Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I
IMECS 2009, March 18 - 20, 2009, Hong Kong

An Approach to Test Case Design for Cost Effective Software Testing

Kiran Kumar J [1], A. Ananda Rao [2], M. Gopi Chand [3], and K. Narendar Reddy [4]

Abstract— **Software testing is the critical component of the software development life cycle. Cost of software testing would affect the cost benefit trade-off of a development organization. Any reduction in the cost of software testing would help to deliver the product to the customer at less cost. The existing research is to find the ways to reduce the testing cost. In this paper, an approach to test case design which reduces software testing cost in black box environment has been proposed. This type of approach has not been used earlier.**

The proposed approach reduces the total number of test cases in black box environment. This reduced test case set covers total functionality and ensures the quality of the product. The proposed approach is applied on four case studies and found that the reduction in testing cost is ranging between 27 and 36 percent. Hence, by using the proposed approach the software testing cost can be reduced considerably.

Index Terms— **Approach, software testing cost, test case design, testing cost reduction**

I. INTRODUCTION

Any single software testing technique is not sufficient to test a product completely. Many software testing techniques are required to test complete functionality of a software product. We get large number of test cases by applying various testing techniques. This test case set contain, some test cases to test the functionality of the product, some test cases to test the boundary values, some test cases to test stress, and some test cases to test performance of the product. With this large number of test cases, we can cover the complete functionality of the product.

When a software product is being developed in an organization, multiple software product builds are given to the Quality Assurance (QA) teams to test the product before it is released to the customer. On most of the builds the QA teams need to run complete test case set to ensure that the product is stable.

In this paper we proposed an approach to reduce the total number of test cases in black box environment without affecting the quality of the product. If the total number of test cases can be reduced without affecting the quality of the product, the time required to execute the minimized test case set will be reduced. This reduction in the total number of test cases will reduce the effort required by the QA teams to execute the test cases.

Most of the existing approaches consider test case set which contain, some test cases to test the functionality of the product, some test cases to test the boundary values, some test cases to test stress, and some test cases to test performance of the product. Any reduction in this test case set will reduce the testing time, effort, and cost. Most of the test cases in this set belong to test cases that test the functionality and boundary values of the product. In this paper we have presented an approach, which reduces test cases considering test cases that test functionality and boundary values.

In the proposed approach, it is shown that the two aspects of testing, that is testing for functionality and testing for boundary values can be tested with reduced test cases as these two aspects can be tested simultaneously in most of the situations. It is also shown with examples the situations where these two aspects can be tested simultaneously. In this paper testing simultaneously means, a single test case can cover both the above aspects for a particular situation.

The proposed approach is applied on four real-time case studies and found that the reduction in testing cost is ranging between 27 and 36 percent. This saved lot of time and effort required by the QA teams.

The work in this paper is organized in different sections. The related work is given in section II. Whereas proposed approach to test case design is presented in section III. Case studies and results are presented and discussed in section IV. The conclusions have been given in section V.

II. RELATED WORK

To develop defect free quality software, test cases are very important. Many techniques and approaches [1-7] have been reported in the literature on designing test cases, automated test case generation and test case optimization in recent years.

Pravin M Kamde described how to avoid loses that is inevitable with poor test cases [1]. It will give practical advice on how to improve productivity, usability, scheduling reliability and asset management. Yuri Chernak presented an approach to improve the software testing process based on a new metric and an associated methodology for in-process validation of test case effectiveness [2]. Yizheng Yao and Yingxu Wang presented a new approach to specification-

Manuscript received on December 30, 2008. Sent review report and paper acceptance on January 14, 2009. Paper Number: ICSE_35.

[1] Kiran Kumar J, Software Engineer, IBM India Software Lab, Hyderabad, AP, India. Mobile: +91 94402 22240; email: kirankumarj9@gmail.com.

