Industry Based Regression Testing Using IIGRTCP Algorithm and RFT Tool

K HemaShankari, R. ThirumalaiSelvi, N. V. Balasubramanian

***Abstract*- Software maintenance is an expensive activity in Software Industry, estimated at nearly 60% of total cost. Regression testing is an integral part of this activity. Any modification to software is verified through Regression Testing. Both the research community and the software industry have paid substantial attention to it. This paper is a survey of the current practice, and the commonality as well as gaps between the two communities. The goal of our research is to improve control of regression testing and reduce redundant testing through proper selection strategies. The proposed Improvised Industry oriented Genetic algorithm for Regression Test case Prioritization (IIGRTP) is compared with previous approach of using APFD metric combined with the use of Rational Functional Tester (RFT), a Java Tool developed by IBM to automate the generation of test cases. RFT not only enables regression test case generation but also integrates with Rational Test Manager (RTM). We demonstrate the improved performance of our approach using a case study.**

***Index Terms*—Test case prioritization, Regression testing, Business rules, APFD metric, RFT tool**

I. INTRODUCTION

Regression Testing is an integral part of any software development methodology. With extreme programming methodology, design documents are often replaced by extensive, repeatable, and automated testing of entire software package at every stage in the software development life cycle.

K HemaShankari is Research Scholar, Bhartath University, and Assistant Professor , Department of Computer Science, Women's Christian College Chennai, India (email: hems_banu@yahoo.com)

R. ThirumalaiSelvi is Research Supervisor, Assistant Professor, Govt.Arts College (Men), Nandhanam, India (email: sarasselvi@gmail.com)

N. V. Balasubramanian was formerly, founding Head of Computer Science Department, and Professor, City University of Hong Kong, and Professor Emeritus, R.M.K. Engineering College, Chennai, India (email:laks.balu@gmail.com)

Thus Regression Testing is not an isolated one-off feature, but a full fledged activity varying in scope and preconditions, and highly context dependent. Several techniques have been proposed and evaluated empirically; but in many cases, they are context specific and do not lend themselves to general use. This research discusses the limitations of current approaches on regression testing, and proposes a practical technique which combines change-impact-analysis, business-rules-model, cost-risk-assessment, and test-case-management. It provides confidence in modified software. The later sections of this paper elaborate how regression test cases are prioritised based on factors such as rate of fault detection, percentage of faults detected, and application of RFT Tool.

II.ISSUES FOR REGRESSION TESTING IN INDUSTRY APPLICATION

A. Issues

There are typically two major problems for regression testing of large-scale business systems. Firstly, regression test coverage cannot be accurately defined with the changes of system; Secondly, the number of test cases expands dramatically with the combination of parameters, so it is unable to complete regression testing of the minimum coverage requirements within the determined period of time at a reasonable cost.

Automated functional testing tools are frequently introduced in the testing of large business systems. These tools provide a basic means of testing, but t automatic function test management framework is not available, which leads to the fact that automated functional tests are often unable to be effectively implemented and carried out. The root cause is that functional testing is based on business, with a strong industry relevance, but automated functional testing tools are not related to business, so it cannot automatically adapt to the specific business needs of each industry, and it requires a lot of human intervention during the implementation of the testing process, and the results are often difficult to meet people's expectations.

Regression testing of large-scale business systems tends to be restrained by the deadline and budget constraints, and engineering properties of the test determine that it is impossible to achieve completely as it describe in theory. With the limited time and resources, in order to make more rational

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

arrangements for testing, a decision-making mechanism is of great need in testing planning phase to constraints resources (time, manpower, budget) based on the premise of risk assessment and (test) cost estimation for decision making.

B. Methodology

The previously mentioned test models are relying on software development process, so there is no practical implementation approach for regression testing. Different from the unit testing, integration testing and performance testing in development process, regression testing repeatedly emphasizes accumulation, which can be completed through the structure and the business rules modeling methods, so that the cycle of regression testing can proceed.

To build a supporting platform of regression testing for decision-making, at first, you need to scan and analyze the source code of the core business systems, and set up an application description model; meanwhile, a bank of expert knowledge of the industry should be established to collect and refine business information. And then, a model of business rules should be established to express business information. Finally, risk assessment model will be established, according to industry application and the characteristics of test implementation. If business systems change with the modification of demands, and with the changes of system maintenance and other reasons; if new versions of the software are produced by the development department, implementation steps regression testing of are as follows:

(1) Scan and analyze the source codes in the new version, and conduct analysis of changes bases on the application model, automatic identify system changes;

(2) Analysis of change impacts analysis accurately pointed out the scopes of functional business directly or indirectly influenced by a change of version.

