AN EMPIRICAL INVESTIGATION ON TEST DATA GENERATION FOR COUPLING BASED INTEGRATION TESTING

Shaukat Ali Khan
Center for Software Dependability, Mohammad Ali Jinnah University (MAJU),Islamabad, Pakistan
shaukatali74@gmail.com
Aamer Nadeem
Center for Software Dependability, Mohammad Ali Jinnah University (MAJU), Islamabad, Pakistan
anadeem@jinnah.edu.pk

ABSTRACT*— Evolutionary approaches are well suited for automatic generation of good quality test data for software testing. Software testing is not effective without good quality test data, so good quality software testing depends upon good quality software test data. Automatic test data generation becomes very critical when we come to a higher level of testing including integration testing and system testing. Unit testing can be managed with manual data to test single methods and classes. In Integration testing and system testing, a large number of methods and classes are involved so there should be effective test data generation strategy for the generation of automatic test data. In this paper, we have selected a case study from industry for our empirical investigations on automatic test data generation for coupling based integration testing. We have identified different coupling scenarios based on the selected application configurations and data. We have performed different experiments using our already proposed approach described somewhere else [46]. Based on our experimental investigations, we have concluded that our approach is very effective for test data generation for different coupling types involved in integration of different components. Our experimental measurements indicate that our approach is very effective as compared to random testing.*

Keywords- Fitness Function; Integration Testing; Test Data ; Antecendent Method; Coupling Variable; Genetic Algorithm

I. INTRODUCTION

The execution of software with intends of finding error is known as software testing [20]. Software testing is one of the most important phases in the software development life cycle. Software testing is an ongoing activity and can be performed at each level starting from requirements to acceptance testing. Different levels of testing are defined starting from unit level to system level testing. Unit level testing ensures the quality of single unit. Integration level testing checks the quality of different interfaces defined for communication of different components, ensures proper message passing and behavior after integration are verified in integration testing. System level testing maps the systems with the requirements and functionalities what the system intends for. In each level of testing, the important component of software testing is test data, without adequate test data execution of the software system is not possible. In other words, we can say that software testing is not possible without adequate test data. Different manual and automated approaches have been proposed for the generation of test data. Manual test data generation is a tedious process and unable to handle the testing data requirements at a higher level of testing. Most of the approaches used for automatic test data generation use evolutionary approaches for test data generation. The application of evolutionary approaches to software testing is known as evolutionary testing [20, 21].

In this paper, we have applied our previous proposed approach [45, 46] for automatic test data generation for coupling based integration testing to a real time case study from industry for coupling based integration testing. Integration testing is concerned with the interactions among components. Does a component call other components correctly? Are the right parameters with right types and ranges are passed? Does the called method return the proper type and the value is in the correct range? These questions are focus of the integration testing. Unfortunately, very little research has been done in the area of integration testing. Coupling based integration testing is based upon coupling relationships that exist among variables across call sites in procedures. In the same way as unit level testing is a base for integration testing, integration test is a base for system level testing. System level testing is difficult to achieve before integration testing [20, 21]. The major contributions of this research paper are:

- Application of the proposed approach for generation of test data at the integration level as most of the work on test data generation is at unit level
- Identification of coupling relations from the large industrial application and based on those relationship testing scenarios are defined.
- Generation of test data for identified coupling scenarios
- Comparison of experimental results with random testing, in order to measure its effectiveness

The rest of the paper is organized as follows: Section 2 elaborates the background knowledge coupling based integration testing. Section 3 describes the chosen case study and section 4 represents the experimental scenarios and setup. Section 5 is related to empirical results and experimental measurements. Section 6 represents the related work. Section 7 concludes the paper and presents the future work.

II. INTRODUCTION TO CASE-STUDY

To determine the potential effectiveness of our approach and prototype tool, *E-Coup Testing*, over random test data generation approach, a case study was performed in telecommunication environment. We selected a large application where integration of different components is very frequent. We have also considered the requirement for coupling based testing in our selection of case study. We have analyzed the whole application before going into experimentation for test data generation and focus more on those components where coupling exists. We have identified all four coupling type relationships in our selected application and then perform experimentation to generate test data for all coupling types in order to test the complete functionality of our proposed approach and tool. Before going into the experimentation details, let's first discuss our selected application, important components of the application, and data flow between different components, different variable definition and use.

We have selected rating management application which is a part of intelligent networks. Rating application is a very important application as it manages the overall rating process between and after call of any subscriber. Different offers and promotions are launched using this application. There are different components of this application each having specific functionality. Test object and configurations described below are used for our testing purposes.

