Automatic Test Case Generation for Orchestration Languages at Service Oriented Architecture

Ebrahim Shamsoddin-Motlagh
Computer Engineering Department, Faculty of Engineering, Science & Research Branch, Islamic Azad University, Tehran, Iran

ABSTRACT
Service oriented architecture (SOA) is one of the latest software architectures. This architecture is created in direction of the business requirements and for removing the gap between softwares and businesses. The software testing is rising cost of activities in development software. SOA has different specifications and features proportion of the other software architectures. According to these features of the system, we cannot apply all approaches and methodologies of testing in the typical software systems for testing in the SOA systems, and there are need to specific procedures for testing the service oriented systems and/or change in the testing approaches.

This document presents an approach for test cases generation automatically at the SOA system. First, this approach creates a control flow graph of BPEL file in the system and services related of the main service, WSIG file is used to create subgraphs of the related services. Then, the test cases create randomly of the primary test for graph in the generated system. Final, it tries to create test cases require to cover of the system graph through randomly generation and the genetic algorithms. This algorithm will compare with standard genetic algorithm and we will show the algorithm has performance better than the other algorithm.

Keywords
Service Oriented Architecture, Software testing, automatic test case generation, SOA Testing

1. INTRODUCTION
Arasanjani, Borges and Holley define SOA as follows [1]: “SOA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions.” System services have features in the design and implementation, these features include: service reusability, standardized service contract, service loose coupling, service abstraction, service composition, service autonomy, service statelessness and service discoverability capabilities.

The SOA system has different nature and the specific characteristics than the traditional system of the test system; it's harder and needs more time. The test facilitates and abilities at the SOA system testing should be recognized and solution(s) should be presented for testing challenges. The key issues of testability limits of the SOA systems include [2]: "dynamicity and adaptiveness, lack of observability of the service code and structure, lack of control, lack of trust, new aspects of testing, test cost, different stakeholders".

This paper is structured as follows. Section 2 a review related work of automatic test case generation in SOA system. Section 3 discusses our approach for automatic test case generation in the SOA system. Finally, Section 4 outline suggests future research steps.

2. Related work
SOA system testing should be performed of aspects functional testing and non-functional testing, the functional testing has different levels, the levels include unit testing in individual services and the combined services, integration testing and regression testing.

Numbers of investigations in the unit testing have been active to test automation, in their attempted to automate process or processes of testing. The researches [3, 4, 5, 6, 7, 8, 9, 10] performed unit testing on the WSDL file automatically.

Numbers of existing researches [11, 12, 13, 14, 15] performed BPEL-based testing in the system with graph operations. In studies [16, 17] implemented test at combining web services used to high level Petri net and specifications BPEL. In the paper [18] presented an approach to generate test cases from web service automata (WSA) automatically [18]; WSA can be used to define the operational logic in BPEL.

Numbers of test frameworks have been prepared for SOA testing, than these are performed the SOA testing with the best way [19, 20]. The DFTT4CWS tools automatically find unusual data flow [21] and the test paths is generated data flow testing with all of cover criteria types. In the reference [22], the BPEL file is mapped DOM object tree to the EMF activities tree. The WebMov is set of tools modelling, evaluates and tests web services composition [23]. One paper was expressed computational strategy for the generation complete computational paths of BPEL based on Tabu search and genetic algorithms to generate test data [24]. The research [25] provided an approach to design test cases based on functional properties of high-level business process model. The study [26] proposed an approach for reducing the costs to test such applications,

and how can semantic stubs enable the client test suite to be partitioned into subsets, some of which don't use to execute remote services. Model driven approach is presented in [27], this approach to generate executable test cases from business processes. In the paper [28] provided a constraint based testing approach in the SOA system.
Researches [29, 30] are proposed an approach to determine the changes use to extensible BPEL flow graph (XBFG) of control flow and to compare the paths in a new version of service composition with the old version. Testing rules and monitoring rules include: checking the functional characteristics services, checking quality of service (QoS), checking interoperability services and service evolution [31]. Changing in SOA systems is big and need time, hence the research [32] attempted to propose a road map to regression testing in the SOA system.

Non-functional requirements include [33]: the needed data to fulfill the monitoring goal is intercepted. Monitoring mechanisms attempt the performance isn't influenced of unmonitored elements and performance is influenced of the monitored elements remains to be minimal. The changes responses are in the monitoring goal and environment topology. Instrumentation must be transparent and performed on demand. System security is one of the characteristics non-functional SOA systems. The paper [34] presented an approach towards an evaluation framework for SOA security testing tools. A research proposes a technique on how define reliability test of composite service in BPEL from the view at business semantics, it used to fault injection [6]. This paper focus on how the reliability problems find relate with business process, the called semantics as the problems are not pure coding error but faults related to business process. In addition, the behavior of composite services in BPEL is analyzed when there are faults in the orchestrated services invoked.

Numbers of existing produced tools was created to test SOA systems automatically. For example TASSA is a framework for automatic testing in functional and non-functional specifications of service-based applications [35]. It provides end-to-end testing of service layer, service composition and coordination and business process. Another tool is WSOTF presented for the automated testing [36]. WSOTF is an automatic conformance testing tool with timing constraints from a formal specification of web services composition that is implemented by an online testing algorithm. In the study [37] is expressed a test approach in BPEL web service composition described. The paper [38] is proposed to generate a testbed for SOA systems that takes into account a mobility model of nodes in the network which the accessed services are deployed. The study [39] is a framework and its supporting tool for automatically generating and executing web-service requests and analyzing the subsequent request-response pairs. The study [40] is proposed an approach to combine the accessibility technologies in graphical applications (GAPs) for a visualization mechanism enables nonprogrammers to generate unit test cases in web services by drag-and-drop operations on graphical user interface (GUI). In the reference [41] is testing techniques to generate a set of test cases for web services automatically. The techniques presented here explore data perturbation of web services messages upon data types, integrity and consistency.