[2] Prof. Dr. A. Ananda Rao, Vice-Principal, JNTU, Anantapur, AP, India. Mobile: +91 90004 93404; email: akepogu@yahoo.co.in.

[3] M. Gopi Chand, Dept of CSE, GNITS, JNTU, Hyderabad, AP, India. Mobile: +91 98490 42448; email: gopi_merugu@yahoo.com.

[4] K. Narendar Reddy, Dept of CSE, CVR COE., JNTU, Hyderabad, AP, India. Mobile: +91 99631 06396; email: knreddy_mist@yahoo.com.

based test generation that enables test cases to be generated before the implementation of the code [3]. W.T. Tsai developed a method for automated test case generation for programs specified by relational algebra queries [4]. This method can be applied on programs specified by relational algebra queries. Bonoit proposed an approach for automatic test cases optimization in .NET environment at bacteriological level [5]. They also presented a general framework for faults injection. Susan proposed a method to assess the differential risk of failure among a system's modules [6]. Managers can use these failure risk estimates to determine how much testing effort can be economically justified.

Among the papers described above, most techniques are focusing on designing good test cases, automated test case generation and test case optimization for a specific program. In this paper, an approach to test case design that leads to reduction of software testing cost in black box environment is proposed.

III. PROPOSED APPROACH TO TEST CASE DESIGN

Test case is a documentation that specifies inputs, predicted results, and a set of execution conditions for a test item [10]. Designing good test cases is very important to develop a quality software product.

Any single software testing technique is not sufficient to test a product completely. Many software-testing techniques are required to test complete functionality of a software product. We get large number of test cases by applying various testing techniques. This test case set contain, some test cases to test the functionality of the product, some test cases to test the boundary values, some test cases to test stress, and some test cases to test performance of the product. With this large number of test cases, we can cover the complete functionality of the product.

But, to execute this total number of test cases, a lot of time and QA team's effort is required. And, product needs to be tested multiple times, i.e. with multiple builds, multiple releases. If the total number of test cases is reduced without affecting the quality of the product, the time required for the testing, effort required by the QA teams and cost of the testing will be reduced.

In this paper we proposed an approach to reduce the number of test cases, without affecting the coverage of the functionality. This reduction in the number of test cases while covering the same functionality will reduce lot of time required to testing and hence the cost of the product testing is reduced.

The proposed approach is focused to reduce test cases considering test cases that test functionality and boundary values.

The proposed approach contains following four steps.

1. View the two aspects that is functionality and boundary value testing together
2. Identify the situation(s) (considering functionality and boundary values) which can be tested in single test case(s) so as to design minimal test cases
3. Proving logically that the single test case(s) in fact covering both the aspects.
4. Applying above three steps to case studies and validating

IV. CASE STUDIES AND RESULTS

The proposed approach is applied on four real-time ETL tool *(Data ware housing tool)* components: Teradata ETL Database (DB) Component, Informix ETL DB Component, DB2 ETL DB Component and Oracle ETL DB Component. Concepts explained in Fig. 1 and Fig. 2 are generic and applicable to all the above four test case studies.

In Fig. 1, ETL, which stands for "extract, transform and load", is the set of functions combined into one tool or solution that enables companies to "extract" data from numerous databases, applications and systems, "transform" it as appropriate, and "load" it into another databases, a data mart or a data warehouse for analysis, or send it along to another operational system to support a business process.

