The Twelfth International Conference on Digital Information Management (ICDIM 2017)
September 12- 14, 2017, Kyushu University, Fukuoka, Japan.

Improving the trust of end users in enterprise SOA using combinatorial group testing methods

Siba Mishra and Chiranjeev Kumar
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines)
Dhanbad - 826004, Jharkhand, India
Email: sibamishracse@gmail.com and k_chiranjeev@yahoo.co.uk

***Abstract*—Over the last decade, Service-Oriented Architecture (SOA) is widely accepted for building large-scale, complex and heterogeneous business process enterprise applications. The SOA systems are developed by composing loosely coupled services that are shared across the organizations. Nowadays, the procreation of SOA is baffled by the presence of faulty business processes (services), which severely influence the experience of end users and causes significant frustrations leading to create a barrier between the service providers and the end users. In this paper, we inspect an abstract formulation of the problem to identify faulty business processes (services) for improving the trust of end users in enterprise SOA systems. In SOA, the faulty business processes are integrated with some erroneous operations and that should be identified during the early stage of testing for increasing the trust between the service providers and end users. Thus, we develop an abstract model of this problem which primarily aimed to improve the desired level of trust in the end users. The trust is assured through the formal correspondence of well-established literature of combinatorial group testing (CGT). Furthermore, we apply a sequential group testing method to the problem of SOA system testing to identify the erroneous services. Lastly, some limitations of the used approach with some future directions, that can lead to new solutions are highlighted.**

***Index Terms*—SOA, combinatorial group testing, test, fault, business processes, trust.**

I. Introduction

The notion of group testing is to determine all the defective items from a large population using minimum number of tests, where each test, is applied to a subset of items instead of testing them individually [1]. This powerful mathematical theory was first introduced by Robert Dorfman for detecting *syphilis* in blood samples taken from the US draftees during the World War II [1].

From its simple nature, during the last decades, the group testing problem found in various domains, like data streams [2], multiple access communication [3], compressive sensing [4], container inspection problem [5], medical sciences [6] and many more. For a concise detail of this subject, we suggest the reader to the monograph by Du et al. [7]. Despite of its vast popularity in other fields, the intervention of this powerful mathematical concept into software testing has also been discovered recently [8]. Similar to all earlier applications of group testing, this concept requires some modifications/renovations of the classical group testing approaches and algorithms to overcome the obstacle by applying the mathematical models to the practical scenarios of enterprise SOA based software testing. For example, under which conditions can group testing be constructed and tested on the enterprise business process SOA systems.

Therefore, in this paper, we introduce the interpretation problem in the area of group testing and then inspect, how closely it is related in identifying faulty business processes[1] of an enterprise SOA system under test. More concisely, this interpretation problem is concerned in *identifying the faulty business process set*, from the system under test developed by the *developers*. It is the *service providers*, who deals with the Quality of Service (QoS) in SOA and are responsible for providing the services to other stakeholders[2] of the SOA system. Therefore, before delivering the developed services to the *integrator*, *third-party certifier* and *end users* (other major stakeholders of the SOA system), the *service providers* should always ensure regarding its correct functioning. A high level representation of major stakeholders and their interaction to the SOA system is shown in Fig. 1.

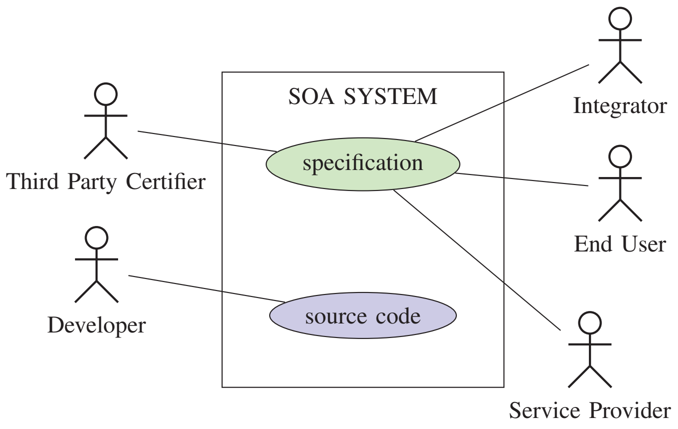

Fig. 1. High level representation of major stakeholders and their interactions to the SOA system.