In the paper [42] is expressed a survey to explore cloud services testing methods. The paper [43] is expressed a review to identify SOA testing researches with dynamic binding, that paper performed to search manually and automatically in journals, conferences and etc. In the papers [44, 2] are expressed a survey to SOA testing before 2012 year.

3. Proposed approach

As regards, the automated testing is one of the challenges in SOA testing and software testing is costly one of the development level, for automatic testing have been attempts of the software testing. Hence this section of the paper presents a proposed approach for automatic test cases generation in the SOA system.

At the beginning of this section describes a proposed approach and that implementation. Then how to development a proposed approach and test program delivers the test results.

3.1 Approach

Manually test cases generation and manually test operations is a difficult and time consuming, and the dynamic nature of SOA system causes the generated test cases lose their usability after some time. To resolve this problem needs to create a dynamic and automatic way to generate test cases in orchestration of SOA. Which results store a lot of cost and time in producing the system. To resolve this problem in this part of an approach is offered for automatic test cases generation of the WS-BPEL and related services.

Activities in the proposed approach to generate test cases automatically from the WS-BPEL file in the activity diagram is shown in Figure 1; this diagram has the following steps:

1) **Graph Generator:** WS-BPEL and WSDL files in the main service at the test system, the main web service and summons related services invoke graph files (WSIG) from the main service system will analyze, and the system control flow graph will provide.

2) **Generate Initial test cases:** If the initial test cases are not available, the test system will generate initial test cases randomly.

3) **Run Batch-Optimistic (BO) algorithm [45]:** Algorithm will call the BO genetic algorithm for better coverage of test cases.

4) **Run Close-Up (CU) algorithm [45]:** Algorithm will call the CU algorithm for nodes not covered by the test cases.

5) **Remove repeat:** Some generated test cases are similar coverage, the test cases with similar coverage are removing in the system, and only keeps one copy of each.

6) **Show test cases:** Test program shows the final test cases generated by node coverage.

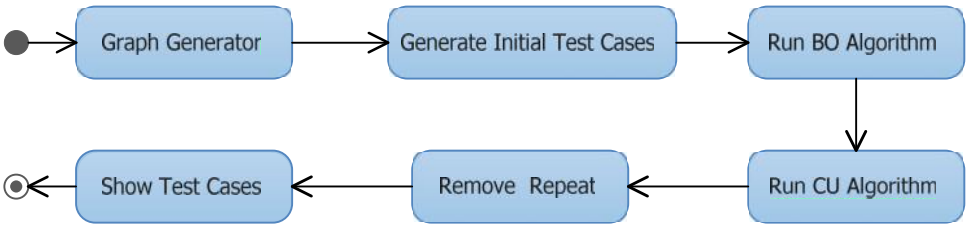

Fig 1: UML activity diagram test system

Proposed approach presented in this paper has been produced in GUTPEL program; this program has two main operations analyzer and generator of automated test cases. The program analyzer receives files and then generates the system control flow graph with a summons related services invoke graph files. Generator is generating test cases needed to cover the graph with using the system running and analyzer information.

3.1.1 Graph Generator

In this section is expressing how to generate control flow graph of BPEL web service and invoke subgraphs of related services. Graph generator uses BPEL analyzer, WSDL analyzer and WSIG analyzer. Each analyzer receives the needed files for analysis, and the results of the analysis graph add to the system. This section output is a control flow graph of the system.

BPEL analyzer receives as input BPEL main service file in the system, and finds service elements, in their moves, and provides service control flow graph. WSDL analyzer input is WSDL file the main service in the system, and it analyzes and finds the needed information to invoke web services. WSIG analyzer input is WSIG related service file and finds the service elements, in their moves, and adds that information to the service control flow graph.

Activity diagram of Figure 2 shows the generate graph system in the proposed approach. If BPEL analyzer found invoke element, then that runs WSIG analyzer for production the service invokes graph.

WSIG files are XML format such as service interfaces (WSDL). In the WSIG file have been specified all nodes in the subgraph, which can generated automatically this file with specified access level by service partners. If the service desired by using BPEL file is created, WSIG file can be created by using that, but in the production with related services using BPEL files is possible only if their files are available from the source to destination or their called directly.

