

Cooperative and Group Testing in Verification of Dynamic Composite Web Services

W. T. Tsai, Y. Chen, R. Paul*, N. Liao, and H. Huang

Department of Computer Science and Engineering

Arizona State University

Tempe AZ, 85287 USA

*Department of Defense, Washington DC

Abstract

Verifying Web Services (WS) in a dynamic Service Oriented Architecture (SOA) is challenging because new services can be composed at runtime using existing WS. Furthermore, in a composite service, any component can be dynamically replaced during execution if the component fails. Another challenge is that the testing is time critical because verification must be conducted at runtime and in real time. This paper compares and contrasts traditional software testing and WS testing techniques and proposes WS group testing technique¹ to test composite services. The group testing technique also has the ability to evaluate the test scripts, automatically establish the oracle of the each test script, and identify faulty WS in a failed composite WS.

Keywords: Web services, service-oriented architecture, collaborative verification and validation.

1. Introduction

A Service-Oriented Architecture (SOA) considers a system consists of a collection of loosely coupled Web Services (WS) [3][6][9]. These services can make use of each other services to achieve their own desired goals and end results. Simple services can cooperate in this way to form a composite service.

From software development point of view, the SOA and WS present a paradigm shift in the entire software development process, including specification, design,

implementation, testing, verification, validation, and reliability modeling.

Recently, WS testing has received significant attention and a variety of techniques have been proposed including functional testing, SOAP testing, interoperability testing, security testing, scalability testing, regression testing, distributed testing, performance testing, stress testing, runtime and message monitoring, UDDI-based testing, integration testing, generating test scripts/cases from WSDL, WS check-in and check-out mechanisms, and reliability modeling [1][5][8][10].

Regardless what testing technique is used, one key concept emerges, i.e., it is no longer valid to use traditional IV&V (Independent Verification and Validation). Instead, Collaborative Verification and Validation (CV&V) is used, i.e., all the parties of WS, including providers, brokers such as UDDI servers, and clients must collaborate and cooperate to perform WS testing to ensure trustworthy computing.

2. IV&V versus CV&V

This section compares and contrasts the CV&V and traditional IV&V as summarized in Table 1. As can be seen from the table, traditional IV&V is often performed statically and off-line. Although traditional testing techniques can be applied to WS, a significant portion of WS testing must be done dynamically and in real time [11][12]. Such dynamic and just-in-time testing and verification challenge the traditional testing theory and practice.

The CV&V model can be used to construct a trustworthy service broker [2][11] based on a UDDI server. A service provider or its independent testing team will still perform IV&V testing during the development of services. When the service provider registers a service to a trustworthy service broker, e.g., an extended UDDI

¹ Patent is pending on WS group testing including voting and ranking of WS, ranking of test cases/scripts, and oracle establishment.

server [11], it must provide a set of sample test scripts and test data. The trustworthy server performs check-in and check-out testing to ensure that the clients receive trustworthy services. When checking-in or registering a service, the server will use the provided test scripts and a set of its own independent set of test scripts to test the service rigorously to ensure the dependability and quality

of the services. The server does not need to copy the code of the service program into the server. The service provider will continue to maintain the code. If a registered service is modified, the service provider is required to inform the trustworthy server to perform another check-in test.

Verification and Validation	Traditional IV&V	Service-Oriented CV&V
Approach	Verification team is independent of development team to ensure objectivity and completeness. Testing is still mostly done by software providers.	Verification is done by collaboration among service providers, clients and independent service brokers. The emphases are on real time and just-in-time testing, and evaluation using data dynamically collected during testing.
Testing location	Centralized multi-phase testing.	Distributed, remote, multi-agent and multi-phase testing.
Operational testing	Off-line field testing.	On-line just-in-time testing
Regression testing	Off-line regression testing.	On-line regression testing using data dynamically collected.
Integration testing	Static unit testing and Integration testing.	Dynamic configuration and integration testing.
Testing coverage	Structural (white-box) or functional (black-box) coverage.	Service providers can have traditional coverage, service brokers and clients may have black-box (such as WSDL description) coverage analysis only.
Test case profiling	Static profiling	Dynamic profiling.
Reliability modeling	Input domain-based and reliability growth.	Reliability models based on dynamic profiles and group testing
Certification	Static certification center.	Dynamic certification based on service history.
Test script repository	Statically maintained repository.	Dynamically expanding repository.
Model Checking	Static model checking.	Just-in-time dynamic model checking or policy enforcement.

Table 1. Testing techniques applied in traditional software testing and service-oriented software testing

When checking-out (offering a service to a client), the UDDI server will perform check-out testing to ensure that the service has not been modified since its last check-in. If it has been modified, the rigorous check-in testing must be performed.

Before a client actually uses a service, it can request an acceptance test. The client can use the trustworthy UDDI server-provided test scripts and/or its own test scripts. Normally, a client will receive multiple matches of its requests and the client can test the services and choose the best deal. In [11], several mechanisms were proposed to address this trial process.

