International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-1, June 2013

Regression Test Case Selection for Testing Database Applications

Vandana Sharma, Arun Prakash Agrawal

Abstract*— *Regression testing is a part of software maintenance and it consumes about two-third of the overall software life cycle cost. It is the process of executing the full or partial test cases from the original test suite after any modifications to the original program. It tests both the modified code and other parts of the program that may be adversely affected by changes introduced in the program or a part of it. It is an expensive activity that is done whenever there are some changes in software. Regression testing tests both the modified code and other parts of the program that may be adversely affected by changes introduced in the program or a part of it. Test case selection selects the test cases to test the modified as well as unmodified part of the program from the original test suite. The regression testing of database applications concerns with the state of the database as it contributes too many components that increase the complexity of the applications because in case of database the test cases are not independent of each other and the database requires to be reset every time. The database applications are frequently modified due to the need of different requirements like, increase in number of users, components and data. Therefore regression testing of database applications is an essential activity as it requires maintaining the state of the database. It may be conducted either manually by re-executing a subset of all test cases of the original test suite or using automated tools. These tools enable the software testers to capture test cases and results for subsequent playback and comparison.

In this paper, we have shown a study of the time taken in resets made to a database that is done manually or automatically with the help of various tools. We have also proposed the way in which the reset time of database state is reduced to a large extent. The database always requires to be reset after executing every query that too is done manually by the tester or with the help of some automated tool. In our work after reducing the reset time of database state we have presented the test cases with the details of the time taken in execution and code coverage of database application. Then the resulted test cases are selected from the original test cases that achieves the selection of maximum number of fault revealing test cases.

Index Terms*—*Database Applications, Database Testing, Regression Testing, Regression Test case Selection, Software Testing.

I. INTRODUCTION

Software Testing is the method of executing a program with the intent of finding failures in the program. This is done after the development phase is complete. When a program is implemented to provide a concrete representation of an algorithm, the developers of this program are concerned with the correctness and performance of the implementation [9].

Manuscript published on 30 June 2013.
***Correspondence Author(s)**
Vandana Sharma, ASET, Amity University, Sector-125, Noida, INDIA.
Prof. Arun Prakash Agrawal, Assistant Professor, ASET, Amity University, Sector-125, Noida, INDIA.

Software engineers must confirm that their systems reach an appropriate level of quality. Hence, software testing is a procedure of evaluating a software item to detect differences between given input and desired output and to ensure the correctness and quality of the software. It is the procedure of evaluating a system with the intention of finding errors and as early as possible. Software testing can be appropriately used in conjunction with correctness proofs and other types of formal approaches in order to develop high quality software systems [Bowen and Hinchley, 1995, Geller, 1978]. A main purpose of software testing is to identify software failures so that faults may be identified and corrected.

Regression testing is a testing process which is applied after a program when it is modified and some changes are made to the original program. It involves testing the modified program with some test cases in order to re-establish our Confidence that the program will perform according to the modified specification. It is a full or partial selection of already executed test cases of the original program which are re-executed to make sure that the existing functionalities will work fine. This type of testing is done to make sure that new code changes must not have any side effects on the presented functionalities of the program and also ensures that old code will still works once the new code changes are finished [2]. Regression test selection algorithms aspire to increase the efficiency by considering only on those test cases that can perhaps be affected by the changes that have been made to the program or system by analyzing the source code of the program before and after the changes that have been introduced, and then selecting those test cases according to some criteria which needs to be re-executed. A regression test means that whenever a new function is added or modifications are done in the software, all previous validated test cases are run, and the results are compared with the standard results previously stored on file. Major objectives of regression testing are firstly retest changed components and secondly check the affected parts. This testing is done to make sure that new code changes should not have any side effects on the existing functionalities. It ensures that old code still works once the new code changes are done. Figure 2.1 below shows the complete process of regression testing [5]. The test cases are classified as: reusable, re-testable, obsolete, new-structural and new-specification test cases [3].