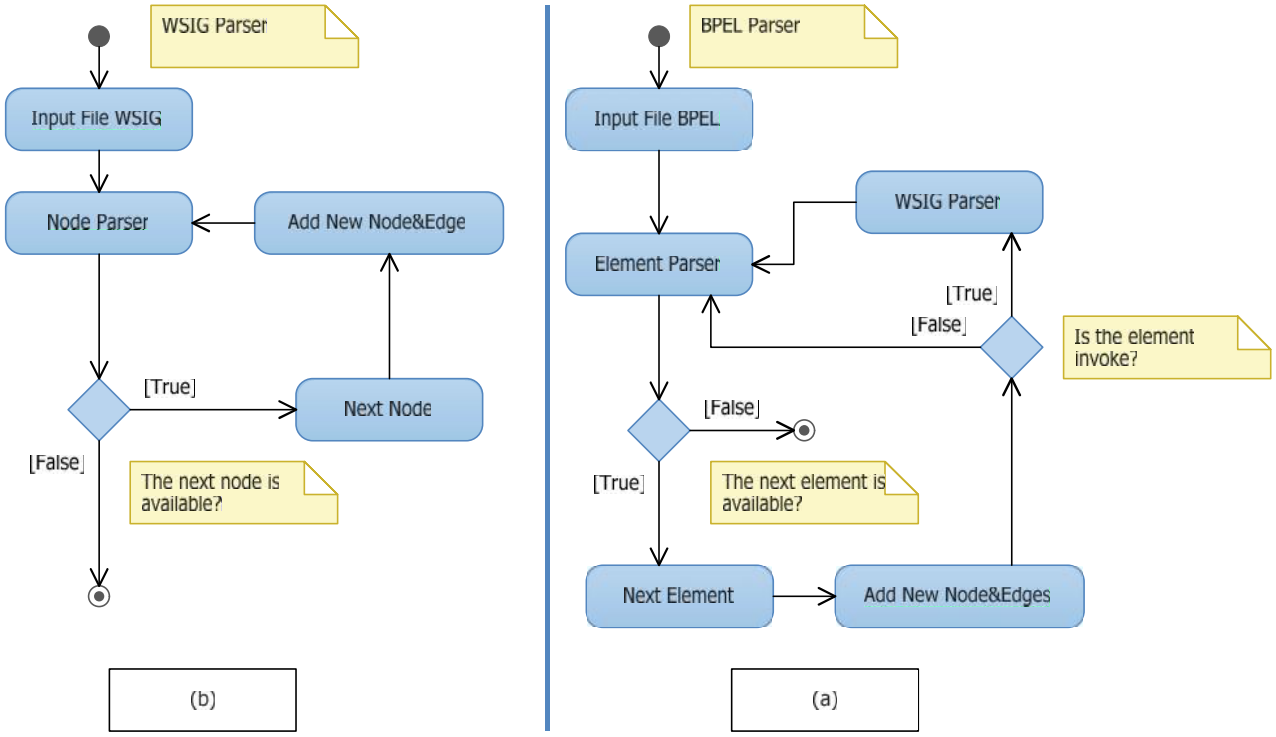

Fig 2: UML activity diagram graph system

Figure 3 (a) shows the WSIG file sample. Nodes specify the related services are WebService1 service in this file, this relationship is in their directly or indirectly. In this Figure, WebService1 and WebService2 and WebService3 are web service, and WebService1 invokes WebService2 and WebService3, and the WebService3 invokes WebService4.

In the Figure 3 (b) shows the invoke subgraphs made from WebService1. This graph specifies each service will be invoke which services in their code. In the figure, each the invoke edge is send/receive. Because the requester invokes to service and the service obtains result and it returns to the requester.

In this implemented system, files WSIG contains only the needed information for the invoke graph. But in this file puts the specifications and other information in the future. These information include the logic of each service is invoked, and similar information, which their will help to improve the test. In this approach taken WSIG file is like the WSDL file.

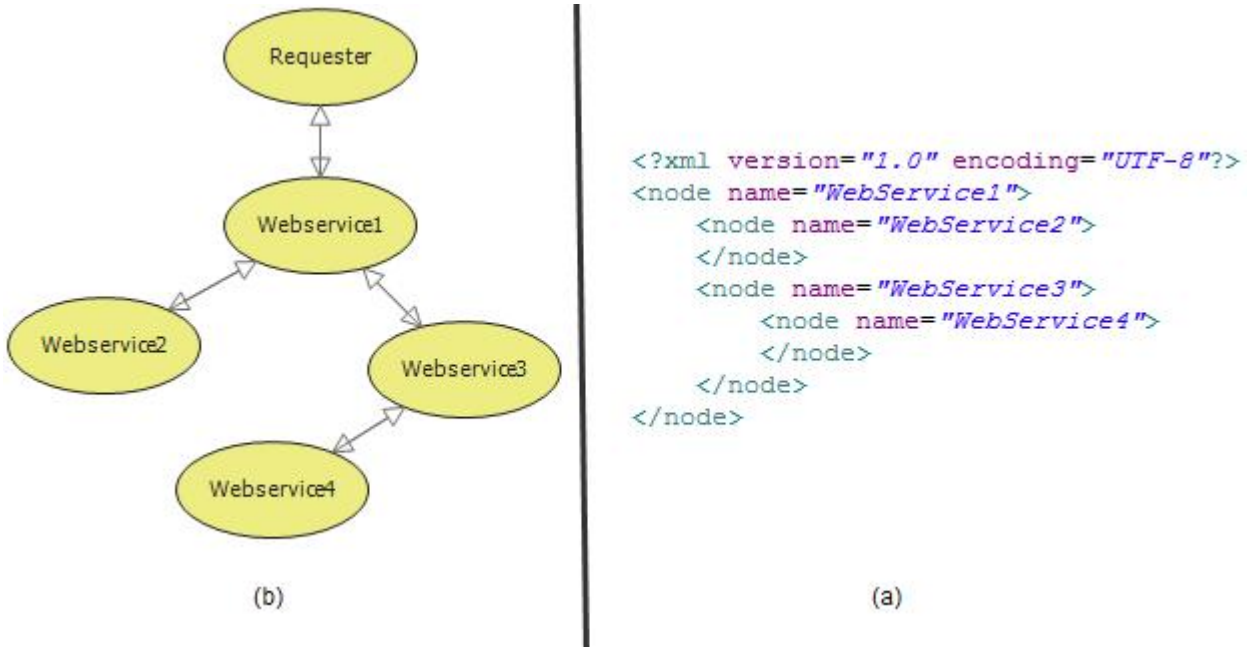

Fig 3: Example WSIG

3.1.2 Test case generator

The proposed approach of creating a service flow graph in the original service at the process system, the required test cases are prepared. Generator of automatic test data contains event BPEL parser. BPEL event data parser gets service events in run of the test, then analyses and gets the traveled path in data analyzing of the graph. Service event parser and BPEL parser are input of generator. Data analyzing use to test of the WSDL file service is generated.

This approach calls test case generator after information identified of the test. The generator specifies the maximum number of test cases, and then the system generates initial test cases randomly. After, BO algorithm will be executed on the initial test cases. Finally, the algorithm CU will be executed on the output of BO algorithm.

The initial test cases are obtained user-defined or pre-existed or randomly. The number of test cases uses a modified version of the McCabe's Cyclomatic Complexity formula [46]:

maxTC = CyclComp(G) + (2#simple_predicates + 1) – 2,

Where CyclComp(G) = #edges _ #vertices + 2, #edges and #vertices are the number of edges and vertices of the control flow graph respectively, and #simple_predicates is the number of simple predicates in the program. McCabe's Cyclomatic Complexity formula, i.e. CyclComp (G) describes only the total number of required test cases to satisfy the statement or edge testing coverage criterion; further to that the proposed formula can determine the additional number of test cases required to achieve condition/edge coverage criterion. In addition, maxTC defines the size of the chromosome, which it's the maximum number of test cases required to achieve full edge/decision coverage.