Furthermore, the service providers also need to assure that

[1]A business process or service is the collection of tasks integrated with operations and attributes that results in a final product, serving a particular goal of the end user. In this section and throughout the paper, the terms business processes and services are used interchangeably.

[2]The major stakeholders of the SOA system have been adapted from Canfora et al. [9].

always the services operate within the condition/constraints of the service level agreement (SLA). Given a subset of services under test, the service provider/tester gives a *positive* answer, if this subset has erroneous operations, i.e. faulty services and a *negative* answer, if the subset is free from any erroneous operations. This interpretation problem is defined as follows.

Interpretation Problem. For a SOA system under test ($SOASUT$), given a set of services (SE) encapsulated with (T) tasks, (O) operations and (E) erroneous services with an (U) upper bound. Identify the subset of faulty or erroneous services $F \subset SE$, where, $E = |F|$, which is unknown, using minimum number of tests.

In this paper, we reveal the fundamental research on SOA system testing, discusses its perspectives and proclaim that our contribution is a crucial step for the entire SOA system under test. We have studied, analyzed, proposed and developed a concrete SOA system testing technique, from the perspective of some major stakeholders of SOA system (see Fig. 1). Our contribution is helpful for delivering a good quality product and also makes an attempt to improve the trust between the service providers and the end users. Section II discussed the review of related works. In Section III, the proposed methodology based on sequential group testing is presented. The proposed algorithm is complemented with the help of a case study, along with its empirical evaluation which is discussed in Section IV, that takes, real time SOA dynamics into consideration. Some discussion highlighting the strengths and limitations of the proposed algorithm and some challenges faced during the empirical evaluation of the proposed approach are shown in Section V. The paper is concluded by summarizing the main perspectives and future work in Section VI.

II. Related Work

SOA has emerged as a leading solution in delivering heterogeneous complex business process applications, mainly due to the property of *loosely coupled* styled architecture and messaging [10]. Systems using SOA are known as SOA based systems, which are helpful in addressing many problems related to the integration of heterogeneous applications in distributed environment [11, 12]. Moreover, the design pattern of SOA is based on layered architecture. Therefore, it advocates more parallel development and break the task of testing phases into definable test areas such as *services*, *security* and governances [13]. In the last decade, due to the *agile* and *versatile* nature of software, possible changes to SOA system increases with the number of involved services. For every change to the required service, the SOA system require more testing.

The testing of SOA systems present several new challenges to the testers, when compared against the traditional softwares. One of the main challenges is to identify the “right time” to *test* the operations associated with the tasks of services, that are affected by the applied changes [14]. The redeeming of system by third parties and the services with no user interface also present new complexities to the project managers, during testing [15]. However, it is necessary that each and every service involved in the system should be tested properly to satisfy the end user in terms of good quality product. Thus, more testing effort is required at the *service level* than the *system level*. The services having bugs and quality related issues are not ideal to reuse. Therefore, it is essential to identify all the faulty and erroneous services during the early stage of testing for improving the trust among the end users. In the literature, there exist plenitude approaches of SOA testing.

In general, the research on SOA system testing constitutes mainly specification based, model based and fault based test case generation [14], test sequence/data generation for single services, business processes and composite services [16–19]. However, it is important and ideal that the testing process of SOA systems should cover all the test domains as shown in Fig. 2. Furthermore, apart from the area of test case generation, the research on identifying the faulty services integrated with erroneous operations is a decisive area, which should be taken into consideration, which primarily aimed to improve the trust between the end users and service providers.

III. Proposed Methodology

In this section, we present a new technique to test the SOA system. For improving the trust between end users and service providers, it is essential that the testing technique should applied before the delivery of product to the end user. Our contribution is based on revealing the faulty services, for the SOA system under test using sequential group testing technique.

Nowadays, SOA has emerged as the popular technology adhering fast growing change of business requirements (agility), which result in a greater ROI (Return on Investment) to the software organization [20]. One biggest benefit SOA provide to the testers is that, it facilitates testing of independent reusable services [10]. Additionally, these *reused services* can be tested independently which forces testers not to test the overall application, unless and until all the services of SOASUT passed successfully [10, 13]. In the literature of SOA system testing, there exist a plethora number of test case generation approaches and that can be used to generate test cases by utilizing the tasks, attributes and operations information of the services under test (SOASUT). In the SOA system testing, the test responses named as actual output (AO) are generated using the test cases, and then these responses were compared with the expected output stored in the test oracle, as shown in Fig. 3. In this process, the test oracle confirmed that, whether the services passed the test or not. Furthermore, if the actual output matches with the expected output, then the test cases are known to be *passed*, otherwise it is *failed* during the process of testing.