.TEST OBJECTS AND CONFIGURATIONS

In order to achieve the testing of RMA application at integration level, we have designed various promotions and offers in the application where coupling is involved. We have tested each promo using our proposed and generate test data after analyzing the trace after every iteration. Our designed offers and promotions contain different coupling types and we have generated successfully data for designed promotion and offer. We have designed following offers and configured in RMA for testing. Table 1 shows the designed offer name, their functionality, components used and coupling type used in the flow.

Bonus on Usage (BoU) offers for minutes, SMS and data are one of the complex offers in RMA as it involves interaction of many components. BoU offers are selected for testing because coupling Type 1 is involved in every BoU offer. Accumulators are used in BoU offers to accumulate the usage of the customer in terms of minutes, SMS, Data or money. Accumulators instance is created in accumulator component using current customer context object. Accumulators are defined in the rating component with initial and maximum values. Then accumulators are used and evaluated in Bonus component to assign bonus to customers. As three components are used in BoU offers; accumulator for instance creation, rating for a definition and bonus for usage of accumulators. So there is coupling type 1 exists in BoU offers. We defined and configured three BoU Offers for coupling type 1 testing.

Free minutes, SMS and Data bundles are used to give benefit to user by using subscription services mostly through SMS on some special number. These offers are testing because coupling Type 2 is involved in bundles offers. Data bundles are defined in dedicated accounts and used in rating components so there is coupling type 2 exists in bundle offers. Hybrid and BoR offers are used in rating component and defined in bonus component again after giving bonus to customer so there is coupling type 3 exists in these offers. Timer based offers are used and defined rating components so there is coupling type 4 for timer based offers.

Table 1. Scenarios Used for Testing

Offer #	Offer Name	Functionality	Component Used	Coupling Type
1	**BoU Minutes**	Subscriber will receive 10 free minutes after achieving usage of 500 minutes	Pre-Analysis, Bonus, Rating, Accumulator, Dedicated account	Coupling Type 1
2	**BoU SMS**	Subscriber will receive 50 free SMS after sending 500 SMS	Pre-Analysis, Bonus, Rating, Accumulator, Dedicated account	Coupling Type 1
3	**BoU Data**	Subscriber will receive 1 MB free SMS after data usage of 10 MB	Pre-Analysis, Bonus, Rating, Accumulator, Dedicated account	Coupling Type 1
4	**Free SMS Bundle**	Subscriber will receive 100 SMS per day for 1USD	Dedicated account, Rating, USSD Message	Coupling Type 2
5	**Free Minutes Bundle**	Subscriber will receive 50 Minutes per day for 5USD	Pre-Analysis, Dedicated account, Rating, USSD Message	Coupling Type 2
6	**Free Data Bundle**	Subscriber will receive 2 MB per day for 3USD	Pre-Analysis, Dedicated account, Rating, USSD Message	Coupling Type 2
7	**Hybrid Offer**	This offer is mixture of SMS, minutes and Data	Pre-Analysis, Dedicated account, Rating, USSD Message, Bonus	Coupling Type3
8	**Hybrid BoU**	Subscriber will receive 50 free SMS, Minutes and 5MB after usage 10 USD	Pre-Analysis, Bonus, Rating, Accumulator, Dedicated account, Bonus	Coupling Type 3
9	**BoR Minutes**	On every recharge customer will receive 10 minutes	Bonus, Rating, Dedicated Account	Coupling Type 3
10	**Time Specific Offer**	There will be no charging in some specific hour of the Day	Pre-Analysis, Rating	Coupling Type 4

III. EXPERIMENTAL PROCEDURE

Testing flow of RMA application is shown in Figure 4. In the first step, data configurations are done on RMA. Data Configurations involve development of all offers to be tested on RMA application and then all these testing scenarios are stored in a database and will be loaded into memory at start of RMA application. After data configurations, each scenario is simulated using the simulator for execution of testing cycles. Simulator executes each testing scenario using configuration stored in RMA application.

After Execution of each cycle trace file is generated for each scenario. This file is very important for our testing purposes. This file contains the same information as one can get by using instrumenting the code. By enabling trace on RMA means that we are getting all information of every component involved in execution and all variables along with required and actual values for certain types of conditions to be true. We can check the trace file for execution of offers and simulation of various scenarios in different offers. If all required scenarios are executed then we stop the execution and test data used for execution is stored in files.