If the initial test cases are available, and their number is less than the required number for the test, other test cases are generating randomly. Otherwise, if their number is most of the required number for the test, the required number is selected. Test system can be improved the performance with an optimized method for selection from the test cases.

The test system generates number and Boolean values randomly, which their generate limitations of numbers and Boolean generator to improve of the test system. The string values based on some test systems are under control and create a random string of predefined arrays, it can be in order to improve random test generation.

The test approach will be monitored of all messages exchanged between services and observer using orchestration engine, and with a specified input gets output system and test path, and generates test case.

3.1.3 Genetic Algorithms

The test program tries to complete covering the generated graph use to genetic algorithms, this means to produce the required test cases cover all nodes in the graph. BO algorithm tries to generate an optimize test set of test cases based on node coverage criteria of graph. BO algorithm input is the initial test cases, which those are generated in the previous step. In the figure 4 shows activity diagram BO algorithm.

Genes are included input values and covered nodes in the test program. Each input system (input value) is identified with the type's string for the name, type and value at input is in the system. If a gene will be valid if it covers in a node or more nodes, otherwise will be not valid. Chromosome is made the number of genes to cover the graph, number of needed genes for the construction chromosome of the McCabe's formula is specified.

The evaluation function acts with covering the chromosome nodes in the algorithm graph. The evaluation function returns the number of nodes not covered, and the output value of the specified chromosome is lower and closer to zero would be better the covered nodes in the graph covering. If this function returns zero, the generated chromosome covers all nodes in the graph.

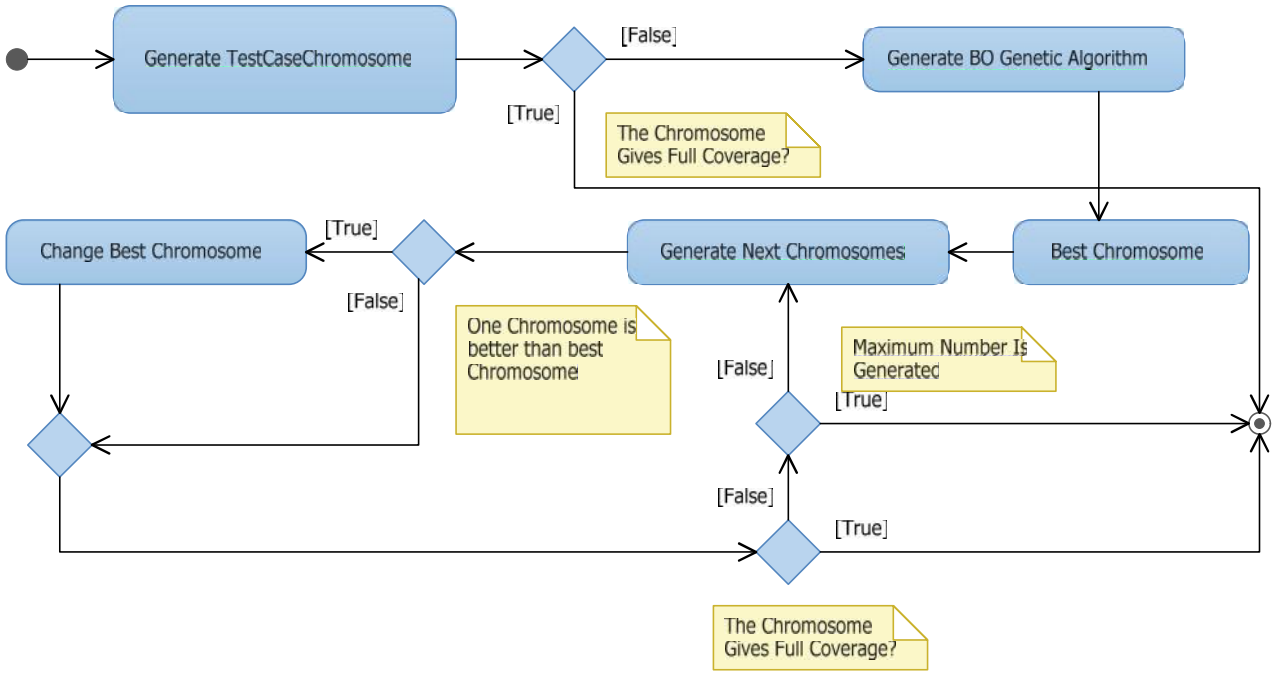

Fig 4: BO algorithm activity diagram

Test cases generated before stage test (run algorithm BO) are input CU algorithm. Figure 5 shows CU algorithm activity diagram. This algorithm is focused on not covered paths, to generate test cases for those paths. That finds the uncover node and in the paths finds the closest covered statement nodes to cover the uncover node. That is trying to change test cases in the passing of statement nodes, and to generate new test cases pass through the uncover node.

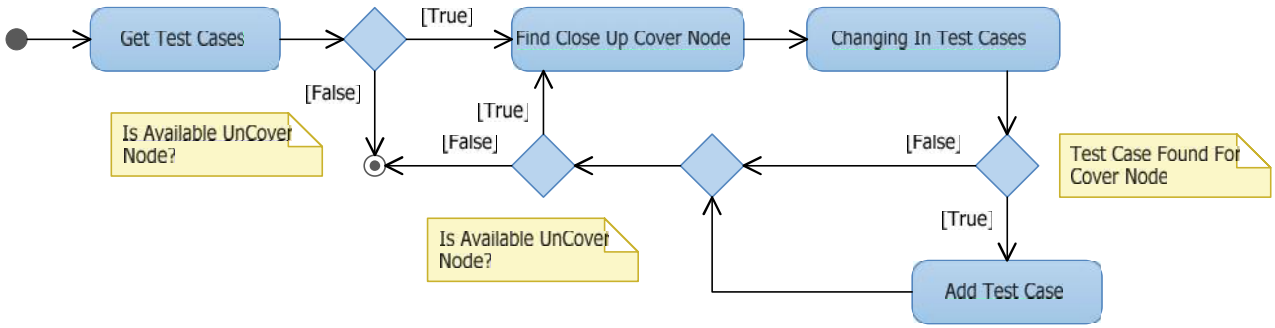

Fig 5: CU algorithm activity diagram

It is possible, generated test cases don't cover all graph paths by the algorithms, random algorithm and BO algorithm and CU algorithm. The number of test cases are repeated of coverage in the system, the test cases repeated will be delete.