The approach shown in Fig. 3 is generally followed to test the SOA system under test. From the perspective of stakeholders of SOA system, it is always necessary to reveal the faulty services during the early stage of testing. To accord the above issue, the author(s) make an attempt to improvise the existing testing approach by identifying only the faulty services, soon

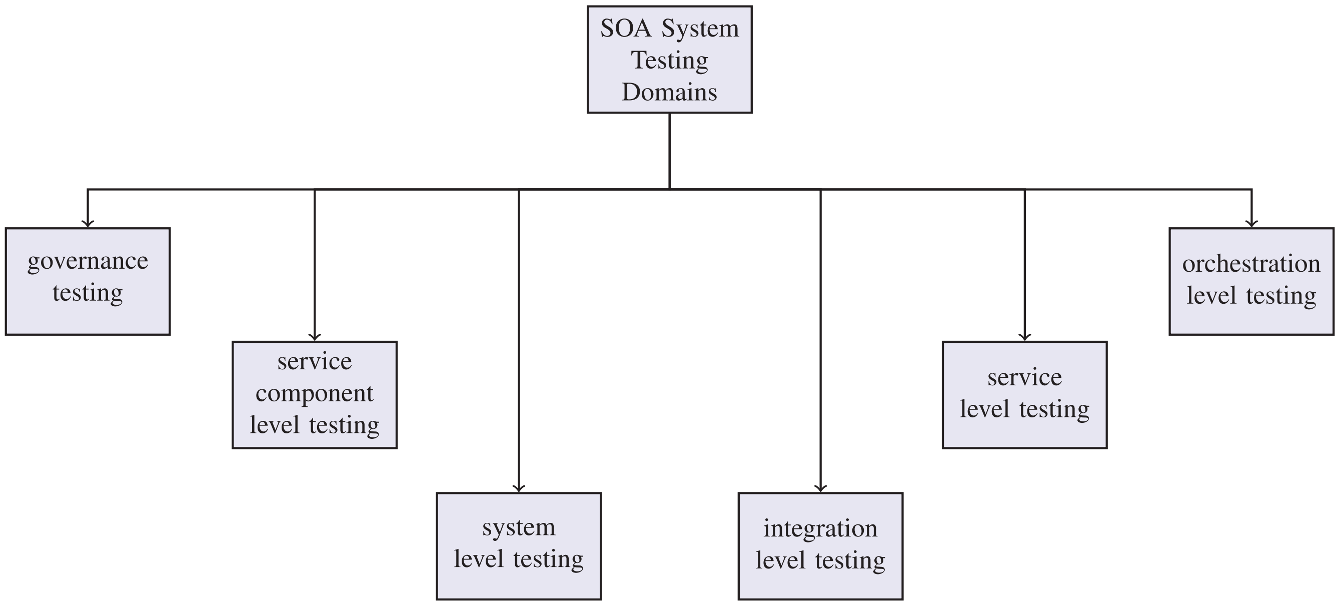
SOA System Testing Domains
governance testing
service component level testing
system level testing
integration level testing
service level testing
orchestration level testing

Fig. 2. Various SOA system testing domains.

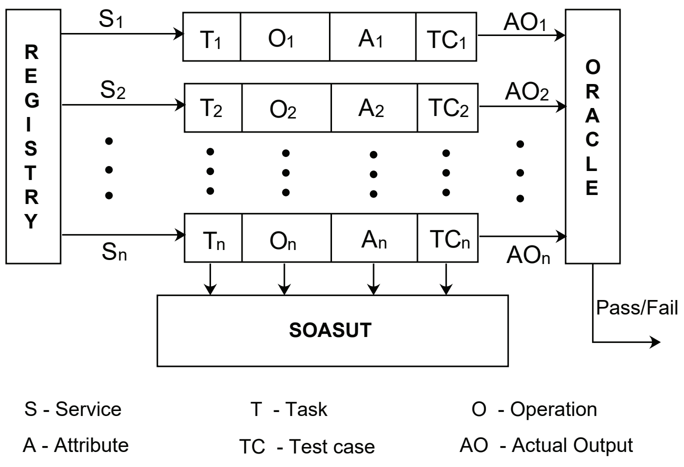

Fig. 3. General overview of SOA system testing.