CV&V also extends the idea that testing is to match the client's needs with services. It makes it possible to have trustworthy UDDI server as a notary public. More specifically, a trustworthy UDDI server can

- Define its own QA requirements and test scripts independently developed by third parties;
- Offer clients WS together with their quality rating and profiles;
- Make its QA requirements and test scripts public so that service providers can check their software before submitted them for registration.
- Involve multiple distributed test agents in executing test commands, monitoring the progress including testing, participating in scalability testing, and collecting test results.

3. Group Testing in CV&V

Besides applying the existing software testing techniques dynamically to WS testing, specific testing

techniques can be developed to improve WS testing. Group testing is an example of such specific technique.

The SOA allows a complex WS to be composed from other WS dynamically and at runtime. The question is, how can a *composite* service, its unit services, and their interoperability be tested in real time? A group testing

technique, originally developed for blood testing and later for software regression testing [11], is one candidate solution. However, blood group testing (BGT) is different from WS Group Testing (WSGT), and Table 2 lists the similarities and differences between them.

Compared features	Blood Group Testing (BGT)	Web Services Group Testing (WSGT)
Testing goals	Find bad samples from a large pool of blood samples.	1) Rank WS in a large pool of WS; 2) Rank the fault detection capacity of test scripts; 3) Determine the oracle of each test case; and 4) Fault identification.
Optimization objectives	Minimize the number of tests needed.	Minimize the number of tests and voting needed.
Sample mix	Arbitrary and physical mix.	Interoperability is constrained by WSDL and composition semantics such as ontology.
Testing methods	Bio/chemical tests.	WS unit, integration, and interoperability testing using adaptive, progressive, and parallel testing.
Testing location	Centralized testing	Distributed and remote testing by agents and voters.
Verification	Contamination analysis.	Oracle comparison and/or majority voting
Test coverage	One test for each mix.	Need <i>many</i> tests for each group of WS to verify a variety of aspects.
Reliability evaluation	Reliability of testing process	Reliability of WS under test and testing process
Reliability of tests	Tests can be reliable or unreliable. Most BGT assumes tests are reliable.	The voting mechanism may be unreliable, and the number of faulty WS may be greater than the number correct WS to mislead the voter.

Table 2. Comparing Blood and WS Group Testing

Now we present how WSGT is applied in integration testing of composite WS. Assume there are n sets of unit WS and Set_i have m_i equivalent unit WS: $Set_1 = \{S_{11}, S_{12}, \dots, S_{1m_1}\}$, $Set_2 = \{S_{21}, S_{22}, \dots, S_{2m_2}\}$, ..., $Set_n = \{S_{n1}, S_{n2}, \dots, S_{nm_n}\}$. A composite WS can be composed using one unit WS out of each set. Then, there are totally $m_1 * m_2 * \dots * m_n$ possible composite WS. For example, a composite WS is a supply-chain system and it is composed of six unit WS: *ordering*, *retail*, *payment*, *transportation*, *lodging*, and *insurance*. Assume there are 10 alternative WS for each unit service, the total number of possible composite WS is 10^6 . Figure 1 outlines the WSGT mechanism. The inputs to the composite WS are broadcast to all composite services WS_1, WS_2, \dots, WS_k to test. Their outputs are directed to a voting service, which synchronizes its inputs and selects the majority based on a weighted voting algorithm. The weights are decided by the reliability history of the services. The outputs of each service are compared with the majority. The number of disagreements is dynamically logged into the profile of each service and is used to evaluate and rank the reliability of the WS, effectiveness of test scripts, and establishment the oracle of each test case.

Various techniques can be used to restrict the participating unit WS and their combinations. Figure 2 shows the technique that applies group testing at the unit level. Using the supply chain as an example, it votes the outputs of multiple *ordering* services available and then applies the majority result to the next level, and so on. The voter at each level can detect the poor unit WS and eliminate them, and thus reduce the possible combinations of the composition.

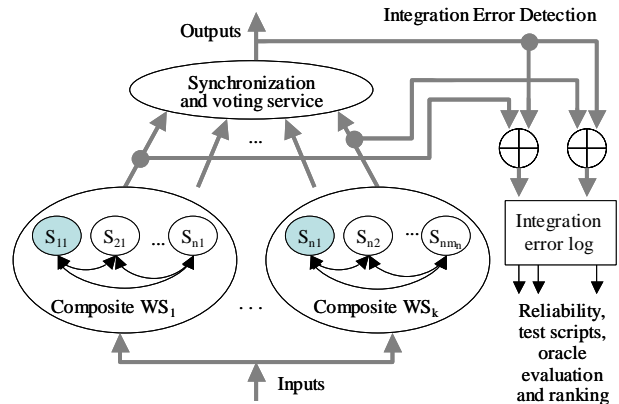


Figure 1. Group Testing of Multiple Composite WS

The WSGT at both composition and unit levels has the following features:

- One of the tough problems in software testing is to verify the correctness of the output for each input, or define an oracle for each input. In this WSGT, the voting service can automatically generate an oracle for each input according to the majority principle.
- The history of testing is taken into account and is used in the weights to form the majority of future testing.
- Testing of the alternative composite WS can be done in parallel or sequentially, depending on the computing power available. The synchronization mechanism in the voting service can identify the outputs corresponding to the same inputs. A remote test agent was proposed to take spare computers on the Internet to perform distributed testing [11].
- Normally, incorrect WS will generate different incorrect results and thus cannot form a false majority. This risk can be further minimized by running the test inputs with known oracles first and reject those WS with poor rating. The weights used in the voting mechanism also reduce the risk because the weights are adjusted after each round of WSGT to ensure that the best WS will have the greatest influence.
- WSGT can be used at each level of WS testing to rank the unit WS or composite WS at that level, and only the best WS have the right to compete at the next level to minimize the number of integration testing.

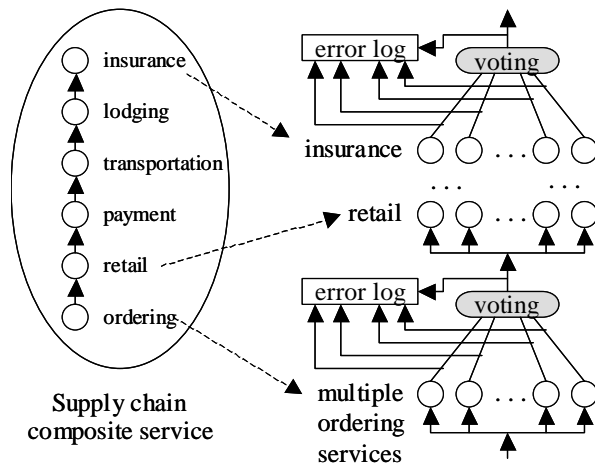


Figure 2. Apply Unit WSGT to Eliminate Poor WS

Another form of WSGT is also possible to identify the faulty participating WS once a composite WS is detected to be faulty. Suppose WS_1, WS_2, \dots, WS_k participated in a composite WS that failed a test, because multiple versions of WS are possible for each WS_1 to WS_k , and some earlier combinations of these WS has passed the

same test. The regression group testing proposed in [7], and traditional BGT techniques [4] can then be used to identify the faulty WS in the composite WS.

4. Summary and Conclusion

WS testing requires a new paradigm CV&V instead of IV&V. In the CV&V all the parties collaborate and cooperate to perform testing. This paper further proposes using group testing in verifying WS in real time by voting at both unit and integration levels. Like BGT, WSGT reduces the number of tests required to identify the faulty participating WS in a faulty composite WS. However, WSGT testing is different from BGT in many aspects. It can be used to rank the fault detection capability of test scripts and to establish the oracle for most test inputs, in addition to rank WS for application.

References

- [1] J. Bloomberg, "Web services testing: Beyond SOAP", ZapThink LLC, Sep 2002, <http://www.zapthink.com>
- [2] Y. Chen, "A Service Scheduler in a Trustworthy System", in proceedings of the 37th Annual Simulation Symposium, Arlington VA, April 2004, pp. 89-96.
- [3] B. De, "Web Services - Challenges and Solutions", WIPRO white paper, 2003, <http://www.wipro.com>.
- [4] D. Z. Du and F. Hwang, *Combinatorial Group Testing And Its Applications*, World Scientific, 2nd edition, 2000.
- [5] A. D. Gordon and R. Pucella, "Validating a Web Service Security Abstraction by Typing", Microsoft MSR-TR-2002-108, December 2002.
- [6] H. He, "What is Service-Oriented Architecture?" <http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>, Sept, 2003.
- [7] A. K. Onoma, W. T. Tsai, M. Poonawala, and H. Sukanuma, "Regression Testing in an Industrial Environment", *Communications of ACM*, Vol. 41, No. 5, 1998, pp. 81-86.
- [8] R. Sumra and R. Venkatvaradan, *Web Service's Test Harness: A Functional, Load, and Performance Testing Framework for Web Services*, <http://www.developer.com/services/article.php/2229161>.
- [9] A. S. Szalay, T. Budavári, T. Malik, Jim Gray, and Ani Thakar, "Web Services for the Virtual Observatory", Microsoft MSR-TR-2002-85, August 2002.
- [10] 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.
- [11] T. Tsai, R. Paul, Z. Cao, L. Yu, A. Saimi, and B. Xiao, "Verification of Web Services Using an Enhanced UDDI Server", *Proc. of IEEE WORDS*, 2003, pp. 131-138.
- [12] W. T. Tsai, R. Paul, L. Yu, A. Saimi, and Z. Cao, "Scenario-Based Web Service Testing with Distributed Agents", *IEICE Transactions on Information and Systems*, 2003, Vol. E86-D, No. 10, 2003, pp. 2130-2144.