To delete test cases repeated: in the first, test cases are scanned, and the repeated test cases are identified, and one of them will keep and others will delete. The proposed approach, the first test case is held in the repeated test cases, and others test cases are deleted. To better test at the algorithm can be used to select appropriate tests and test cases covering the repetitions would repeat deleted.

3.1.4 Show Test Cases

Simple Object Access Protocol (SOAP) is a protocol, and the combination of XML and HTML, this protocol shows how to put messages on the web and data transfer between the service provider and service consumer. In the GUTPEL program after to generate test cases, generated test cases will show on the SOAP protocol based.

3.2 Case Study

This section describes in the two categories of experiments carried out with the testing tools on a Intel Core Dou 2.0 MHZ with 2 GB Ram and JDK 1.6.0_19 running in the Windows 7 operating system and JBoss-5.1.0.GA server.

In this paper is used in the example of RiftSaw tool. BPEL_BluePrint3 prototype system is selected, this example shows a sample purchase order system. This system has two BPEL file, those are InventoryService.bpel and POService.bpel.

Generated graph shows in Figure 6 for the BPEL_BluePrint3 system. The control flow graph displays in POService service and related web services with InventoryService service, to determine the related web service is used with a different color than the other nodes in the graph. As shown in the figure, the edges of the nodes are bidirectional in related web service, because this is a send-receive node.

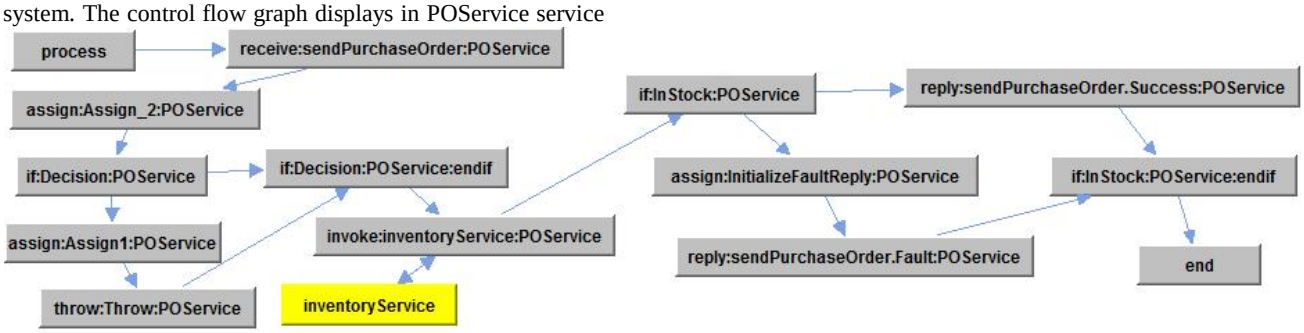

Fig 6: Graph BPEL_BluePrint3

Another system is a reservation system used to testing. The system services invoke graph shows in Figure 7. This system has two subsystem a hotel reservation system and an airline reservation system (the reservation has two airline reservation).

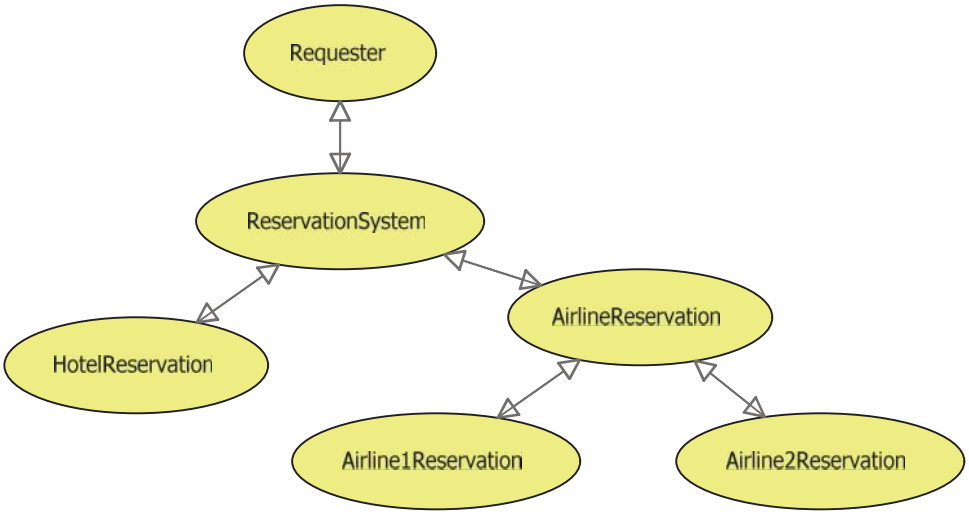

Fig 7: Invoke subgraph ReservationSystem system

3.2.1 Results

In the purchase order system, graph generated has 15 nodes and 17 edges, and number of simple propositions is two, also there is a related web service and it is put two of nodes to send-receive. According to the McCabe's formula number of test cases is required 5 of test cases. In the samples run on the system, generated test cases shown in Figure 8.

This approach was performed 10 times on the systems, its performing results for each system as shown in Table 1 and 2. Part of the run algorithms, all graph system is covered, and the algorithm is completed.

In the reservation system, graph generated has 12 nodes and 15 edges, and there are four related services. According to the McCabe's formula number of test cases is 4.