A large number of database applications are currently in use. These applications are usually composed of several components contributing to an increase in their sophistication. Database applications also need to be regularly modified due to various different requirements and its state also requires to be maintained. The requirements and challenges in RTS of data-driven applications are different from the module of programs that we have already discussed so far.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

A. Regression testing components

1) **Test Case -** In software testing, a test or a test case is a set of variables and conditions through which a tester will find that whether a requirement of a software system or an application is functioning correctly or not. A test case is also referred as a series of steps which determines the correct functionality of the software system. It determines if a requirement upon an application is partially or fully satisfied. In order to completely test that all the specified requirements of the system are met, there need to be at least one test case for every requirement unless there is a sub requirement of the requirement. In this case, there has to be at least one requirement for every sub requirement. The process of testing relies on the concept of test case, which is defined as a detailed specification of testing, including some important information about the test, like pre-conditions, post-conditions, data for entry, interdependency, etc. (Bastos, 2007).
2) **Test oracle -** t is a mechanism which helps in determining that whether the application or software system has passed or failed the test. In some conditions, test oracle might be a use case or a requirement, while in other case it could be a heuristic. It might take a set of many test cases to conclude that a system or a software program is sufficiently analyzed to be released. Test cases are generally referred as **test scripts,** especially when written. The set of test cases is referred as **test suite**. Hence the combination of test cases may form a test suite. Test case definitions include the following information (as per IEEE 610)
   - Pre-conditions
   - Set of input values
   - Set of expected results
   - How to execute the test and check results
   - Expected post conditions
3) **Execution Time -** The total amount of time a test case consumes in executing a program is known as its execution time. Lower is the value of execution time, better the test case is. As time is an essential feature which is responsible in determining the fitness, execution time of each test case is used for finding their fitness function. The run time of the test cases can be recorded by using the testing tools as very limited time is available for software testing and especially for regression testing.
4) **Test Coverage -** The metrics used to measure testing thoroughness include statement testing (whether each statement in the program has been executed at least once), branch testing (whether each exit from each branch has been executed at least once) and path testing (whether all logical paths, which may involve repeated execution of various segments, have been executed at least once) [4]. Statement testing is most regularly used as it is comparatively easier to implement. Two forms of test adequacy criteria are also considered: a) method coverage: a method is covered when it has been entered, b) block coverage: a basic block is a sequence of statements or instructions without any jump targets is covered whenever it is entered because several high level languages source statements can be in the same basic block, it is sensible to keep track of basic blocks rather than individual statements and time of execution [4].

B. Regression Testing Techniques [6]

1) Retest all- This is the process of regression testing in which all the tests in the presented test suite are re-executed. This is very expensive as it requires huge time and resources. It is 100% fault detecting technique and also there is no size reduction in the test suite.
2) Regression Test Selection (RTS) – RTS selects specific test cases from the existing test suite instead of running the entire test suite again. The strategy of RTS is to minimize the test suite and maximize fault detection ability.
3) Prioritization of Test Cases - It is ordering of test cases for testing depending on business impact, critical & frequently used functionalities. This type of ideal ordering of test cases will greatly reduce the test suite of regression.

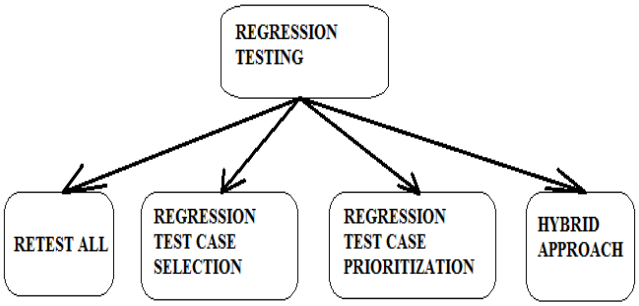

Fig: 1 Types of Regression Testing Techniques

II. REGRESSION TEST CASE SELECTION

The RTS techniques select a subset of valid test cases from original test suite {T} to test that the modified part of the program does not affect the unmodified parts of a program and hence the program must continue to work correctly. It affects the cost-effectiveness of regression testing and involves two major activities:

1) Identification of the modified parts of the program.
2) Selecting the subset of test cases that has high probability of detecting errors.