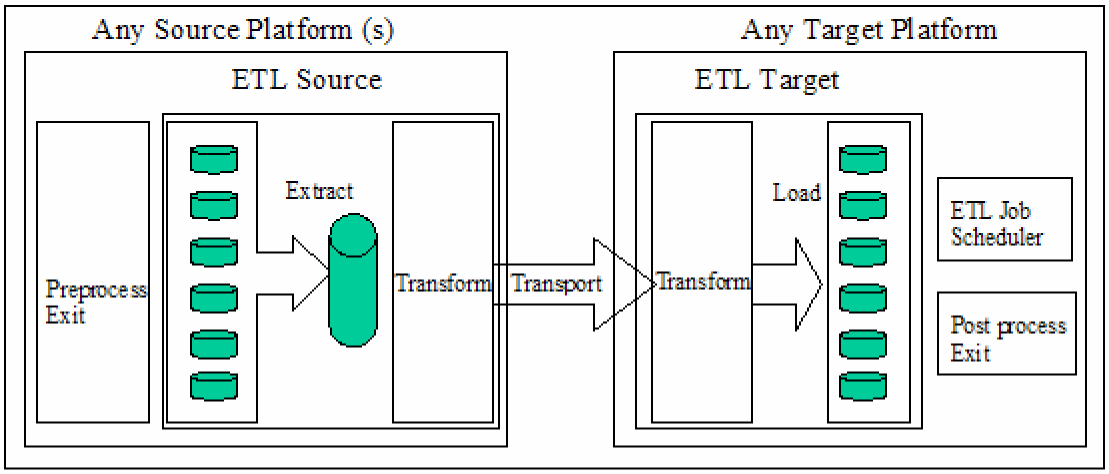


Fig. 1. ETL Process

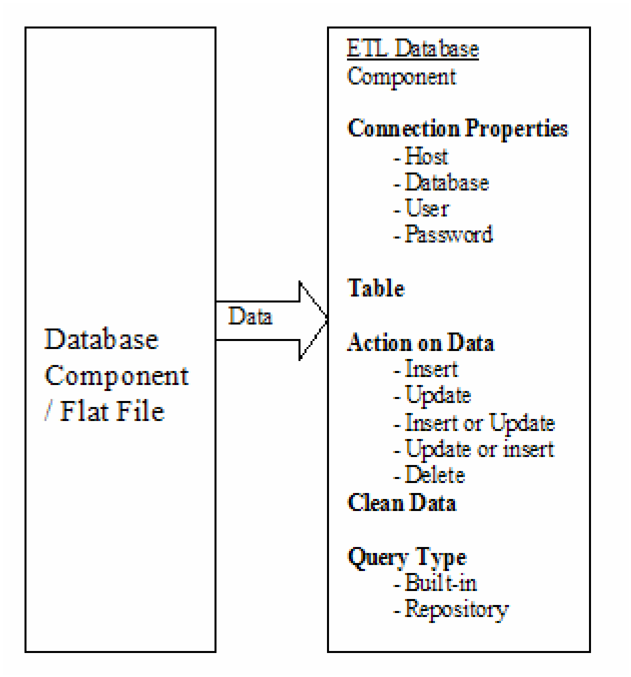


Fig. 2. ETL Database Component write process

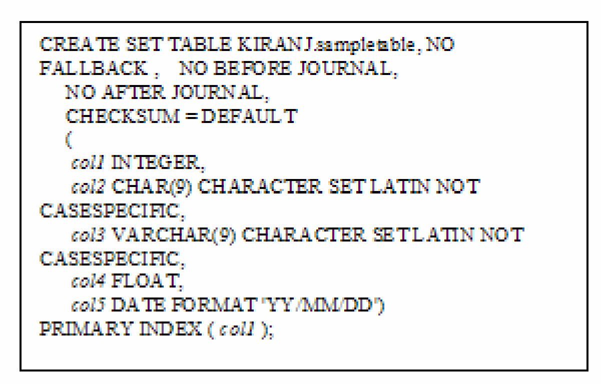
Many test cases are required to test an ETL tool completely. These test cases include: Functional test cases (T_f), Boundary Value test cases (T_b), Stress test cases (T_s), Performance test cases (T_p) and other test cases (T_o) like negative test cases. So the Total Number of test cases (T_n) are: $T_n = T_f + T_b + T_s + T_p + T_o$.