after the completion of development activities. Hence, this paper solve the problem of SOA system testing by dealing in identifying the faulty services captured with erroneous tasks and operations during the development activities. The main *objective* of this interpretation problem is to reveal all the faulty services using the minimum number of tests that is essential to improve the trust between the service providers and the end users. Moreover, the association of SOA system and group testing furnishes a powerful technique to overcome the above mentioned issues and scenarios. We argue and show that the generic SOA testing procedure is improved using the sequential group testing. From the literature of CGT, we have adapted the sequential group testing algorithm [21], modified the algorithm for our above mentioned interpretation problem. The algorithm is named as IDENTIFY_FAULTY SERVICES and is described in Algorithm 1.

The proposed methodology is illustrated step by step as follows.

Step 1 Initialization: Initialize the service (SE) size and define the expected output (EO) in the oracle.

Step 2 Generation of Test Cases (*TC*) and Calculation of the Actual Output (AO): In this step, firstly, the test cases[3] are required to be generated for the eXtensible Markup Language (XML) based protocols like Simple Object Access Protocol (SOAP)[4] and Representational State Transfer (REST). Then, by using these test cases the actual output for the services under test are to be calculated.

Step 3 Initial Operations: Before commencing the testing process, some trivial operations are required to be performed to check whether to test the services independently or by using group. This is achieved by utilizing the initial defined service set size and the erroneous service upper bound.

Step 4 Calculation of the Non-Negative Integers: In the proposed methodology, two non-negative integers namely l and α are to be calculated. The values of these non-negative integers are used in the later part of the proposed algorithm. The value of α is used to define the size of group that are to be tested.

[3]The test cases are to be generated for all the services (SE) under test, using information specified to develop the services like tasks, operations and attributes.

[4]SOAP is most commonly used XML based protocol in the academia and industry to define the services of SOA.

Algorithm 1 IDENTIFY_FAULTY SERVICES

```
Input: $SE_i$ and $U$
Output: $E$
 1: initialize the service set (SE) and define the expected output set (EO) in the test oracle    ▷ where size(EO) = size($SE$)
 2: start: $A(U, SE)$    ▷ This step mark the problem formulation to solve the problem for the initialized values of $SE$ and $U$
 3: for $i$ = 1 to size($SE$) do
 4:     $SE_i$ service generates $TC_i$ test case ▷ $TC$ is the test case
 5:     compute $AO_i$ using $TC_i$ test case ▷ $AO$ is the actual output
 6:     if size(SE) $\leq 2U$ - 2 then
 7:         test the $SE_i$ services independently
 8:     else if size(SE) $\geq 2U$ - 1 then
 9:         calculate $l = SE - U + 1$
10:         calculate $\alpha$ satisfying $2^\alpha > \frac{l}{u} \geq 2^\alpha$    ▷ $l$ and $\alpha$ are the calculated non-negative integers
11:     end if
12:     take a group ($G$) of service size $2^\alpha$ to test    ▷ the service size equals to $2^\alpha$
13:     if $G[AO_i]$ = $G[EO_i]$ then
14:         $G$ is error free    ▷ $2^\alpha$ services are error free
15:         go to start: ▷ solving the remaining part of the problem
16:         compute, A($U$, $SE - 2^\alpha$)
17:     else
18:         find all the erroneous services in $\alpha$ tests by using binary search and discard all the error-free services
19:         add all the erroneous services to the faulty service set ($F$)
20:     end if
21: end for
22: after (1 + $\alpha$) tests, go to start: to solve the remaining problem $G(U - 1, SE^{'})$    ▷ $SE^{'} \leq SE$ - 1
23: return F    ▷ where, F $\subset SE$ and |F| = $E$
24: count($F$)
```

Step 5 Testing the Services: The services are tested in a group of size 2^α. For testing the services under test, the calculated actual output based on the test cases of the group are matched with the expected output stored in the test oracle. If matching occurs, then the test case for that particular service is *passed*, otherwise it is failed. Next, from the group under test, the exact erroneous services are revealed using the binary search. The total number of faulty services are collected from the erroneous service set (F). The same process is repeated to solve the remaining part of the problem. Finally, the erroneous/faulty services are calculated by counting the set F.