(3) With the application of business rules, the regression test ranges are determined by experts and analysts

(4) Test suite is generated in the assessment model of cost and risk, and it will be compressed with optimization algorithm;

(5) Complete automatic testing by refusing used test cases in the library or developing new cases.

C. Limitations of the APFD Metric

The APFD metric just presented relies on two assumptions: (1) all faults have equal severity, and (2) all test cases have equal costs. In practice, however, there are cases in which these ssumptions do not hold: cases in which faults vary in severity and test cases vary in cost. In such cases, the APFD metric can provide unsatisfactory results.

(i) Average Percentage Block Coverage (ABC).

This measures the rate at which a prioritized test suite covers the blocks.

(ii) Average Percentage Decision Coverage (ADC).
This measures the rate at which a prioritized test suite covers the decisions (branches).

(iii) Average Percentage Statement Coverage (ASC).

This measures the rate at which a prioritized test suite covers the statements.

(iv) Average Percentage Loop Coverage (ALC).

This measures the rate at which a prioritized test suite covers the loops.
(v) Average Percentage Condition Coverage (ACC).

This measures the rate at which a prioritized test suite covers the conditions.

(vi) Problem Tracking Reports (PTR) Metric

The PTR metric is another way that the effectiveness of a test prioritization may be analyzed. Recall that an effective prioritization technique would place test cases that are most likely to detect faults at the beginning of the test sequence. It would be beneficial to calculate the percentage of test cases that must be run before all faults have been revealed. PTR is calculated as follows:

$Ptr(t,p) = nd/n$

Let t - be the test suite under evaluation, n - the total number of test cases in the total number of test cases needed to detect all faults in the program under test p

III. Regression Testing Methods for Industry-oriented Application

Building a decision-support platform of regression testing provides a viable solution to industrial applications of regression testing. The construction involves models of business rules, application description model, change-impact-analysis, cost-risk-assessment, and test case management.

A. Extraction and Loading of Business Rules

Business rules are defined as constraints and norms for business structure and operation. They are important resources for enterprise business operations and management decisions. Business rules should be managed by the rule-based system, thereby separating application logic from the business process logic of application system. Rules engine is an embedded component in an application program. Its task is to test and compare the object data which have been submitted by the rule with the original rules, activate rules that meet the current

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

state of the data, and trigger corresponding actions in the application program, according to the rules declared in the executive logic.

To build business rules model supported by regression testing is to inherit the accumulated knowledge of senior analysts, so that there is an explicit expression for the actually used rules. On this basis, combining test theories and rules integration and optimization algorithms with the case, we can establish a generation system, which is not less efficient than an average level of case generation system in manual test.

The sources of business rules generally include:

(1) Rules derived from business needs (Rdbn)
(2) Rules derived from the theoretical testing principles (Rdtp)
(3) Rules from the industrial tradition (Rdit)
(4) Rules from the common sense of industry (Rcsi)

. This shows the Test Suite Reduction Technology has been utilized in the real industry applications. has a process for requesting and managing changes to an application during the product development cycle.

The basis of business rules model is the accumulation of a series of designing rules, industry standards, and special constraints from operations in manual test cases. Business rules model is used to express these rules in manual testing age, and establish a structure of rule engine which can be loaded rules. With these rules, a basic template case can be generated in the supportive system of decision-making for a specific business process.

Loading rules is to add a rule to the rule base. The key point is how to express the applicable conditions and specify optimization algorithms.

The expression of business rules is specific, and its basic form is If (applicable conditions of rules) Then op, among which Op both means generation of test points and case algorithms.

For a target system, it is impossible to exhaust all possibilities, it can only advance progressively. Therefore, manual addition should be allowed, and it is regarded as a learning process for business rule model.For industrial applications, tools for the source code analysis also need to extract some relationships of business process and component, component and component, component and class hierarchy, components and associated database table.

IV. CASE STUDY

[13] presents a complex industry application, they exemplify on the basis of a concrete case study (Siemens' HPCO Application, a complex Call-Center Solution) how test engineers can now work with the Integrated Test Environment. The above figure is one scenario regression test environment setting for the Call-Center Solution. We can see that even the simple scenario demonstrates the complexity of CTI platforms from the communication point of view because there are several internal protocols involved. This case study exposes the problem that in current industry practice, regression testing is intended to integrate with complex test environments. New methodology and technology should be developed to solve this problem.

