VOL. 10, NO. 3, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

BRANCH COVERAGE BASED TEST CASE PRIORITIZATION

Arnaldo Marulitua Sinaga
Department of Informatics, Faculty of Electronics and Informatics Engineering, Institut Teknologi Del, District Toba Samosir (Tobasa), North Sumatera, Indonesia
E-Mail: aldo@del.ac.id

ABSTRACT

Software testing is aimed to detect existing faults in a software. The nature of software shows that modification is unavoided. Testing of a modified software is a must to ensure that the software is still free of failures. This process is named as regression testing. Regression testing can be very expensive if all test cases have to be re-tested. To reduce the cost, it is important to prioritize the test case execution to enhance the capability of detecting failures. Test case prioritization is intended to schedule and order the execution of test case based on the certain criteria. In this research, four test case prioritization methods studied emperically are additional branch coverage prioritization, Manhattan distance-based ART (Adaptive Random Testing), additional branch coverage-based with ART, and ART with additional branch coverage-based. Random Testing, as the basic test selection method, is used as a benchmark of the performance of all studied methods. The conducted experiments using two programs as under test program are Replace and Space programs. The experiment results show that all studied methods improve the effectiveness of RT significantly for large program. The used effectiveness measurement is F-measure, the number of test cases executed to detect the first failure. The additional branch coverage-based with ART comes as the best method in terms of F-measure. This method combines the advantage of the additional branch coverage method and the ART. It also reduces the complexity of the additional branch coverage.

Keywords: test case prioritization, adaptive random testing, regression testing, Manhattan distance.

INTRODUCTION

Software testing is a necessary activity in software development. The quality of software can be assured by software testing process. However the cost of software testing can be very expensive. Proper and effective testing is important to reduce the cost of software [1]. Software testing also needs to be conducted when software is modified. The modification may introduce new faults which need to be revalidated. This process is called as regression testing. In regression testing, retesting all test cases will be very costly. Therefore, selecting effective test cases is very important to reduce the cost of testing. An effective test case is a set of test cases with high fault detection capability [2]. The earlier failures detected, the earlier the correspondent faults can be fixed, and hence reduce the cost of software development and maintenance.

There are two approaches to conduct test case selection: black-box and white-box testing [3]. Black box testing is a method that examines the functionality of software whereas the white box testing is a method that examines the internal structure of software [1]. Coverage-based testing is one of white box testing methods that selects test cases based on the coverage of each test cases. There are some criteria that have been used such as statement coverage testing, branch or decision coverage testing, condition coverage, decision or condition coverage, modified condition/ decision coverage, and multiple-condition coverage [4]. Branch or decision coverage is one of the most used in test case prioritization research [5,6,7]. Test case prioritization is a method to order or schedule test cases so that the highest priority test case will be executed earlier than lower one. In branch coverage, the test cases are selected to maximize the execution of branches in the source code of the program under test.

Rothermel et al. [5], found that Additional branch coverage prioritization is an effective coverage-based test case prioritization. Additional branch coverage (known as additional in the remaining of this paper) is a technique that orders test cases which comprise most coverage of uncovered branch (by previously selected/executed test cases) [5]. On the other hand, Chen et al. introduced Adaptive Random Testing (ART) to improve the fault-detection capability of Random Testing (RT) to generate, select or prioritize test cases [3]. RT is the most fundamental technique in software testing. All test cases have uniform probability to be selected. ART is proposed by Chen et al. [3] with the intuition that the similar test cases are clustered in the same area of domain, hence it is better to select test cases from different area to maximize the coverage of test cases. Jaygarl et al. [8] stated that ART is an effective method to generate test cases. The difference or distance between test cases is one of the main issues in ART. For numeric program (input type is numeric), distance of test case t1 and t2 can be calculated by t2 – t1. Euclidean distance has been used for numeric program but difficult to be implemented for non-numeric program. Zhou [6] proposed methods to measure the distance between test cases for non-numeric program by using Manhattan Distance and Jaccard distance. Zhou [6] and Zhou et al. [7] found that ART with Manhattan distance performed significantly better than RT.