IV. Case Study

In this section, we empirically evaluate the proposed methodology with the help of a case study that utilizes real SOA dynamism into account. For conducting the experiment, we develop small sized services. During the development process, we randomly set o erroneous operations integrated with the tasks of services. The main goal here is to compute the minimum number of tests required to reveal all the faulty services.

In our case study, we have developed 20 services (SE) having three tasks integrated with two operations each. For evaluating the efficacy and conducting the empirical evaluation of the proposed algorithm, 3 erroneous operations have been set in the developed services. Now, the main objective is to extract all the faulty services using minimum number of tests. The test cases are generated by using the available system specifications. The service related information are extracted from the XML schema specification. The test cases generated for each service are based on the operation coverage of the services.

The upper bound (U) of 4 is set and utilized for the proposed algorithm. So, for solving the problem $(4, 20)$, the value of α is calculated and found to be 2. Thus, from the algorithm, a group of size 2^α is to be chosen randomly to test and reveal the faulty services integrated with erroneous operations. We have divided the initial service set into 5 distinct groups, namely $G1$, $G2$, $G3$, $G4$ and $G5$, where each group having 2^α, i.e. 4 services in the sequential order. The test oracle is having the expected output. Out of the 20 services, $SE5$, $SE7$ and $SE9$ are set with erroneous operations.

For the first group under test ($G1$), there are no faulty services. In this way, from the first test, 4 services are error free. Similarly, the process of group testing reduces the testing effort. There is no need to test further the group under test. Thus, for identifying the 3 erroneous services, only three tests is required. Furthermore, the faulty services integrated with erroneous services are identified using the process of binary search for the faulty group under test. Instead of testing 20 services, only three detailed tests is enough to extract all the faulty services, which significantly improves the testing effort and time. This is because to test 20 services individually, we need 20 tests, which is achieve only by 3 tests.

V. Discussion

In this section, some discussion(s) related to the proposed algorithm have been introduced and described by analyzing its limitations and strengths. Furthermore, the challenges faced while conducting the empirical experiments are also highlighted in this section.

A. Limitations

This subsection present the limitations of our proposed algorithm.

- The proposed algorithm has been tested on a *small size* service set. For demonstrating its efficacy, it is necessary that the algorithm should be tested for large size service set.

- The *complexity* of proposed algorithm is $O(n \log n)$. We hope that this complexity could be improved by conducting further research with the existing algorithms reported in the CGT literature.

- For our proposed algorithm, the value of upper bound (U) for the erroneous/faulty services is needed, as this upper bound (U) is used as an input to the proposed algorithm. Moreover, we hope that for many practical cases, there is a possibility that the upper bound of the faulty/erroneous services may not be known.

B. Strengths

This subsection discusses the strengths of our proposed algorithm.

- **Easy to develop:** The proposed algorithm is very easy to implement. This is because the service group size, that are to be tested is calculated from the initially computed α value.
- **Simple and flexible:** Although, the proposed algorithm is based on a powerful mathematical concept. It is very much simple and flexible because it requires only the upper bound and does not involve any complex probability distributions.
- **Reducing the testing effort:** For SOASUT, the services under test are tested in a group. The test is applied to a group of services, instead of testing them individually. This itself save a significant number of tests and results in improving the testing effort and time.

C. Challenges

Some challenges were faced during the empirical evaluation of the proposed algorithm. The challenges are highlighted as follows.

- **Defining the test oracle:** In a large scale, defining the test oracle is a very complex process and also costly in terms of project budgetary resources. In the literature of software testing, some researchers suggest to test the system without the test oracle, however that is beyond the scope of this paper.
- **Test Case Generation:** In the literature of SOA system testing, the research on test case generation falls into three major categories: specification, model and fault based approaches depending on the single service and composite services. However, considering the service governance, service composition and service orchestration as a whole is a difficult task.

VI. Conclusion

Our work has addressed the SOA system testing and revealed the previously unexplored connections to the field of combinatorial group testing. The main contributions are summarized as follows.

- We exhibit that the testing of SOA systems correspond fundamentally to the well-established area of CGT. The intersection of these two research fields discloses enormous technique for the SOA system under test.
- From the CGT literature, we have adapted and modified the sequential combinatorial group testing algorithm for the interpretation problem. The empirical assessment of the proposed methodology is facilitated with the help of a case study. The empirical evaluation incurred from the case study of the proposed algorithm shows that they outperform the generic software testing.