A. Outline for Regression Test Selection (RTS) Techniques [4].

Some of the matrices and characteristics on the basis of which RTS techniques can be categorized as: inclusiveness, precision, efficiency, and generality. **Inclusiveness -** It measures the extent to which the RTS technique chooses modification revealing tests from previous test suite {T} for inclusion in modified test suite {T`}. **Precision -** It measures the extent to which the RTS technique omits test cases that does not reveal modifications. **Efficiency -** The efficiency of RTS technique measures the space and time efficiency which depends on the computational cost and the size of the test suite selected by the technique. **Generality -** The generality of RTS technique is its ability to perform in different environment and situations. The technique needs to be practical, able to handle realistic modifications and does not depend on some specific tools.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

II. TESTING THE APPLICATIONS INVOLVING DATABASE

A database application means a software system that uses an RDBMS (Relational Database Management System) as its primary persistence mechanism. A database–driven application is achieved by the manipulation of both the database state and the program state. The application programs which does not involves database are called as stateless applications and those applications which involves database are known as state-full applications. Fig 2 which shows the steps involved in testing the database applications using DBunit. The regression testing of database applications concerns with the state of the database as it contributes too many components that increase the complexity of the applications because in case of database the test cases are not independent of each other and the database requires to be reset all the time [9].

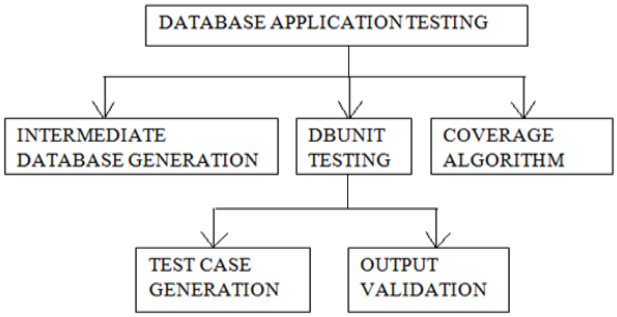

Fig: 2 Database Application Testing

A. Database testing involves:

Testing process consists of writing tests which is a 3-step process of set up the tests, run the test and check the results with the expected results [9]. Setting up the database involves two common strategies i.e., rebuilding the database and data re-initialization. Applications involving database also consists of black-box testing and white-box testing.

1) Black-Box Testing – This testing is done at the interface which includes O/R mappings including the data information, Incoming data values, Outgoing data values as of queries, stored functions, views, etc…
2) White-Box Testing – This testing is done internally within the database and includes code, unit tests, view definitions, referential rules, existence tests for database schema elements, etc…

Database aware regression testing can be achieved by considering both the program and database state which is not satisfied by any of the technique exists. It involves interaction of an application with database schema in a session by an application program using SQL statements. The complete SQL command is constructed at run-time during the interaction session. SQL statements are created in order to modify or view the state of relational database. The regression testing tools is developed with modern integrated development environment especially designed for regression testing of database applications [10]. Database applications characteristics such as Query Language programming, integrity constraints, exception handling and triggers creates problem for maintenance activities especially for regression testing which is a part of maintenance activity of database applications. As a solution two-phase regression testing methodology is proposed in which the control and data flow analysis of database applications are explored as well as two algorithms – Graph-walk and Call Graph Firewall algorithms are proposed for reducing the number of test cases of regression tests [9].

1) Test Suite Prioritization - The prioritization technique specifies which test case will be addressed first from the original test cases [9]. It does not discard any test case and the efficiency of the regression testing depends on the criteria of the prioritization. It also increases the rate of fault detection and code coverage.
2) Test Suite Reduction - The reduction technique aims to find the subset or smaller set of test cases that covers the similar requirements as fulfilled by the original test suite [9]. It reduces the cost of regression testing and size of test suite.

III. DESIGN AND METHODOLOGY

The idea of test selection of database applications is guaranteed to be safe only in terms of the manipulation that the database state is introduced. That allows more efficient regression test selection in cases where the management of program state is secondary to the management of the external persistent state.