Since the results of Rothermel [5], Zhou [6] and Zhou et al. [7] show that additional and ART performed well for test case selection, in this research the combination of these two methods is investigated. Two types of combination are proposed to produce two new

techniques in test case prioritization: additional branch coverage-based with ART and ART with additional branch coverage-based. The hypothesis is that these two techniques will combine the advantages of each combined method, namely additional and ART.

STUDIED METHODS

In this research, there are five test case prioritization methods being experimentally studied: RT, additional, Manhattan distance-based ART, additional with ART, and ART with additional. RT is a simple and applicable technique. Test cases are selected randomly. As the most fundamental technique, RT is implemented as a benchmark for all studied methods.

Additional is a technique of test case prioritization, that gives high priority to test case with the most additional branch coverage of the uncover branch by the previously executed test cases. As proposed by Rothermel et al. [5], the algorithm of this technique is as follows: (i) select a test case randomly from test suite, (ii) all branches that have been covered are marked, (iii) count the number of unmarked branches that touched/covered (additional coverage) by each remaining test cases (test cases that not yet selected), (iv) select test case with the highest additional coverage, (v) mark all covered branches, (vi) check the coverage, if all branches have been covered then reset the branch coverage information and go to step (i), otherwise go to step (iii).

ART method studied in this research is a method proposed by Zhou [6]. This method adopts the Fixed-Size Candidate Set (FSCS) ART with ten candidates (as recommended by Chen et al. [3]). The distance between test cases is calculated by using branch-coverage Manhattan distance. The algorithm of this technique is as follows: (i) the first test case is selected randomly, (ii) all the executed test cases are stored in a set, (iii) the next test case is selected by sampling a fixed size of candidate set randomly, (iv) the minimum distance of each candidate from the executed test cases is calculated, (v) the next test case to be executed is candidate with the maximum distance, (vi) the executed test case is added to the set of executed test case and all other candidates are discarded, (vi) if the stopping criterion is not satisfied, this process is repeated. The distance measurement in this technique applies Manhattan distance [6]. During the execution of a test case, branch coverage is observed. In this technique if a branch is executed/covered then its flag is set to 1, otherwise is set to 0. The formula to calculate Manhattan distance between test case x and y that is used for program with n branches is as follows:

$$MD(x,y) = \sum_{i=1}^{n} | x_i - y_i | \quad (1)$$

where

x_i = The flag of test case x for branch-i

y_i = The flag of test case y for branch-i

For illustration, Table-1 lists branch-coverage information of five test cases (Tc1 to Tc5), assume there are four branches in the program under test.

Tabel-1. Branch-coverage information.

Branch-Id	Tc1	Tc2	Tc3	Tc4	Tc5
1	1	1	1	1	0
2	1	0	0	1	1
3	0	0	1	0	0
4	1	1	1	0	1

By using coverage information in Table-1, the coverage of Tc1 is {1,1,0,1} and Tc2 is {1,0,0,1}, the distance between Tc1 and Tc2 is 1.

The Additional with ART combines the additional and the ART technique. The first round of additional is conducted, that is performed step (i) to (vi) of additional algorithm until the first set of test cases that covered all branches obtained. The remaining test cases are prioritized by using ART method, that performed step (i) to (vi) of ART algorithm until all test cases are ordered.

The ART with Additional combining the ART and the additional All of the steps in ART are performed in this technique, but different in distance measurement. Instead of using Manhattan distance, in this technique the distance between test cases is calculated by using additional coverage. By using information in Table-1, distance between Tc1 to Tc2 is 1 because there is additional coverage by Tc2 to Tc1, that is for branch-2, whereas distance between Tc1 and Tc2 is 0 because there is no additional coverage of Tc1 to Tc2.