In this paper, we directly use the CGT technique to the problem of SOA system testing. Furthermore, a number of research directions remain open, and offer the prospect of more cost-effective SOA system test solutions as well as some new theoretical frameworks. Some direction for future work are: testing the SOA system without the upper bound of faulty services. We believe that in many practical real time scenarios and cases, there is a possibility that the number of faulty services are not known before the testing process. Thus, some methodology should exist to address the same. This can be achieved by studying, analyzing and applying competitive and non-adaptive group testing algorithms reported in the literature of CGT.

References

[1] R. Dorfman, "The detection of defective members of large populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[2] G. Cormode and S. Muthukrishnan, "What's hot and what's not: tracking most frequent items dynamically," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 249–278, 2005.

[3] M. T. Goodrich and D. S. Hirschberg, "Improved adaptive group testing algorithms with applications to multiple access channels and dead sensor diagnosis," *Journal of Combinatorial Optimization*, vol. 15, no. 1, pp. 95–121, 2007.

[4] A. C. Gilbert, M. A. Iwen, and M. J. Strauss, "Group testing and sparse signal recovery," in 42^{nd} *Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 1059–1063.

[5] A. L. Concho and J. E. Ramirez-Marquez, "Optimal design of container inspection strategies considering multiple objectives via an evolutionary approach," *Annals of Operations Research*, vol. 196, no. 1, pp. 167–187, 2012.

[6] P. Hou, J. M. Tebbs, C. R. Bilder, and C. S. McMahan, "Hierarchical group testing for multiple infections," *Biometrics*, 2016. [Online]. Available: http://dx.doi.org/10.1111/biom.12589

[7] D.-Z. Du and F. K. Hwang, *Combinatorial group testing and its applications*. World Scientific, 1999, vol. 12.

[8] W.-T. Tsai, X. Zhou, Y. Chen, and X. Bai, "On testing and evaluating service-oriented software," *Computer*, vol. 41, no. 8, pp. 40–46, 2008.

[9] G. Canfora and M. Di Penta, "Service-oriented architectures testing: A survey," *Software Engineering*, pp. 78–105, 2009.

[10] N. M. Josuttis, *SOA in practice: the art of distributed system design*. O'Reilly Media, Inc., 2007.

[11] S. S. Yau, Y. Yao, and A. B. Buduru, "An adaptable distributed trust management framework for large-scale secure service-based systems," *Computing*, vol. 96, no. 10, pp. 925–949, 2014.

[12] S. Mishra and C. Kumar, "A novel adaptive structure for SOA system effort estimation," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 8, pp. 1115–1127, August 2016.

[13] T. Erl, *SOA: Principles of Service Design*. Prentice Hall, Press, 2007.

[14] M. Bozkurt, M. Harman, and Y. Hassoun, "Testing and verification in service-oriented architecture: a survey," *Software Testing, Verification and Reliability*, vol. 23, no. 4, pp. 261–313, 2013.

[15] "SOA Testing Tutorial." [Online]. Available: http://www.softwaretestinghelp.com/soa-testing

[16] M. De Barros, J. Shiau, C. Shang, K. Gidewall, H. Shi, and J. Forsmann, "Web services wind tunnel: On performance testing large-scale stateful web services," in 37^{th} *Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN'07*, 2007, pp. 612–617.

[17] E. Martin, S. Basu, and T. Xie, "Websob: A tool for robustness testing of web services," in *Companion to the proceedings of the* 29^{th} *International Conference on Software Engineering*, 2007, pp. 65–66.

[18] J. Zhang and L.-J. Zhang, "Criteria analysis and validation of the reliability of web services-oriented systems," in *IEEE International Conference on Web Services (ICWS'05)*, 2005, pp. 621–628.

[19] A. T. Endo and A. Simao, "Event tree algorithms to generate test sequences for composite web services," *Software Testing, Verification and Reliability*, 2017. [Online]. Available: http://dx.doi.org/10.1002/stvr.1637

[20] S. Mishra and C. Kumar, "Estimating development size and effort of business process service-oriented architecture applications," in 2^{nd} *International Conference on Systems and Informatics (ICSAI)*, 2014, pp. 1006–1011.

[21] F. Hwang, "A method for detecting all defective members in a population by group testing," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 605–608, 1972.