The Fig. 2 shows some of the attributes of a generalized ETL Database Component write process. In this write process, the source could be a ETL DB Component or a flat file and the target is a ETL DB Component. In the write process, the target ETL DB Component reads data from the source component, connects to the respective database using the connection properties specified and writes that data in to the target table.

The test case design using the proposed approach for Teradata ETL DB Component is described in section A.

A. Teradata ETL DB Component Test Case Design

The Fig. 3 shows the metadata of the table '*sampletable*' used in the Teradata ETL DB Component case study. This is a Teradata table that contains 5 columns. The *col1* is integer type, *col2* is character type, *col3* is varchar type, *col4* is float type and *col5* is date type.



```
CREATE SET TABLE KIRANJ.sampletable, NO
FALLBACK ,   NO BEFORE JOURNAL,
   NO AFTER JOURNAL,
   CHECKSUM = DEFAULT
   (
    col1 INTEGER,
    col2 CHAR(9) CHARACTER SET LATIN NOT
CASESPECIFIC,
    col3 VARCHAR(9) CHARACTER SET LATIN NOT
CASESPECIFIC,
    col4 FLOAT,
    col5 DATE FORMAT 'YY/MM/DD')
PRIMARY INDEX ( col1 );
```

Fig. 3. Metadata of the sample table

The Table I shows some sample Functional test cases for the Teradata ETL DB Component write process. Each of these test cases tests a single functionality or scenario of the Teradata ETL DB Component to ensure the particular attribute or function is working properly.

Table I. Functional test cases before applying the proposed approach

Test Case ID	Description	Preconditions	Expected Result	Test Status	Comments
TC$_f$1	Test on writing the data to the target table with Action on data = Insert		The job should add new rows to the target table and stop if duplicate rows are found.		
TC$_f$2	Test on writing the data to the target table with Action on data = Update		The job should make changes to existing rows in the target table with the input data.		
TC$_f$3	Test on writing the data to the target table with Action on data = Insert or Update		The job should add new rows to the target table first and then update existing rows.		
TC$_f$4	Test on writing the data to the target table with Action on data = Update or Insert		The job should update existing rows first and then add new rows to the target table.		
TC$_f$5	Test on writing the data to the target table with Action on data = Delete		The job should remove rows from the target table corresponding to the input data.		

Table II. Boundary Value test cases before applying the proposed approach

Test Case ID	Description	Preconditions	Expected Result	Test Status	Comments
TC_b1	Test on writing the data to col1 with INTEGER data type boundary values		The job should read the INTEGER data type boundary values from input data and write to the target table successfully.		
TC_b2	Test on writing the data to col2 with CHAR data type boundary values		The job should read the CHAR data type boundary values from input data and write to the target table successfully.		
TC_b3	Test on writing the data to col3 with VARCHAR data type boundary values		The job should read the VARCHAR data type boundary values from input data and write to the target table successfully.		
TC_b4	Test on writing the data to col4 with DOUBLE data type boundary values		The job should read the DOUBLE data type boundary values from input data and write to the target table successfully.		
TC_b5	Test on writing the data to col5 with DATE data type boundary values		The job should read the DATE data type boundary values from input data and write to the target table successfully.		

The Table II shows some sample Boundary Value test cases for the Teradata ETL DB Component write process. Each of these test cases tests a single column or data type to ensure the boundary values of that data type are written properly to the target table.

The test case design for Teradata ETL DB Component using the proposed approach is described in the following four sections (A.1 – A.4).

A.1. View the two aspects together (Step 1)

Many software-testing techniques are required to test complete functionality of a software product. We get large number of test cases by applying various testing techniques. These test cases include: functional test cases (T_f), Boundary Value test cases (T_b) , Stress test cases (T_s), Performance test cases (T_p) and other test cases (T_o) like negative test cases. $T_n = T_f + T_b + T_s + T_p + T_o$.

With this large number of test cases, we can cover the complete functionality of the product. But, to execute this total number of test cases, a lot of time and QA team's effort is required. And, product needs to be tested multiple times, i.e. with multiple builds, multiple releases. If the total number of test cases is reduced without affecting the quality of the product, the time required for the testing, effort required by the QA teams and cost of the testing will be reduced.