In this work we have implemented an application that shows both the reset that made to the database state manually and reset that made to the database state automatically with the help of different tools. We have showed the way in which the reset time of database state is reduced to a large extent. The database always required to be reset after executing every query that too is done manually by the tester or with the help of some automated tool.

We have used the DBunit tool for intermediate database generation which is to be used in for of xml file for generating the database as shown in Fig 3 below and Eclipse IDE as integrated development environment for developing the whole scenario.

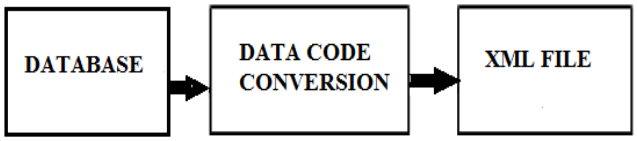

Fig: 3 Intermediate Code Generation

After generating the database we had designed an application which interacts with the database. We had used a real dataset of the employees of Sisoft pvt ltd for implementing and regression testing our database in a real scenario. That data is highly confidential so we'll not be able to provide with the details of the data, the only thing we can show is the test cases generated by using that data in my application.

The application I have created runs more than one query in one go. This can be achieved by selecting the right order of test cases from the given test cases that are created only after resetting the database which reduces the time of resetting the number of time the query is being executed. The whole scenario is created in Eclipse IDE (indigo). The Fig 4 below shows the execution time and code coverage of the application with failure and passed test cases.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

