G., Yevtushenko, N. (eds.) ICTSS 2015. LNCS, vol. 9447, pp. 55–70. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25945-1_4
45. Elqortobi, M., Bentahar, J., Dssouli, R.: Framework for dynamic web services composition guided by live testing. In: Belqasmi, F., Harroud, H., Agueh, M., Dssouli, R., Kamoun, F. (eds.) AFRICATEK 2017. LNICST, vol. 206, pp. 129–139. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67837-5_13
46. Mei, L., Cai, Y., Jia, C., Jiang, B., Chan, W.K.: Test pair selection for test case prioritization in regression testing for WS-BPEL programs (Report). Int. J. Web Serv. Res. **10**(1), 73(30) (2013)
47. Petrova-Antonova, D., Ilieva, S., Manova, D.: TASSA: testing framework for web service orchestrations. In: 2015 IEEE/ACM 10th International Workshop on Automation of Software Test, pp. 8–12, May 2015
48. Cao, D., Félix, P., Castanet, R.: WSOFT: an automatic testing tool for web services composition. In: 5th International Conference on Internet and Web Applications and Services (2014)
49. Xu, C., Qu, W., Wang, H., Wang, Z., Ban, X.: A Petri Net-based method for data validation of web services composition. In: 2010 IEEE 34th Annual Computer Software and Applications Conference (COMPSAC), pp. 468–476, July 2010
50. Zhang, J., Yang, R., Chen, Z., Zhao, Z., Xu, B.: Automated EFSM-based test case generation with scatter search. In: Proceedings of the 7th International Workshop on Automation of Software Test, 02 June 2012, pp. 76–82 (2012)
51. Shan, N.: Applications research in ultrasonic testing of carbon fiber composite based on an optical fiber F-p sensor. In: Proceedings of SPIE - The International Society for Optical Engineering, 25 October 2016, vol. 9685, pp. 968511–968511-6 (2016)