The process includes:

Step 1. Collect change requests

Step 2. Identify the scope of the next release and the scope of the next release and determine which change requests will be included in the next build.

Step 3. Document the requirements, functional requirements, functional specification and implementation plans for each grouping of change requests.

Step 4. Implement the change.

Step 5. Test or verify the change. Unit testing is done by the person who made the change, usually the programmer. Function testing tests a functional area of the system to see that everything works as expected.

Step 6.Release.

A. *Factors Taken For Proposed Approach*

We consider three factors for proposed prioritization technique. These factors are discussed as follows.

(i) Rate of Fault Detection

The rate of fault detection (RFD) is defined as the average number of defects found per minute by a test case For the test case k.

$RFD_k = (N_k / \text{time } k) * 6$ (1)

(ii) Percentage of Fault Detected

The percentage of fault detected (PFD) for test case Tk can be computed by using number of

$PFD_k = (N_k)$ (2)

(iii) Risk Detection Ability

Risk value was allocated to every fault depending on the fault"s impact on software. To every fault a Risk value has been allocated based on a 10 point scale expressed as follows.

Very High Risk: RV of 10

High Risk: RV of 8

Medium Risk : RV of 6

Less Risk: RV of 4

Least Risk : RV of 2.

For test case Tk, RDAk have been computed using severity value Sk, Nk is the number of defects found by Tk, and timek is the time needed by Tk to find those defects. The equation for RDA can be expressed as follows.

$$RDA = (S_k * N_k)/\text{time } k \quad (3)$$

.

B.Test Case Ranking

Test case Ranking is the summation of the three factors which are RFD, PFD and RDA. For test case T_k, Test case ranking (TCR_k) can be calculated by the equation given below:

$$TCR_k = RFD_k + PFD_K + RDA_k \quad (4)$$

C.IIGRTCP (Improvised Industry oriented Genetic algorithm for Regression Test Case Prioritization)

The proposed prioritization technique expressed as follows.
Input: Test suite TK, and test case ranking (TCR) for every test case are inputs of the algorithm.

Output: Prioritized order of test cases.

Algorithm:

Step1. Start

Step 2. Set TK empty

Step 3. For each test case Tk ε T1 do

Step 4. Calculate test case ranking using equation (4)

Step 5.end for

Step 6. Sort TK according to descending order of TCR value

Step 7. Let TK be T

Step 8.end

V. EXPERIMENT AND ANALYSIS

The Improvised Industrial Genetic Algorithm is well suited for solving problems where solution space is huge and time taken to search exhaustively is very high. For the purpose of motivation this example assumes a priori knowledge of the faults detected by T in the program P.

Table I: Fault Matrix

Faults / Test cases	F1	F2	F3	F4	F5	F6	F7	F8
T1	X	X		X	X	X	X	X
T2	X							
T3	X				X			
T4		X	X				X	
T5				X		X		X
T6		X		X		X		

For example, suppose that regression test suite *T* contains six test cases with the initial ordering {*T1, T2, T3, T4, T5. T6*} as described in Table I.

Table II: Binary representation of Test cases

Test cases	Binary form
T1	11011111
T2	10000000
T3	10001000
T4	01100001
T5	00010101
T6	01010100

Table III: Number of faults detected by every test case, the time required to detect faults, and severity value of faults for every test case

Test cases	No of faults covered	Execution time	Risk severity
T1	2	12	8
T2	3	14	10
T3	1	11	4
T4	4	10	20
T5	2	10	12
T6	2	13	6

In Table III for the purposes of motivation, this example assumes a priori knowledge of the faults detected by *T* in the program *P*.

Table IV: RFD, PFD, RDA for test cases T1..T6

Test cases	RFD	PFD	RDA
T1	1	2	1.333
T2	1.285	3	2.142
T3	0.54	1	0.3636
T4	2.4	4	8
T5	1.2	2	2.4
T6	0.9	2	0.923

The values of rate of fault detection (RFD), percentage of fault detected (PFD) and risk detection ability (RDA) for test cases T1..T10 is calculated by using equation (1), equation (2) and equation (4) respectively. Table 4 represents the values for all three factors which are RFD, PFD, RDA for test case T1..T6 respectively.

Table V: Test case ranking for T1..T6 respectively

Test cases	**Test case ranking TCR=RFD+PFD+RDA**
T1	4.33
T2	6.427
T3	1.909
T4	14.4
T5	5.6
T6	3.8

For test cases, T1..T6, TCR value computed from equation (4) as given below. Table V shows test case ranking for each test case.