The proposed approach is focused to reduce test cases considering test cases that test functionality and boundary values. Most of the test cases in this set belong to test cases that test the functionality and boundary values of the product. In this paper an approach is presented, which reduces test cases considering test cases that test functionality and boundary values.

A.2. Identifying the situations that can be tested in a single test case and designing minimized test case set (Step 2)

The test case TC_f1 tests the functionality of the Teradata ETL DB Component when the attribute 'Action on Data' is set to 'Insert' and the test case TC_b1 tests the INTEGER data type boundary value that is written to the target Teradata table. Both of these test cases TC_f1 and TC_b1 are testing the two aspects i.e. functionality and boundary values of the Teradata ETL DB Component.

By using the proposed approach these two test cases could be viewed together and tested in a single test case. For example, the test cases TC_f1 and TC_b1 are viewed together and designed a single test case TC_m1 (Table III) that covers the both aspects. The minimized test case set designed using the proposed approach is shown in the Table III.

A.3. Providing logically that the single test case in fact covers both the aspects (Step 3)

Each test case in the minimized test case set described in Table III will test the functionality of the Teradata ETL DB Component to ensure that the particular attribute is working properly *and* also tests the boundary values for various columns in the target table to ensure that the boundary values of that column data type are written properly. For example, the TC_m1 in the minimized test case set tests whether the Teradata ETL DB Component is working properly when the attribute 'Action on Data' is set to 'Insert' and also tests whether the INTEGER data type boundary value is written to the target table properly which were tested by the test cases TC_f1 and TC_b1.

Table III. The minimized test case set designed using the proposed approach

Test Case ID	Description	Pre-conditions	Expected Result	Test Status	Comments
TC***m***1	Test on writing the data to the target table with Action on data = Insert and col1 contains INTEGER data type boundary values		The job should read the input data, add new rows to the target table successfully and stop if duplicate rows are found.		
TC***m***2	Test on writing the data to the target table with Action on data = Update and col2 contains CHAR data type boundary values		The job should read the input data and make changes to existing rows in the target table with the input data		
TC***m***3	Test on writing the data to the target table with Action on data = Insert or Update and col3 contains VARCHAR data type boundary values		The job should read the input data, add new rows to the target table first and then update existing rows.		
TC***m***4	Test on writing the data to the target table with Action on data = Update or Insert and col4 contains DOUBLE data type boundary values		The job should read the input data, update existing rows first and then add new rows to the target table		
TC***m***5	Test on writing the data to the target table with Action on data = Delete and col5 contains DATE data type boundary values		The job should read the input data and remove rows from the target table corresponding to the input data		

In similar way, the remaining test cases in the minimized test case set {TC***m***1 – TC***m***5} described in Table III will test the both aspects, functionality and the boundary values of Teradata ETL DB Component which have been tested by the test cases {TC***f***1-TC***f***5 and TC***b***1-TC***b***5}.

A.4. Applying the above three steps to case studies and validating (step 4)

If the number of boundary value test cases that are viewed together with functional test cases, the number of test cases test cases reduced is Tbr. Then, after applying the proposed approach the total number of test cases is minimized to:

$T_{min} = T_n - Tbr$

And,

The percentage of test case reduction ($t_{reduction}$) is:

$t_{reduction} = (T_{br}/T_n) * 100$

In similar way, the proposed approach is also applied on Informix ETL DB Component, Oracle ETL DB Component and DB2 ETL DB Component. The Table IV describes the Total number of test cases (T_n) before applying the proposed approach, the total number of test cases in the minimized test case set (T_{min}) after applying the proposed approach and the percentage of test case reduction ($t_{reduction}$).