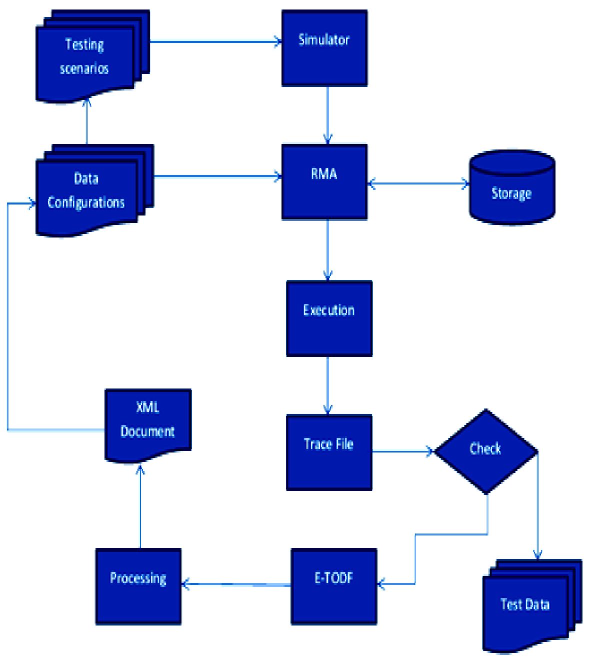

Figure 4. Testing Flow of Application using ECOUP

If all the required scenarios are not executed, then we pass the trace file to ECOUP our proposed tool for test data generation. Our proposed tool parsed the file; extract the information for each accumulator, dedicated account, accumulation counter and bonus calculator. Compare the actual and required values, calculates the new values by using cost function proposed by [6]. After processing of trace files, each file is written in XML file in the required format accepted by RMA. After loading of new values in RMA by XML then execution is performed again and the same steps are repeated until all scenarios are complete with required test data.

IV. EXPERIMENTAL MEASUREMENTS

In order to prove the effectiveness of our approach, we defined various measurements for comparison of our approach with random test data generation. We have compared our approach with random test data generation based upon the following information for each test object:

- Success Rate
- Average Coverage
- Failure Rate

Failure rate is defined as:

Failure Rate= (Unsuccess Searches/Total Searches) *100

Success rate is defined as:

Success Rate= (Successful Searches/Total Searches) *100

Average number of generations for a successful search of data is the average number of generation came from different experiments on different test object based on coupling type. We have compared the average number of generations of our approach with random test data generation. Our approach has much better results as compared to random testing.

Maximum time for a successful search is the maximum time required by any test object for the generation of test data. We have compared our approach's maximum time with random test data generation. Our approach has much better results as compared to random testing.

We identified ten test objects, but up till now we have performed our experiments with only three BOU test objects and our proposed approach has much better results as compared to random testing. The detailed parameters used during the testing are shown in table 2.

Table 2. Testing Parameters

Parameter	Values
Population Size	80
Number of Generations	400-600
Mutation Rate	0.2
Crossover Rate	0.8
Termination Criteria	Coverage>90% or Generation=600

Table 3. Average Coverage of Proposed Approach

Number of Generations	Coverage			Average Coverage
	BoU Minutes	BoU SMS	BoU Data	
450	77%	73%	78%	76%
475	82%	78%	82%	80%
500	85%	81%	83%	83%
550	87%	82%	83%	84%
600	88%	82%	84%	84%

Table 4. Average Success Rate and Failure of Proposed Approach

Number of Generations	Success Rate			Avg. Success Rate	Avg. Failure Rate
	BoU Minutes	BoU SMS	BoU Data		
100	34%	40%	35%	36%	64%
100	36%	41%	36%	37%	63%
100	38%	37%	32%	35%	65%
100	33%	35%	37%	36%	64%
100	36%	32%	41%	36%	64%

Table 5. Average Coverage of Random Testing

Number of Generations	Coverage			Average Coverage
	BoU Minutes	BoU SMS	BoU Data	
450	25%	23%	21%	23%
475	28%	28%	24%	26%
500	36%	32%	26%	31%
550	41%	33%	32%	35%
600	44%	34%	31%	36%

Table 6 .Average Success and Failure of Random Testing

Number of Generations	Success Rate			Average Success Rate	Average Failure Rate
	BoU Minutes	BoU SMS	BoU Data		
100	8%	11%	13%	10%	90%
100	10%	9%	10%	9%	91%
100	7%	8%	9%	8%	92%
100	9%	12%	9%	10%	90%
100	6%	13%	8%	9%	91%