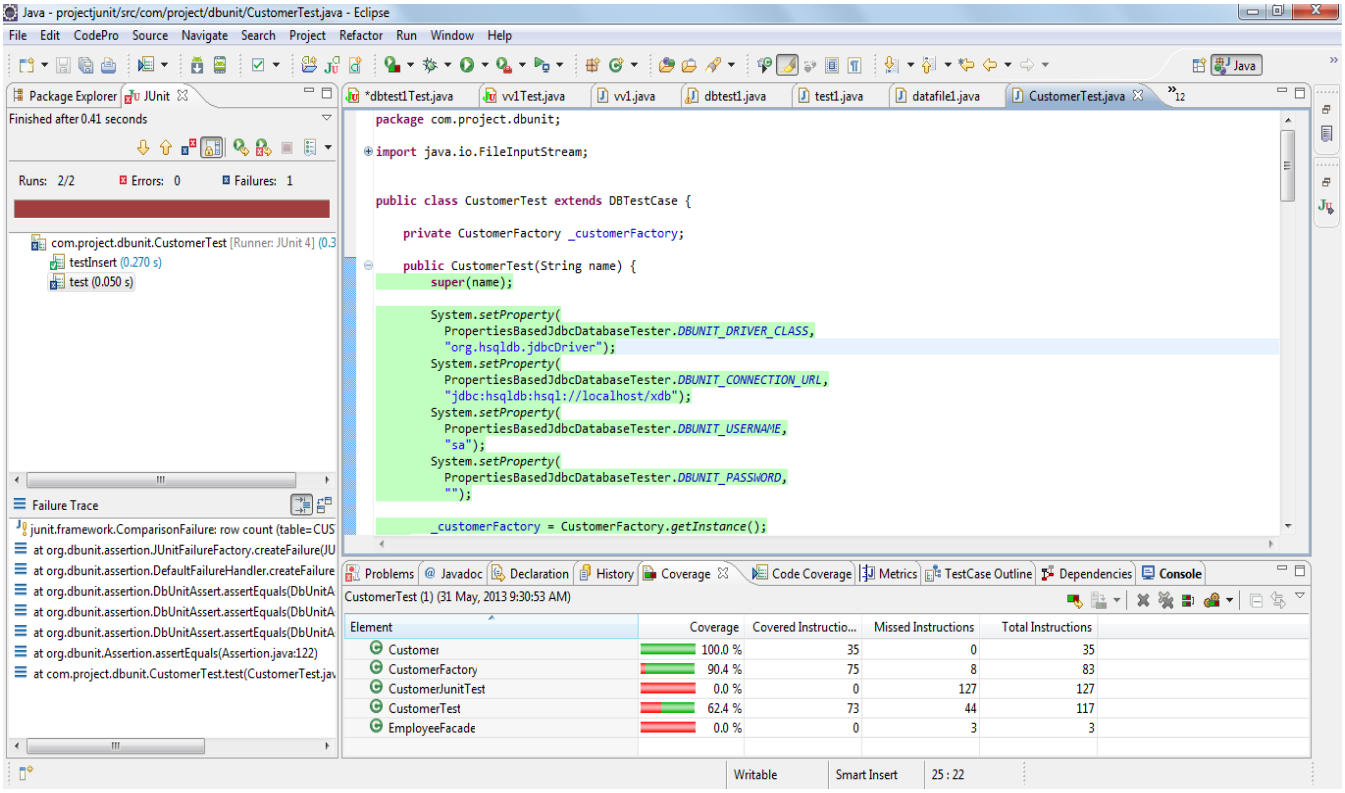

Figure: 4 Execution of the test case of database application with execution time and code-coverage view

After reducing the reset time of database state we have generated different test cases with the details of their time taken in execution and code coverage. The original test suite is shown in table 1 which consists of 20 test cases. Table 2 shows the fault revealing test cases from the original test suite and table 3 shows the test cases after changing the order (test cases are taken from the test cases shown in table2 for changing the order)

Table: 1 Original Test Suite

Test cases	Output	Time taken in Execution	Code coverage
T1	P(Pass)	0.312	100%
T2	F(Fail)	0.281	75%
T3	F(Fail)	0.312	75%
T4	F(Fail)	0.312	75%
T5	F(Fail)	0.297	75%
T6	F(Fail)	0.297	75%
T7	P(Pass)	0.296	100%
T8	P(Pass)	0.312	80%
T9	P(Pass)	0.312	80%
T10	F(Fail)	0.296	75%
T11	F(Fail)	0.296	75%
T12	F(Fail)	0.312	60%
T13	F(Fail)	0.296	42.90%
T14	F(Fail)	0.296	42.90%
T15	P(Pass)	0.281	80%
T16	F(Fail)	0.297	42.90%
T17	F(Fail)	0.016	50%
T18	F(Fail)	0.016	33.30%
T19	P(Pass)	0.312	100%
T20	F(Fail)	0.016	50%

Table: 2 Test Suite after selecting fault revealing test cases

Test cases	Output	Time taken in Execution	Code coverage
T2	F(Fail)	0.281	75%
T3	F(Fail)	0.312	75%
T4	F(Fail)	0.312	75%
T5	F(Fail)	0.297	75%
T6	F(Fail)	0.297	75%
T10	F(Fail)	0.296	75%
T11	F(Fail)	0.296	75%
T12	F(Fail)	0.312	60%
T13	F(Fail)	0.296	42.90%
T14	F(Fail)	0.296	42.90%
T16	F(Fail)	0.297	75%
T17	F(Fail)	0.016	50%
T18	F(Fail)	0.016	33.30%
T20	F(Fail)	0.016	50%

Table: 3 Test Suite after changing the order of the test cases

Test cases	Output	Time taken in Execution	Code coverage
T17	F(Fail)	0.016	50%
T18	F(Fail)	0.016	33%
T20	F(Fail)	0.016	50%
T2	F(Fail)	0.281	75%
T10	F(Fail)	0.296	75%
T11	F(Fail)	0.296	75%
T13	F(Fail)	0.296	43%

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

T14	F(Fail)	0.296	43%
T5	F(Fail)	0.297	75%
T6	F(Fail)	0.297	75%
T16	F(Fail)	0.297	75%
T3	F(Fail)	0.312	75%
T4	F(Fail)	0.312	75%
T12	F(Fail)	0.312	60%

Test Case Selection is performed on the bases of maximum code coverage and 1 minute of time duration taken for executing the combination of test cases in order to achieve the maximum number of faults and minimum number of resets in a database of given application program (inserting, deleting and reading the data from the database simultaneously without getting any incorrect results).
Table 4 shows the test suite consists of test cases {T2, T10, T11, T17, T18, T20} that will achieves the target of minimum database reset by taking only 0.921 seconds in the code execution of the application with maximum code coverage of the program and hence achieves the selection of maximum number of fault revealing test cases.