After applying the proposed approach, the total number of test cases for Teradata ETL DB Component is reduced by 27 %, Informix ETL DB Component are reduced by 30 %,

Table IV. The minimized test cases

Module	T_n	T_{min}	$t_{reduction}$
Teradata ETL DB Component	1543	1126	27 %
Informix ETL DB Component	1439	1008	30 %
DB2 ETL DB Component	1296	934	29 %
Oracle ETL DB Component	1559	997	36 %

DB2 ETL DB Component test cases are reduced by 29 % and Oracle ETL DB Component test cases are reduced by 36 %. The proposed approach is applied to design test cases for four real-time ETL Components and found that the number of test cases reduction is ranging between 27 to 36 percent (Table IV). Hence the propped approach validated.

Graphical representation of the results for various ETL DB Components is shown in Fig. 4. The X-axis denotes various ETL DB Components and Y-axis denotes the total number of test cases for each component.

By applying the proposed approach, $t_{reduction}$ percent test cases are reduced for a ETL DB Component. So the time required to execute these test cases of ETL DB Component is also reduced by $t_{reduction}$ percentage. Hence, the cost of the software testing will be reduced by $t_{reduction}$ percentage.

The effort required to apply this approach is a one-time effort, but it will reduce the effort and time required for all the remaining testing cycles of the product.

These case studies show that, the proposed approach saves a substantial amount of testing time and effort.

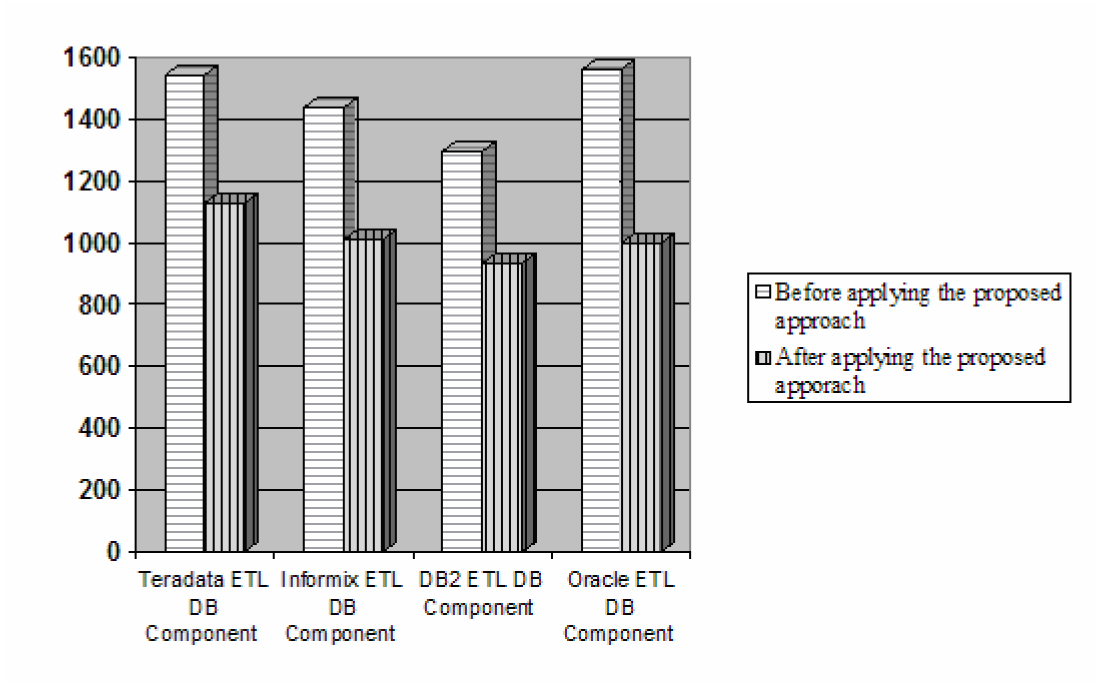


Fig. 4. The Total number of test cases before and after applying the proposed approach

V. CONCLUSIONS AND FUTURE RESEARCH

In this paper an approach is proposed to reduce the number of test cases in black box environment, without affecting the coverage of the functionality. This reduction in the number of test cases while covering the same functionality will reduce lot of time required for testing, effort required by the QA teams, and the cost of product testing.