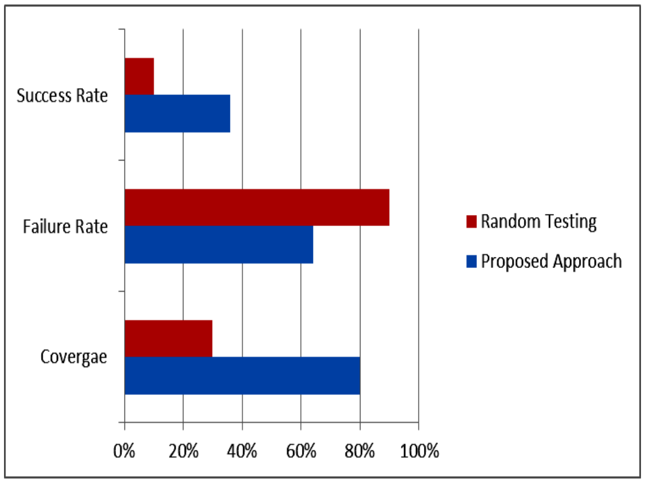

Figure 5. Comparison of Experimental Results

From our experimental results, it has been concluded that our approach has much better results as compared to random testing as shown in figure 5.

V. RELATED WORK

Tonella [16] used evolutionary algorithm for testing of object oriented program at unit level. McMinn and Holcombe [9] used ant colony optimization for the solution of state issues in object oriented programs. McMinn and Holcombe [9] used extended chain approach for the resolution of state problem in object oriented program. Watkins [17] experiments confirmed that branch predicate and inverse path probability based fitness functions are more accurate and correct. Wegener et al. [18] [19] developed an automated framework for structural testing of software programs and their approcahe generates auotamated test data for execution of test cases. Baresel et al. [1] peforms several experiments for improvement of fitness function inorder to minimize number of iterations and time for generation of test data. McMinn [10] peformed a comprehensive analysis of test data generation techniques using evolutionary approaches and suggests various directions for future work.

Test data generation for unit testing of software system are discussed in the following approches [8, 11, 13, 14, 15, 17] using genetic algorithms. Cheon et al. [3, 4] suggested a fitness function and applied the fitness function on unit testing of Java program. Dharsana et al. [5] used genetic algorithm for the optimization of test cases for java programs. Jones et al. [6]apllied genetic algorithm for white box testing of software testing. Bilal and Nadeem [2] used genetic algorithm to cater state problem in object oriented program and proposed fitness function for the solution of state probelm.

Smith and Robson used classes instances in their testing approach [23]. Fiedler apllied both structural and specification based techniques in his tesing approcah[22]. Edwards used specification for the generation of test cases[28]. Perry and Kaiser [29] suggested that integration testing is more challenging in object oriented programs. Jorgensen and Erickson describe an approach to integration testing that is similar to many black box testing techniques [14]. Object oriented testing at intra calss level is described by the work of Hong et al. [30], Parrish et al. [31], Turner and Robson [32], Doong and Franklin [33], and Chen et al. [34]. Chen and Kao [26] prosed a technique call object flow graph in their work for testing of object oriented program. Alexander and Offutt [42, 38] proposed techniques for coupling based integration tesing of object oriented programs. S.A khan and Nadeem [45, 46,47] apply genetic algorithm and particle swarm optimization to data flow testing at unit and integration level for test data generation.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have carried out a case study on real time industry application. We have identified various test objects, their coupling types and based on our experimental measurements, we have performed experiments. On the basis of empirical results, we claim that our approach is more efficient than random testing. In future more experimentation will be performed on remaining test objects before any conclusive statements can be made.

REFERENCES

[1] Baresel, A., Sthamer, H., Schmidt, M., (2002 July) "Fitness Function Design to improve Evolutionary Structural Testing", Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 02), New York (NY), USA.

[2] Bilal, M., Nadeem, A., (2009 April) “A State based Fitness Function for Evolutionary Testing of Object-Oriented Programs”. Studies in Computational Intelligence, 2009, Volume 253/2009, 83-94, DOI: 10.1007/978-3-642-05441-9. Software Engineering Research, Management and Applications 2009

[3] Cheon, Y., Kim, M. Y., Perumandla, A., (2005 June) "A Complete Automation of Unit Testing for Java Programs", The 2005 International Conference on Software Engineering Research and Practice (SERP), Las Vegas, Nevada, USA.