RESEARCH METHODS

The studied methods are experimentally investigated. This empirical method uses two programs under test, namely Replace and Space. These two programs have been used in many research of software testing. All the instruments of these two programs are downloaded from Software-artifact Infrastructure Repository (SIR) [9], a website that provides instruments for the experiments in software testing.

Replace produced by Siemens Corporate Research is applied to replace a pattern of regular expression. Replace program consists of 564 lines of C code with 180 branches and 21 functions. In Replace package, it has 32 faulty versions, and 5,542 test cases. The experiments only involve 28 versions. Four versions, #13, #23, #26, and #32, are not stable when executed. Their outputs are different in different execution. Thus, it is impractical to find the failure causing input.

Space program read a file that consists of some ADL (Array Definition Language), and check the correctness and consistency of the ADL. Space Package is also downloaded from SIR. In that package it has 38 faulty versions. Each of the faulty versions has a fault found in

the development process. The program of Space consists of 6,199 lines of C code with, 1.190 branches and 136 functions. The test suite of this program consists of 13,585 test cases [9]. In the experiments, faulty versions #1, #2, #3, and #4 are not involved. The outputs of these versions are the same as the output of the original version. It indicates that all test cases in test suite can not detect any failures in those faulty versions, hence are not included in the experiments.

THE EXPERIMENTS

The pre-process of the experiments is the activity to obtain all Failure Causing Input (FCI) of each faulty version and coverage element of each test case. FCI is obtained by comparing the output of a corresponding faulty version with the output of the original version. If the output of the faulty version is different from the output of the original version then a failure is detected, and consequently the test case being executed is an FCI.

Since the studied methods applying coverage information and obtaining the coverage element of each test case are very important issue, coverage element is used to calculate the distance between test cases. It shows which branch has been covered by a test case. Since the under test programs are c-program, gcov is used to get the coverage information. Information regarding count_execution resulted by gcov is used and processed to produce a text file containing the flag of 1 or 0. It means the frequency of a branch being touched/executed is not considered.

After the FCI and the coverage information are obtained. The five studied methods are executed to the two under test programs, Space and Replace. Test cases in their test suites are prioritized by each of the studied methods. The FCI is compared to the ordered test suite. The first FCI found in the ordered test suite is known as the F-measure. The F-measure is the number of selected or executed test case until the first failure is detected.

To increase the validity of the result, each method is executed for 100 trials. Therefore, there are 500 trials for each of the under test program. The average of all trials for each faulty versions is used to compare the performance of studied methods.

RESULTS AND DISCUSSIONS

In this section, the findings for Replace and Space program are explained.

Result of the experiments with replace program

The results of the experiments with Replace program are presented in Table-2. The average of F-measures of each studied method to each of faulty versions is provided. The shaded cells indicate that the corresponding method performed worse than RT for the corresponding faulty version.

Tabel-2. Results with replace program.

V	*F-measure*				
	RT	***Add***	**ART**	**Add ART**	**ART Add**
1	80.1	5.9	39.3	45.9	73.1
2	165.6	5.9	83.4	73.7	215.4
3	42.8	16.1	10.3	5.9	46.4
4	40.5	15.9	9.6	5.9	42.0
5	25.9	13.8	15.9	20.4	18.3
6	59.7	4.1	60.0	3.9	46.3
7	60.1	120.6	21.5	80.2	66.2
8	90.8	136.5	45.7	114.1	95.3
9	154.4	231.9	174.0	220.0	206.6
10	247.6	268.0	189.6	239.0	230.3
11	154.4	231.9	174.0	220.0	206.6
12	17.5	12.3	16.6	18.1	17.7
14	94.0	371.0	17.3	89.5	99.9
15	60.1	120.6	21.5	80.2	66.2
16	201.8	302.5	195.2	191.7	235.3
17	24.4	34.0	21.0	30.3	25.0
18	1,490.5	422.0	594.4	1,279.6	1,232.1
19	211.7	303.6	217.7	202.7	244.9
20	1,524.9	2,540.0	800.8	1,576.5	1,483.6
21	1,524.9	2,540.0	800.8	1,576.5	1,483.6
22	281.4	249.2	281.1	288.3	256.2
24	18.6	60.1	8.7	24.5	21.0
25	1,479.3	990.5	762.1	1,393.0	1,506.2
27	19.5	11.7	7.0	5.2	18.3
28	24.4	34.0	21.0	30.3	25.0
29	74.4	39.6	27.5	113.7	85.4
30	19.5	11.7	6.9	5.2	18.4
31	24.4	34.0	21.0	30.3	25.0
Avg	293.3	326.0	168.0	284.4	285.7