In the nature of ReservationSystem system can be found a test case to cover all the nodes, Reasons expressed in run randomized algorithm gives the most complete coverage.

As can be seen in the tables, when run the algorithm CU, The number of test cases is higher than other times. Because this algorithm generates new test cases are needed with the available test cases for the covers, and adds them to the test cases.

4. Conclusions and future work

The first part of this paper expressed the related work of SOA testing. Then proposed approach presented to solve the number of SOA testing challenges.

The proposed approach in part 3 of this paper was to generate test cases for BPEL language automatically. Then the prototype system established for the test approach on the coverage criteria.

This approach generates test cases automatically, and the work tester reduces. This approach can be used to the system under consideration uses all services of the system properly perform its operations, or change in a service, the system have the necessary reliability. The generated test cases can be used

in the regression testing, and impact of the change a service identified in the systems. As regards, SOA has expanded in the wide range of industries and organizations and it is still expanding, the stakeholders are organizations, companies and industries.

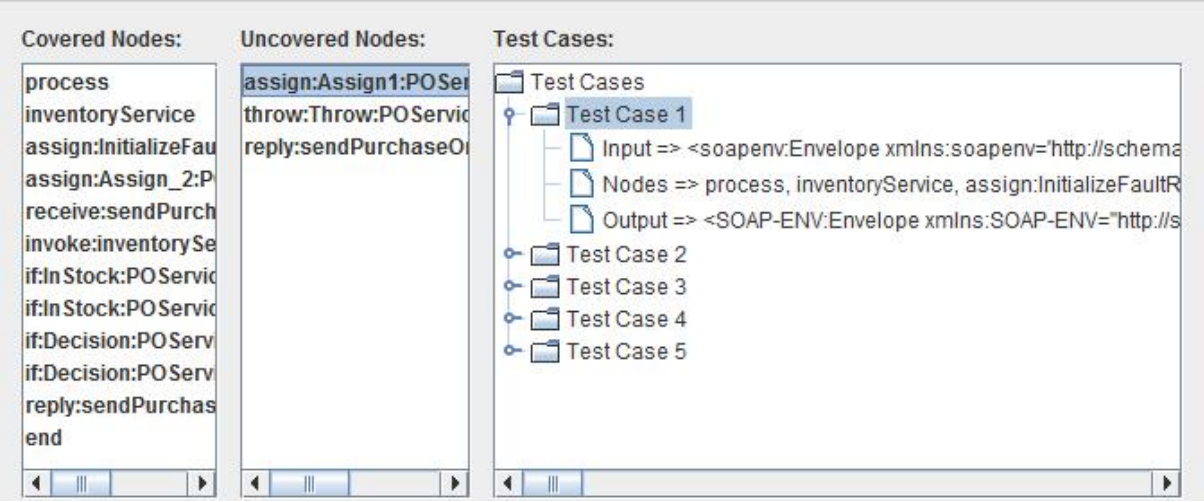

Fig 8: Generated test cases for BPEL_BluePrint3 system

Table 1. Run algorithms in BPEL_BluePrint3 system

Number run	Random Generator node cover	BO Algorithm	CU Algorithm	Number Test cases
1	14	15	Not run	5
2	12	15	Not run	5
3	15	Not run	Not run	5
4	12	15	Not run	5
5	14	15	Not run	5
6	14	15	Not run	5
7	13	15	Not run	5
8	10	15	Not run	5
9	6	14	15	6
10	13	15	Not run	5

Invokes graph generator needs to generate invokes graph from the services were invoked directly. By combining these graphs, the desired service invoke graph will generated. This approach is generator to generate test cases in test integration services by using BPEL, but this approach can be used to test at the unit service.

The test program was created in the graph of a large system by different services; the generated graph will be very large, and display it on the screen and stored in the memory is one of the challenges. Another the challenge is different stakeholders, consequently, all of the partners don't allowed to monitor services to test the system, if they are allowed under certain conditions that are predefined.

Another challenge with this approach, the SOA system is distributed system, and that have been established on the systems with different hardware and different software platforms. It is therefore necessary to test the system with a variety of platforms to support. That sometimes the new implementation of system is easier than the complete system testing in the bad implementation of system.

It is possible, change in the service changes on the service invoke; as a result the invoke graph web service is changed.

The optimal approach is needed use to these approaches, the changed of a service to the services used this service will be optimize, also the generated invoke graph will be optimize.

Table 2. Run algorithms in ReservationSystem system

Number run	Random Generator	BO Algorithm	CU Algorithm	Number Test cases
1	12	Not run	Not run	4
2	12	Not run	Not run	4
3	12	Not run	Not run	4
4	12	Not run	Not run	4
5	11	12	Not run	4
6	12	Not run	Not run	4
7	12	Not run	Not run	4
8	11	12	Not run	4
9	12	Not run	Not run	4
10	12	Not run	Not run	4

The GUTPEL test tool were covered graph nodes in the benchmark, which the test can be performed with the edges covering, paths covering and other measures of the graph covering. SOA testing will create on the logic based. In the test program needs to generate and efficient approach for test cases generation in the parallel nodes. And inaccessible paths will be delete automatically to improve the test graph.

The optimized genetic algorithms used to create test systems. In future work, several approaches and combine them to create an optimized platform for the test. Initial test cases obtained from other methods such as logic system. The optimized algorithm in the target test will generate on specific parts of the system.

5. ACKNOWLEDGMENTS

The author would like to thank specially Dr. Seyed Hasan Mirian Hossienabadi who has extended his support for successful completion of this paper.