[4] Cheon, Y., Kim, M., (2006 July) "A specification-based fitness function for evolutionary testing of object-oriented programs", Proceedings of the 8th annual conference on Genetic and evolutionary computation, Washington, USA.

[5] Dharsana, C.S.S., Askarunisha, A., (2007 December) "Java based Test case Generation and Optimization Using Evolutionary Testing". International Conference on Computational Intelligence and Multimedia Applications,Sivakasi, India.

[6] Jones, B., Sthamer, H., Eyres, D., (1996) "Automatic structural testing using genetic algorithms", Software Engineering Journal, vol. 11, no. 5, pp. 299 – 306.

[7] Liaskos, K., Roper, M., Wood, M., (2007 July) "Investigating data-flow coverage of classes using evolutionary algorithms", Proceedings of the 9th annual conference on Genetic and evolutionary computation, London, England.

[8] McGraw, G., Michael, C., Schatz, M., (2001) "Generating software test data by evolution." IEEE Transactions on Software Engineering, 27(12):1085—1110.

[9] McMinn, P., Holcombe, M., (2003 July) "The state problem for evolutionary testing." In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Lecture Notes in Computer Science vol. 2724, pages 2488-2497, Chicago, USA. Springer-Verlag.

[10] McMinn, P., (2004) "Search-based Software Test Data Generation: a Survey", Journal of Software Testing, Verifications, and Reliability, vol. 14, no. 2, pp. 105-156, June.

[11] Pargas, R., Harrold, M., Peck, R., (1999) "Test-data generation using genetic algorithms. Software Testing", Verification and Reliability, 9(4):263-282.

[12] Roper, M., (1997 May) "Computer aided software testing using genetic algorithms." In 10th International Software Quality Week, San Francisco, USA.

[13] Sthamer, H., (1996) "The automatic generation of software test data using genetic algorithms", PhD Thesis, University of Ghamorgan, Pontypridd, Wales, Great Britain.

[14] Seesing, A., Gross, H., (2006) "A Genetic Programming Approach to Automated Test Generation for Object-Oriented Software", International Transactions on Systems Science and Applications, vol. 1, no. 2, pp. 127-134.

[15] Tracey, N., Clark, J., Mander, K., McDermid, J., (2000) "Automated test-data generation for exception conditions", SOFTWARE—PRACTICE AND EXPERIENCE, vol., Pages 61-79, January.

[16] Tonella, P., (2004 July) "Evolutionary Testing of Classes", In Proceedings of the ACM SIGSOFT International Symposium of Software Testing and Analysis, Boston, MA, pp. 119-128.

[17] Watkins, A., (1995 July) "The automatic generation of test data using genetic algorithms." In Proceedings of the Fourth Software Quality Conference, pages 300--309. ACM, 1995.

[18] Wegener, J., Baresel, A., Sthamer, H., (2001) "Evolutionary test environment for automatic structural testing." Information and Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms, 43 pp.841–854.

[19] Wegener, J., Buhr, K., Pohlheim, H., (2002 July) "Automatic test data generation for structural testing of embedded software systems by evolutionary testing", In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pages 1233-1240, New York, USA. Morgan Kaufmann.

[20] Lee Copeland, "A Practitioner's Guide to Software Test Design", STQE Publishing, 2004.

[21] Boris Beizer, "Software Testing Techniques", International Thomson Computer Press, 1990.

[22] Fiedler SP. Object-oriented unit testing. Hewlett-Packard Journal 1989; 40(2):69–75.

[23] Smith MD, Robson DJ. Object-oriented programming: The problems of validation. Sixth International Conference onSoftware Maintenance. IEEE Computer Society Press: Los Alamitos, CA, 1990; 272–282. Overbeck J. Integration testing for object-oriented software. PhD Dissertation, Vienna University of Technology, 1994.

[24] Pande HD, Landi WA, Ryder BG. Interprocedural def–use associations for C systems with single level pointers. IEEE Transactions on Software Engineering 1994; 20(5):385–403.

[25] Chen M-H, Kao M-H. Testing object-oriented programs—An integrated approach. Proceedings of the 10th International Symposium on Software Reliability Engineering. IEEE Computer Society Press: Boca Raton, FL, 1999; 73–83.

[26] Kung D, Gao J, Hsia P, Toyoshima Y, Chen C. A test strategy for object-oriented systems. Nineteenth Annual International Computer Software and Applications Conference. IEEE Computer Society Press: Los Alamitos, CA, 1995; 239–244.