In Table-2, column RT presents the F-measure of RT calculated as the average of 100 trials. Column Add describes the F-measure obtained from additional branch method. Column Add ART shows the F-measure obtained from the combination of additional branch and ART. Column ART Add shows the F-measure obtained from the combination of ART and additional branch. The F-measure results show that the ART is the best, followed by Add-ART and ART-Add.

From the F-measure result, ratio of the improvement of the four investigated methods to RT is calculated. The ratio for each faulty versions is presented in Table-3.

Tabel-3. Ratio of comparison to RT with replace (%).

V	Add	ART	Add ART	ART Add
1	7.32	49.13	57.37	91.27
2	3.55	50.35	44.51	130.06
3	37.46	24.11	13.66	108.22
4	39.19	23.74	14.44	103.53
5	53.08	61.22	78.76	70.39
6	6.93	100.49	6.60	77.54
7	200.78	35.85	133.47	110.29
8	150.32	50.29	125.66	104.99
9	150.17	112.69	142.45	133.76
10	108.21	76.55	96.51	92.99
11	150.17	112.69	142.45	133.76
12	70.27	94.80	103.72	101.43
14	394.86	18.42	95.22	106.38
15	200.78	35.85	133.47	110.29
16	149.91	96.74	94.97	116.59
17	139.31	85.91	124.08	102.33
18	28.31	39.88	85.85	82.67
19	143.44	102.84	95.75	115.69
20	166.57	52.52	103.38	97.29
21	166.57	52.52	103.38	97.29
22	88.56	99.89	102.47	91.05
24	323.41	46.56	132.08	112.81
25	66.96	51.52	94.16	101.82
27	59.75	35.84	26.68	93.45
28	139.31	85.91	124.08	102.33
29	53.24	36.91	152.76	114.71
30	59.75	35.43	26.68	94.01
31	139.31	85.91	124.08	102.33
Avg	117.77	62.66	92.10	103.55

Some findings from Table-3.

1. Additional performed better than RT 13 faulty versions except for versions #7, #8, #9, #10, #11, #14, #15, #16, #17, #19, #20, #21, #24, #28, #31. The best performance of Additional is found from version #2, the ratio is 3.55% which means that the Additional improved the RT with the gained percentage is 96.45% (uses 96.45% less test cases than RT to detect a failure). Overall the Additional performed worse than RT, the ratio 117%, which means that Additional uses more test cases than RT.
2. ART outperformed RT with most of the faulty versions except for versions #6, #9, #11 and #19. The differences between ART and RT for those four versions are relatively small. The biggest saving of ART is obtained from version #14 with the ratio of 18.42%, indicates that ART uses 81.58% test case less than RT to detect failure. Overall the ratio of F-measure for ART to RT is 62%.
3. The combination of Additional and ART performed better than RT 14 faulty versions and worse for version #7, #8, #9, #11, #12, #15, #17, #20, #21, #24, #28, #29 and #31. The biggest saving is obtained with version #6, that 93.40% less than RT. Overall the ration of Additional with ART is 92%.
4. The combination of ART and Additional performed better than RT only for 10 faulty versions: #1, #5, #6, #10, #18, #20, #21, #22, #27 and #30. This method mostly performed worse than RT. The best performance obtained with version #5 with the saving of 31.61% (ratio of 70.39 %). Overall the ratio of this method compared to RT is 103%.