6. REFERENCES

[1] Torry Harris Business Solutions (THBS) Company (2007). SOA Test Methodology [White paper]. Retrieved from www.thbs.com/pdfs/SOA_Test_Methodology.pdf

[2] Shamsoddin-Motlagh, E. (2012). A SURVEY OF SERVICE ORIENTED ARCHITECTURE SYSTEMS TESTING. International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, November 2012. 19-27. DOI : 10.5121/ijsea.2012.3602

[3] Wang, Y., Ishikawa, F., Honiden, S. (2010). Business Semantics Centric Reliability Testing for Web Services in BPEL. IEEE 6th World Congress on Services, 237-244. Doi: 10.1109/SERVICES.2010.88

[4] Magedanz, T., Schreiner, F., Wahle, S. (2009). Service-Oriented Testbed Infrastructures and Cross-Domain Federation for Future Internet Research. 2009 IFIP/IEEE Intl. Symposium on Integrated Network Management – Workshops, 101-106. Doi: 10.1109/INMW.2009.5195944

[5] Bai, X., Dong, W., Tsai, W. T., Chen, Y. (2005). WSDL-Based Automatic Test Case Generation for Web Services Testing. 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05), 1-6. Doi: 10.1109/SOSE.2005.43

[6] Jiang, Y., Li, Y. N., Hou, S. S., Zhang, L. (2009). Test-Data Generation for Web Services Based on Contract Mutation. 2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement SSIRI 2009 Short Paper, 281-286. Doi: 10.1109/SSIRI.2009.49

[7] Ma, C., Du, C., Zhang, T., Hu, F., Cai, X. (2008). WSDL-Based Automated Test Data Generation for Web Service. International Conference on Computer Science and Software Engineering, 731-737. Doi: 10.1109/CSSE.2008.790

[8] Dong, W. (2009). Testing WSDL_based Web Service Automatically. World Congress on Software Engineering, 521-525. Doi: 10.1109/WCSE.2009.133

[9] Bartolini, C., Bertolino, A., Marchetti, E. (2009). WS-TAXI: a WSDL-based testing tool for Web Services. International Conference on Software Testing Verification and Validation, 326-335. Doi: 10.1109/ICST.2009.28

[10] Noikajana, S., & Suwannasart, T. (2009). An Improved Test Case Generation Method for Web Service Testing

from WSDL-S and OCL with Pair-wise Testing Technique. 33rd Annual IEEE International Computer Software and Applications Conference, 115-123. Doi: 10.1109/COMPSAC.2009.25

[11] Mei, L., Chan, W.K., Tse, T.H. (2008, May 10-18). Data Flow Testing of Service-Oriented Workflow Applications, ICSE '08, Leipzig, Germany, 371-380. Doi: 10.1145/1368088.1368139

[12] Lertphumpanya, T., & Senivongse, T. (2008, May). Basis Path Test Suite and Testing Process for WS-BPEL. WSEAS TRANSACTIONS on COMPUTERS. ISSN: 1109-2750, Issue 5, Volume 7, 483-496. Retrieved from www.wseas.us/e-library/transactions/computers/2008/26-156.pdf

[13] Yuan, Y., Li, Z., Sun, W. (2006). A Graph-search Based Approach to BPEL4WS Test Generation. Proceedings of the International Conference on Software Engineering Advances (ICSEA'06). Doi: 10.1109/ICSEA.2006.261270

[14] Bartolini, C., Bertolino, A., Elbaum, S., Marchetti, E. (2011). Bringing white-box testing to Service Oriented Architectures through a Service Oriented Approach. The Journal of Systems and Software, 84, 655–668. Doi:10.1016/j.jss.2010.10.024

[15] Ma, C., Wu, J., Zhang, T., Zhang, Y., Cai, X. (2008). Testing BPEL with Stream X-machine. International Symposium on Information Science and Engieering, 578-582. Doi: 10.1109/ISISE.2008.201

[16] Dong, W.L., YU, H., Zhang, Y.B. (2006). Testing BPEL-based Web Service Composition Using High-level Petri Nets. Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 441 – 444. Doi: 10.1109/EDOC.2006.59

[17] Dong, W. (2009). Test Case Generation Method for BPEL-based Testing. 2009 International Conference on Computational Intelligence and Natural Computing, 467-470. Doi: 10.1109/CINC.2009.229

[18] Zheng, Y., Zhou, J., Krause, P. (2007, September). An Automatic Test Case Generation Framework for Web Services. JOURNAL OF SOFTWARE, VOL. 2, NO. 3, 64-77. Retrieved from http://epubs.surrey.ac.uk/1975/1/fulltext.pdf

[19] Lee, Y. (2009). 2-Layered SOA Test Framework Based on BPA1-Simulated Event. Fifth International Joint Conference on INC, IMS and IDC, 1058-1063. Doi: 10.1109/NCM.2009.337

[20] Mei, L. (2009, May 16-24). A Context-Aware Orchestrating and Choreographic Test Framework for Service-Oriented Applications. ICSE'09, Vancouver, Canada, 371-374. Doi: 10.1109/ICSE-COMPANION.2009.5071024

[21] Hou, J., & Xu, L. (2009). DFTT4CWS: A Testing Tool for Composite Web Services Based on Data-Flow. Sixth Web Information Systems and Applications Conference, 62-67. Doi: 10.1109/WISA.2009.19

[22] Huang, J, & Gong, Y. (2010). An EMF Activity Tree Based BPEL Defect Pattern Testing Method. 2nd International Conference on Computer Engineering and Technology, 7, 468-471. Doi: 10.1109/ICCET.2010.5485536

[23] Cavalli, A., Cao, T.D., Mallouli, W., Martins, E., Sadovykh, A., Salva, S., Za¨ıdi, F. (2010).WebMov A dedicated framework for the modelling and testing of Web Services composition. IEEE International Conference on Web Services, 377-384. Doi: 10.1109/ICWS.2010.24

[24] Bo, Y., Ye-mei, Q., Ge, Y., Chang, G. (2009). Tabu Search and Genetic Algorithm to Generate Test Data for BPEL Program. Computational Intelligence and Software Engineering (CiSE), 1-6. Doi: 10.1109/CISE.2009.5363674