[27] Edwards SH. Black-box testing using flowgraphs: An experimental assessment of effectiveness and automation potential. Software Testing, Verification and Reliability 2000; 10(4):249–262.

[28] Perry DE, Kaiser GE. Adequate testing and object-oriented programming. Journal of Object-oriented Programming 1990; 2(5):13–19.

[29] Hong HS, Kwon YR, Cha SD. Testing of object-oriented programs based on finite state machines. The 1995 Asia Pacific Software Engineering Conference. IEEE Computer Society Press: Los Alamitos, CA, 1995; 234–241.

[30] Parrish AS, Borie RB, Cordes DW. Automated flow graph-based testing of object-oriented software modules. Journal of Systems and Software 1993; 23(2):95–109.

[31] Turner CD, Robson DJ. The state-based testing of object-oriented programs. Conference on Software Maintenance. IEEE Computer Society Press: Los Alamitos, CA, 1993; 302–310.

[32] Doong R-K, Frankl P. Case studies on testing object-oriented programs. Fourth Symposium on Software Testing, Analysis and Verification. ACM Press: New York, 1991; 165–177.

[33] Chen HY, Tse TH, Chan FT, Chen TY. In black and white: An integrated approach to class-level testing of object-oriented programs. ACM Transactions on Software Engineering and Methodology 1998; 7(3):250–295.

[34] Meyer B. Object-Oriented Software Construction (2nd edn). Prentice-Hall: Englewood Cliffs, NJ, 1997.

[35] Harrold MJ, Rothermel G. Performing data flow testing on classes. Second ACM SIGSOFT Symposium on Foundations of Software Engineering. ACM Press: New York, 1994; 154–163.

[36] Alexander RT. Testing the polymorphic relationships of object-oriented components. Technical Report ISE-TR-99-02, Department of Information and Software Engineering, George Mason University, February 1999.

[37] Alexander RT, Offutt J. Analysis techniques for testing polymorphic relationships. Thirtieth International Conference on Technology of Object-oriented Languages and Systems (TOOLS30), Santa Barbara, CA, 1999; 104–114.

[38] Frankl PG, Weyuker EJ. An applicable family of data flow testing criteria. IEEE Transactions on Software Engineering 1988; 14(10):1483–1498.

[39] Rapps S, Weyuker WJ. Selecting software test data using data flow information. IEEE Transactions on Software Engineering 1985; 11(4):367–375.

[40] Alexander RT. Testing the polymorphic relationships of object-oriented programs. Dissertation, George Mason University, 2001.

[41] Alexander RT, Offutt J. Criteria for testing polymorphic relationships. Proceedings of the International Symposium on Software Reliability and Engineering (ISSRE00). IEEE Computer Society:SanJose,CA,2000.

[42] Zhenyi Jin and A. Jefferson Offutt, Coupling-based Crite-ria for Integration Testing. The Journal of Software Test-ing, Verification, an d Reliability, 1998.8(3): p. 133-154.

[43] Xiyang Liu., Miao Zhang, Zhiwen Bai, Lei Wang, Wen Du, Yan Wang.Function Call Flow based Fitness Function Design in Evolutionary Testing.APSEC '07 Proceedings of the 14th Asia-Pacific Software Engineering Conference, Pages 57-64, Nagoya Japan,Dec5-7,2007.

[44] A. Baresel, H. Sthamer, and M. Schmidt. Fitness func-tion design to improve evolutionary structural test-ing. In Proceedings of the Genetic and Evolution-ary Computation Conference (GECCO'02), pp.1329-1336. New York, USA, July 2002.

[45] S.A Khan, A. Nadeem, "Automated Test Data Generation for Coupling Based Integration Testing of Object Oriented Programs Using Particle Swarm Optimization (PSO)", Proceedings of the Seventh International Conference on Genetic and Evolutionary Computing, ICGEC 2013, August 25 - 27, 2013 - Prague, Czech Republic.

[46] SA Khan, A Nadeem, "Automated Test Data Generation for Coupling Based Integration Testing of Object Oriented Programs Using Evolutionary Approaches",Proceedings of the 2013 10th International Conference on Information Technology: New Generations (ITNG 2013), Pages 369-374, LAS Vegas, Nevada, USA.

[47] SA Khan, A Nadeem, "Applying Evolutionary Approaches to Data Flow Testing at Unit Level", Proceedings of the International Conference on advanced software engineering and its application, 2011,(ASEA, 2011) Jeju Island, Korea, December 8-10, 2011.