Table: 4 Selected Test Cases that will achieve the target

Test cases	**Output**	**Time taken in Execution**	**Code coverage**
T17	F(Fail)	0.016	50%
T20	F(Fail)	0.016	50%
T18	F(Fail)	0.016	33.30%
T2	F(Fail)	0.281	75%
T10	F(Fail)	0.296	75%
T11	F(Fail)	0.296	75%

IV. CONCLUSION

Regression testing is an important software maintenance activity to guarantee the integrity of software after modifications that were made to the program. Most of the methods and tools that exist for software testing does not work well with database applications. These tools will only work well if applications are stateless or test cases can be designed in such a way that they do not alter the database state.
To execute tests for testing or regression testing database applications effectively and efficiently, the challenge is to control the database state of the database during testing and to change the order of the test runs in such a way that expensive database reset operations that carry the database into the accurate state need to be executed as infrequently as possible.

V.FUTURE WORK

In the current scenario there is very less work in maintaining the state of database applications. In my work I also have worked on the same limitation and this work can be extended in following ways:

1) To implement a safe, efficient algorithm of regression test selection.
2) Implementing the database safety and combined safety algorithms for a safe regression testing of database applications.

REFERENCES

1. G. M. Kapfhammer, “Software Testing”, Department of Computer Science Allegheny College.
2. H. Leung, and L. White, “Insights into regression testing,” In Proceedings of the Conference on Software Maintenance IEEE CH2744-1/89/0000/0060.
3. S. Yoo, M. Harman, “Regression Testing Minimisation, Selection and Prioritization: A Survey,” King’s College London, Centre for Research on Evolution, Search & Testing, Strand, London, WC2R 2LS, UK
4. S. Nachiyappan, A. Vimaladevi and C. B. SelvaLakshmi, “An Evolutionary Algorithm for Regression Test Suite Reduction”, International Conference on Communication and Computational Intelligence, Dec-2010, p. 503-508.
5. G. Rothermel, M. J. Harrold, “Analyzing Regression Test Selection Techniques,” IEEE Transactions on Software Engineering, VOL. 22, NO. 8, AUGUST 1996
6. X. Lin, “Regression Testing in Research And Practice,” Computer Science and Engineering Department University of Nebraska, Lincoln 1-402-472-4058.
7. G. M. Kapfhammer, “Regression Testing,” Department of Computer Science Allegheny College.
8. X. Lin, “Regression Testing in Research And Practice,” Computer Science and Engineering Department University of Nebraska, Lincoln 1-402-472-4058.
9. D. Kossmann, C. Binnig, E. Lo, “A Framework for Testing DBMS Features,” The VLDB Journal, Springer-Verlag 2009. DOI 10.1007/ s00778-009-0157.
10. Regression Testing Tools and Methods [online], Available at: http://www.softwaretestinghelp.com/regression-testing-tools-and-methods/ (Accessed: 4th May 2013).

AUTHOR PROFILE

Vandana Sharma has obtained her B.Tech in Information Technology from Maharishi Dayanand University, Rohtak. Presently she is persuing her M.Tech in Computer Science from Amity University, Noida. Her area of interest includes software testing, database testing, mobile applications and computer networks.

Prof. Arun Prakash Agrawal has ten years of experience of teaching at graduate and post-graduate levels. He obtained his M.Tech. in Computer Science and Engineering from Guru Gobind Singh Indraprastha University, Delhi. He is also the gold medalist of his batch. He has taught at various engineering colleges of repute. Presently he is pursuing his Ph.D. in the area of software engineering from Guru Gobind Singh Indraprastha University, Delhi. He has several research papers of national and international repute to his credit. He He is a member of IEEE computer society, IAENG and ACM. His area of interest includes software engineering, software testing, computer networks and mobile computing.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)