[25] Bakota, T., Beszédes, Á., Gergely, T., Gyalai, M. I., Gyimóthy, T., Füleki, D. (2008). Semi-Automatic Test Case Generation from Business Process Models. This research was supported in part by the Hungarian national grants RET-07/2005, OTKA K-73688 and TECH_08-A2/2-2008-0089 SZOMIN08. Retrieved from http://www.inf.u-szeged.hu/~beszedes/research/bakota09_semiautomatic.pdf

[26] Mani, S., Sinha, V. S., Sinha S. Dhoolia, P. Mukherjee, D. Chakraborty, S. (2009). Efficient Testing of Service-Oriented Applications Using Semantic Service Stubs. IEEE International Conference on Web Services, 197-204. Doi: 10.1109/ICWS.2009.40

[27] Yuan, Q., Wu, J., Liu, C., Zhang, L. (2008). A Model Driven Approach Toward Business Process Test Case Generation. 10th International Symposium on Web Site Evolution (WSE), 41-44. Doi: 10.1109/WSE.2008.4655394

[28] Jehan, S., Pill I., Wotawa, F. (2013). Functional SOA Testing Based on Constraints. AST 2013, San Francisco, CA, USA. 33-39. 978-1-4673-6161-3/13 IEEE.

[29] Li, B., Qiu, D., Ji, S., Wang, D. (2010). Automatic Test Case Selection and Generation for Regression Testing of Composite Service Based on Extensible BPEL Flow Graph. 26th IEEE International Conference on Software Maintenance in TimiSoara, Romania, 1-10. Doi: 10.1109/ICSM.2010.5609541

[30] Li, B., Qiu, D., Leung, H., Wang, D. (2012). Automatic test case selection for regression testing of composite service based on extensible BPEL flow graph. The Journal of Systems and Software Volume 85, Issue 6, 1300–1324. Doi:10.1016/j.jss.2012.01.036

[31] Canfora, G., & Penta, M. D. (2006). SOA Testing and Self-Checking. International Workshop on Web Services Modeling and Testing (WS-MaTe 2006), 3-12. Retrieved from http://www.selab.isti.cnr.it/ws-mate/Canfora_WS-MaTe.pdf

[32] MOHANTY, R. K., PATTANAYAK, B. K., PUTHAL, B., MOHAPATRA, D. P., (February 2012). A ROAD MAP TO REGRESSION TESTING OF SERVICEORIENTED ARCHITECTURE (SOA) BASED APPLICATIONS. Journal of Theoretical and Applied Information Technology. 60-65. 15 February 2012. Vol. 36 No.1

[33] Zmudaa, D., Psiuk, M., Zielinski, K. (2010). Dynamic monitoring framework for the SOA execution environment. International Conference on Computational Science (ICCS 2010), 1, 125-133. Doi:10.1016/j.procs.2010.04.015

[34] Kabbani, N., Tilley, S., Pearson, L. (2010, April 5–10). Towards an Evaluation Framework for SOA Security Testing Tools. SysCon 2010 – IEEE International Systems Conference San Diego, CA, 438-443. Doi: 10.1109/SYSTEMS.2010.5482322

[35] Ilieva, S., Pavlov, V., Manova, I. (2010). A Composable Framework for Test Automation of Service-Based Applications. 2010 Seventh International Conference on the Quality of Information and Communications Technology, 286-291. Doi: 10.1109/QUATIC.2010.54

[36] Cao, T.D., Felix, P., Castanet, R. (2010). WSOTF An Automatic Testing Tool for Web Services Composition. Fifth International Conference on Internet and Web Applications and Services, 7-12. Doi: 10.1109/ICIW.2010.9

[37] Lallali, M., Zaidi, F., Cavalli, A., Hwang, I. (2008). Automatic Timed Test Case Generation for Web Services Composition. Sixth European Conference on Web Services, 53-62. Doi: 10.1109/ECOWS.2008.14

[38] Bertolino, A., Angelis, G.D., Lonetti, F., Sabetta, A. (2008). Automated Testbed Generation for Service-oriented Mobile Applications. 34th Euromicro Conference Software Engineering and Advanced Applications, 321-328. Doi: 10.1109/SEAA.2008.33

[39] Martin, E., Basu, S., Xie, T. (2007). Automated Testing and Response Analysis of Web Services. IEEE International Conference on Web Services (ICWS), 647-654. Doi: 10.1109/ICWS.2007.49

[40] Conroy, K. M., Grechanik, M., Hellige, M., Liongosari, E. S., Xie, Q. (2007). Automatic Test Generation From GUI Applications For Testing Web Services. Software Maintenance, IEEE International Conference on ICSM, 345-354. Doi: 10.1109/ICSM.2007.4362647

[41] Melo A. C.V. d., & Silveira, P. (2011). Improving data perturbation testing techniques for Web services. Information Sciences 181, 600–619. Doi:10.1016/j.ins.2010.09.030

[42] Mohammad, A. F., & Mcheick, H. (2011). Cloud Services Testing An Understanding. The 2nd International Conference on Ambient Systems, Networks and Technologies, Procedia Computer Science, 5, 513–520. Doi:10.1016/j.procs.2011.07.066

[43] Palacios, M., Garcio-Fanjul, J., Tuya, J. (2011). Testing in Service Oriented Architectures with dynamic binding: A mapping study. Information and Software Technology, 53, 171–189. Doi:10.1016/j.infsof.2010.11.014

[44] Kalamegam, P., Godandapani, Z. (October, 2012). A Survey on Testing SOA Built using Web Services. International Journal of Software Engineering and Its Applications. 91-104, Vol. 6, No. 4.

[45] Sofokleous, A. A., Andreou, A. S. (2008). Automatic, evolutionary test data generation for dynamic software testing. The Journal of Systems and Software, 81, 1883–1898. Doi:10.1016/j.jss.2007.12.809

[46] Sofokleous, A. A., Andreou, A. S. (2008). Automatic, evolutionary test data generation for dynamic software testing. The Journal of Systems and Software, 81, 1883–1898. Doi:10.1016/j.jss.2007.12.809