The proposed approach is applied on four real-time ETL Tool (Data ware housing tool) Components that are used by many people all over the world. The tested ETL tool components are Teradata ETL DB Component, Informix ETL DB Component, Oracle ETL DB Component and DB2 Component. The detailed analysis results are given for Teradata ETL DB Component, where for the other three case studies final results are given. It is known from the case studies that the number of test cases can be reduced by applying the proposed method and the reduction in testing cost is ranging between 27 and 36 percent. Hence, by using the proposed approach the software testing cost can be reduced considerably.

The effort required to apply this approach is a one-time effort, but it will reduce the effort and time required for all the remaining testing cycles of the product.

As part of future research, the reduction in regression testing cost using the proposed method has to be estimated.

VI. REFERENCES

[1] Pravin M. Kamde, V. D. Nandavadekar, R. G. Pawar, "Value of Test Cases in Software Testing", *International Conference on Management of Innovation and Technology,* IEEE, 2006.

[2] Yuri Chernak, "Validating and Improving Test-Case Effectiveness", January / February 2001, IEEE Software.

[3] Yizheng Yao and Yingxu Wang, "A New Approach to Test Case Generation based on Real-Time Process Algebra (RTPA)", *CCECE 2004*, Niagara Falls, IEEE, 2004

[4] W.T. Tsai, Dmitry Volovik and Thomas F. Keefe, "Automated Test Case Generation for Programs Specified by Relational Algebra Queries", IEEE Transactions on Software Engineering, Vol. 16, No. 3, March 1990.

[5] Benoit Baudry, Franck Fleurey, Jean-Marc Jézéquel and Yves Le Traon, "Genes and Bacteria for Automatic Test Cases Optimization in the .NET Environment", *Proceedings of the 13 th International Symposium on Software Reliability Engineering* (ISSRE'02, IEEE, 2002

[6] Susan A. Sherer, "A Cost-Effective Approach to Testing", IEEE Software, March 1991.

[7] Xiaoyuan Xie, Baowen Xu, Liang Shi, Changhai Nie and Yanxiang He, "A Dynamic Optimization Strategy for Evolutionary Testing", *Proceedings of the 12th Asia-Pacific Software Engineering Conference* (APSEC'05), IEEE 2005.

[8] Tsuneo Yamaura, Hitachi Software Engineering, "How to Design Practical Test Cases", IEEE Software November/ December 1998

[9] Tafline Murnane, Karl Reed and Richard Hall, "Tailoring of Black-Box Testing Methods", *Proceedings of the 2006 Australian Software Engineering Conference* (ASWEC'06), 1530-0803/06 $20.00 © 2006 IEEE

[10] IEEE Standard for Software Verification and Validation, IEEE Computer Society.

[11] Section 2 "Definition" of ANSI/ IEEE Standard 829, 1983.

[12] Section 3.2.4 "Features to be tested" of ANSI/ IEEE Standard 829, 1983.

[13] Section 3.2.6 "Attributes to be tested" of ANSI/ IEEE Standard 829,1983.

[14] William Perry (1995); "Effective methods for software testing", John Wiley, New York.

[15] Elaine J. and F. I. Vocolos; "Experience with performance testing of software systems: Issues, approach and case study", IEEE transaction son software engineering, Vol-26, No – 12 December 2000 PP 1147-1156.
[16] Moller and Paulish; "Software matrics: A practitioner guide to improve product development", Champnan and Hall, second edition.
[17] Roger S. Pressman; "Software engineering: A practitioner approach", Mc-Graw Hill International edition, computer science series.
[18] E. Dustin; "Effective software testing", low-price publication first edition 2003.