This research was analysed by using statistical analysis, that is by conducting one way ANOVA and paired sample test. The ANOVA test is used to analyse whether the compared methods are significantly different. The p-value resulted from ANOVA test is 0.00 (less than 0.05), which indicates that the studied methods are significantly different. To analyse the difference of each studied methods, the paired sample test is conducted. The results of paired sample are presented in Table-4.

Table-4. Paired sample t-test for replace.

No.	Pair	p-value
1	RT-ART	0.02
2	RT-ADD	0.64
3	RT-ADDART	0.40
4	RT-ARTADD	0.68

Table-4 indicates as follows:

1. The p-value resulted by paired-sample t-test between RT and ART is 0.02. This indicates that ART is significantly outperformed RT.
2. The paired-sample t-test between RT and Additional return a p-value of 0.64 (larger than 0.05). This indicates that the two techniques are not significantly different.
3. The paired-sample t-test between RT and Additional with ART return a p-value of 0.40 (larger than 0.05). This indicates that the two techniques are not significantly different.
4. The paired-sample t-test between RT and ART with Additional return a p-value of 0.68 (larger than 0.05). This indicates that the two techniques are not significantly different.

Result of the experiments with space program

The experiments results for Space program are presented in Table-5.

Tabel 5. Results with space program.

V	*F-measure*				
	RT	**Add**	**ART**	**Add ART**	**ART Add**
5	21.2	22.9	19.5	24.6	39.6
6	1.7	1.1	1.1	1.1	1.1
7	3.2	2.4	2.8	2.3	3.3
8	1.0	1.0	1.1	1.1	1.1
9	83.3	8.6	21.3	7.9	70.4
10	132.4	19.1	42.2	17.7	119.5
11	2.9	2.0	2.4	1.8	3.5
12	9.9	4.4	6.5	4.1	9.9
13	11.9	5.5	8.8	6.0	9.2
14	419.2	29.3	156.0	30.4	369.6
15	18.0	3.0	8.9	3.0	15.1
16	7.3	2.3	4.2	2.3	8.4
17	4.6	1.8	2.6	1.8	4.2
18	25.7	2.9	7.2	2.7	20.8
19	61.7	97.7	74.5	107.3	57.7
20	419.2	29.3	156.0	30.4	369.6
21	9.5	3.5	9.9	3.5	10.6
22	59.3	79.0	20.3	78.9	56.0
23	59.3	79.0	20.3	78.9	56.0
24	189.1	11.7	54.0	13.1	65.5
25	51.4	23.3	17.3	22.5	41.2
26	19.8	2.9	6.3	2.8	11.9
27	2.9	1.8	2.4	1.9	3.7
28	9.5	2.3	4.2	2.4	12.1
29	374.1	25.2	91.9	23.3	336.1
30	1.9	1.7	1.7	1.6	2.1
31	19.6	8.1	9.4	9.0	17.7
32	1.2	1.2	1.2	1.2	1.3
33	8.4	2.8	5.8	2.8	8.5
34	378.4	473.3	279.3	568.5	178.3
35	59.0	58.6	18.5	56.7	53.6
36	126.3	9.1	123.6	9.3	127.0
37	160.2	3.8	39.6	3.6	103.1
38	378.5	2.0	87.2	21.2	300.8
Avg	92.1	30.7	38.5	33.7	73.2

Results on Table-5 shows that all studied methods outperformed the RT. It also indicates that the performance of all studied methods with Space is better than performance with Replace. The Additional performed the best with the lowest F-measure, followed by the combination of Additional and ART, ART and the combination of ART and Additional comes as the worst.

As the basic strategy, RT is used as the benchmark for all methods. The ratio of comparison between each of studied methods to RT is presented in Table-6.

Table 6. Ratio of comparison to RT with space (%).

V	Add	ART	Add ART	ART Add
5	108.06	91.70	115.75	186.66
6	99.06	100.00	103.77	104.72
7	75.00	87.03	73.42	103.80
8	100.00	100.96	102.88	104.81
9	10.35	25.53	9.50	84.56
10	14.41	31.88	13.33	90.24
11	69.90	84.43	63.32	120.76
12	44.61	65.86	41.39	100.00
13	46.34	74.18	50.55	76.96
14	6.99	37.21	7.24	88.17
15	16.42	49.36	16.81	83.69
16	32.00	57.66	31.72	115.86
17	38.65	56.55	40.17	91.92
18	11.30	27.89	10.56	81.18
19	158.47	120.87	173.96	93.55
20	6.99	37.21	7.24	88.17
21	37.00	104.55	36.68	111.52
22	133.24	34.27	133.07	94.53
23	133.24	34.27	133.07	94.53
24	6.17	28.53	6.91	34.62
25	45.27	33.60	43.65	80.03
26	14.41	31.90	14.36	60.06
27	63.01	81.85	65.75	125.68
28	24.05	43.70	25.21	126.79
29	6.73	24.56	6.22	89.86
30	87.83	88.36	83.60	110.58
31	41.38	48.11	45.66	90.20
32	103.48	103.48	100.87	116.52
33	33.18	68.72	32.58	100.95
34	125.08	73.83	150.24	47.12
35	99.25	31.40	96.03	90.85
36	7.23	97.84	7.32	100.51
37	2.36	24.71	2.26	64.38
38	5.81	23.04	5.59	79.47
Avg	53.16	59.56	54.43	95.10

Table-6. indicates as follows.

1. The Additional performed better than RT except for versions #5, #8, #19, #22, #23, #32 and #34. The biggest saving is obtained from version #37 with the ratio of 2.36%. It means that the Additional executes 97.64% test case less than RT to detect the first failure.
2. ART outperformed RT for most of the faulty versions except for versions #21, #19, #8 and #6, however the difference between ART and RT for those five versions are very small (almost similar). The biggest saving is obtained from the version #38 with the ratio of 23.04%.
3. The combination of the Additional and ART performed better than RT with most of the faulty versions, except for the versions of #5, #6, #8, #19, #22, #23, and #32. The best performance of this

method is obtained from version #37. It reduces the number of test cases execution to 97.74 % of RT.

4. The combination of ART and Additional performed better than RT for faulty versions #5, #6, #7, #8, #11, #12, #16, #21, #27, #28, #30, #32, and #33. The biggest saving of this method is for version #24 with the ratio of 34.62%, which means executed 65.38% test cases less than RT to detect the first failure.

As conducted for Replace program, the one way ANOVA test and the paired-sample test are also conducted for Space program for statistical analysis. The one way ANOVA test result shows that all studied methods are significantly different (the p-value is 0.00). The paired-sample test results are presented in Table-7.

Table-7. Paired sample t-test for space.

No.	*Pair*	*p-value*
1	RT-ART	0.01
2	RT-ADD	0.00
3	RT-ADDART	0.01
4	RT-ARTADD	0.01

The results on Table-7 indicates all studied methods outperformed the RT significantly. The p-value for all pairs with RT is less than 0.05.

The experiments with the two programs under tests, Replace and Space program show that there is different performance of the studied methods when applied to different type of program. The results with Space show that the studied methods performed better when applied to larger program with larger test suite.

The results also indicate that the Additional performed extremely well for some faulty versions such as versions #37, #38, #24, #29, #13, #20, #36 of Space program and versions #1, #2, #6 of Replace program (all with ratio of less than 10%). On the other hand, the Additional can be very bad such for faulty versions #19 of Space and versions #14 and #24 of Replace program. From the investigation, it is found that the Additional performed very well when the FCI is included in the first round of Additional algorithm.

Generally ART performed better than RT, but when it is compared to the Additional, overall the Additional is better. However in some cases such as for faulty versions #14 and #24 of Replace program and #22 amd #23 of Space program, there are conditions when the ART performed better than the Additional. This is in line with the intuition of this research. The combination of Additional and ART can combine the advantages of the two methods. The performance Additional with ART performed very well when the Additional performed well (see versions V37, #38, #36, #29, # 4, #20 and #13 of Space and version #6 of Replace), whereas it can improve the performance Additional when the Additional performed very bad (see versions #14 and #24 of Replace and versions #22 and #23 of Space). However the combination of ART with Additional can not improve the original ART or Additional.

CONCLUSIONS AND SUGGESTIONS

Five test case prioritization methods investigated experimentally in this research are RT, Additional, ART, additional with ART and ART with Additional. All the coverage based methods apply the branch coverage information. Empirically, it is found that for large program with large test suite (Space program), all four test case prioritization methods outperformed RT significantly in terms of F-measure. The four methods apply fewer test cases than RT to detect the first failure in the faulty versions. Different results are found for Replace, a smaller program with smaller test suite. Three studied methods, Additional, Additional with ART and ART with Additional, are not better than RT. The ART is the only branch-coverage technique that outperformed RT significantly. In conclusion, the effectiveness of branch-coverage methods is influenced by the size of the under test program and the test suite.

The results also indicate that the combination of Additional and ART can improve the Additional and ART by combining the advantages of these two methods. The performance of the combination of Additional and ART is more stable than the Additional and is better than ART. This method is suggested to be applied for test case prioritization of a large test suite of a large program. On the other hand, the combination of the ART and Additional is not suggested to be used since it does not perform any improvement.

However the validity of this experiment needs to be enhanced in the next research. The number of program under tests with vary types of program (size, test suite, language) need to be applied. It is also suggested to use real faulty versions obtained from the real development process of the program under test. Beside the F-measure, it is also important to measure the time complexity and space consumed by the proposed methods.

REFERENCES

[1] S. Pressman Roger. ”Software Engineering a practitioner’s approach”, McGraw-Hill Higher Education, 2001.

[2] G. J. Myers “The Art of Software Testing 2nd Edition”, New York: John Wiley & Sons, 2004

[3] T.Y. Chen H. Leung I.K. Mak “Adaptive Random Testing” M.J. Maher (Ed.): Springer-Verlag Berlin Heidelberg ASIAN 2004, LNCS 3321, 2004, pp. 320–329.

[4] G. Christophe and S. Dirk, ”Evaluating Coverage Based Testing”, M. Broy et al. (Eds.): Model-Based Testing of Reactive Systems, LNCS 3472, 2005, pp. 293-322.

[5] G. Rothermel R. H. Untch C. Chu M. J. Harrold ”Prioritizing Test cases For Regression Testing”. IEEE Transactions on Software Engineering, vol. 27, no. 10, October, 2001, pp 929-948.

[6] Z.Q. Zhou “Using Coverage Information to Guide Test case Selection in Adaptive Random Testing”, 2010. in Proceedings of the 34th Annual IEEE Computer Software and Applications Conference Workshops (COMPSAC 2010). IEEE Computer Society, 2010, pp. 208-212.

[7] Z. Q. Zhou A. Sinaga. and W. Susilo “On the fault-detection capabilities of adaptive random test case prioritization: Case studies with large test suites,” in Proceedings of the 2012 45th Hawaii International Conference on System Sciences (HICCS-45). IEEE Computer Society Press, 2012, pp. 5584–5593.

[8] H. Jaygarl C. K. Chang. and S. Kim “Practical extensions of a randomized testing tool,” in Proceedings of the 33rd Annual International Computer Software and Applications Conference (COMPSAC 2009). IEEE Computer Society Press, 2009, pp. 148-153.

[9] M. Hutchins H. Foster T. Goradia. and T. Ostrand, “Experiments on the effectiveness of dataflow- and controlflow-based test adequacy criteria,” in Proceedings of the 16th International Conference on Software Engineering (ICSE'94). IEEE Computer Society Press, 1994, pp. 191–200.

[10] “Software-artifact Infrastructure Repository”, http://sir.unl.edu, accessed on 3 December 2013.