Table VI: Test cases ordering for proposed approach and previous work

Test cases	**Prioritized order**
T1	T4
T2	T2
T3	T5
T4	T1
T5	T6
T6	T3

For execution, test cases are arranged in decreasing order of TCR. Test cases are ordered in such a manner, that those with greater TCR value executes earlier

VI. RFT TOOL

A. *Features of RFT*

Rational Functional Tester software is an automated tool which provides testers with automated testing capabilities for functional testing, regression testing, GUI testing and data driven testing.

As an automated testing tool, RFT has several features below:
1) Provide robust testing support for Java, Web 2.0, SAP, Siebel, terminal-based and Microsoft Visual Studio .NET Windows Forms applications
2) Perform story board testing to combine natural language test narrative with visual editing through application screenshots
3) Use keywords to bridge the gap between manual and automated testing
4) Manage validation of dynamic data with multiple verification points and support for regular expression pattern matching
5) Reduce rework, minimize the rerecording of scripts, and reduce script maintenance

B. *SAMPLE CODE:*

```
public void testMain(Object[] args)
{
    //Get the property
    Input_Properties pro = new Input_Properties();
    String DataSharingFlag = pro.getDatasharing().trim();
    String DB2s[]= pro.getDb2s().trim().split(",");
    int RNum = Integer.valueOf(pro.getCicsregions().trim());
    String Mode = pro.getMode().trim();
    String WLType = pro.getWorkloadtype().trim();
    int TlimitNum = Integer.valueOf(pro.getCicsthreadlimit().trim());
    int UTINum = Integer.valueOf(pro.getUti().trim());
    int WarmupTime = Integer.valueOf(pro.getWarmupduration().trim());
    int MachineCPU = Integer.valueOf(pro.getMachinecpu().trim());

    //Main Procedure
    Step1_PUTTY step1_putty = new Step1_PUTTY();
    step1_putty.testMain(args,MachineCPU);
    this.sleep(5);
    Step1_HMC_Adjust_CPs step1_hmc_adjust_cps = new Step1_HMC_Adjust_CPs();
    step1_hmc_adjust_cps.testMain(args,DataSharingFlag);
    this.sleep(5);
    Step2_HMC_IPL_P58 step2_hmc_ipl_p58 = new Step2_HMC_IPL_P58();
    step2_hmc_ipl_p58.testMain(args,DataSharingFlag);
    this.sleep(5);
    RTW rtw = new RTW();
    rtw.testMain(args,DB2s,Mode,WLType,TlimitNum);
    this.sleep(5);
    Measurement measurement = new Measurement();
    measurement.testMain(args,UTINum,WarmupTime);
    this.sleep(5);
    Clean_Up clean_up = new Clean_Up();
    clean_up.testMain(args);
```

Fig 1 : Code

Fig 1 represents sample Code written

C. *Comparison with the previous work*

In this section, the proposed prioritized order is compared with previous work Table VII represents proposed order of test cases and the prioritized order proposed

Table VII: APFD % for no prioritization, Random and proposed prioritization techniques

Prioritization Technique	**APFD %**
Non Prироritized	59%
Random approach	66%
IIGRTP	88%

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

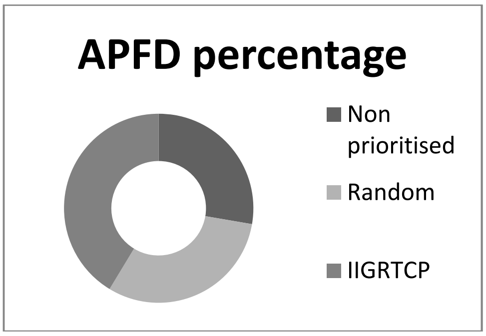

Fig 2: APFD Percentage for no order and the IIGRTP

In Fig 2 the percentage of APFD for both no order and the IIGRTP .APFD % for no prioritization and proposed prioritization techniques

VII. CONCLUSION

This paper presents a regression testing methodology for industry-oriented applications to overcome current limitations such as low degree of automation and difficulty of defining test coverage. This methodology is compared with different prioritization techniques making use of APFD metric. We take the weighted average of the number of faults detected during the execution of the test suite. The results confirm the efficacy of this proposal. Test Case Prioritization (TCP) is an effective and practical technique to monitor regression testing. It is proposed that other factors such as Weighted Defect Density (WDD), Defect Removal Efficiency (DRE), Weighted Percentage based on Fault Severity (WPFS), and popular risk metrics be incorporated in future. The proposed methodology is easily integrated with RFT Tool. Any attempt to improve functionality of regression testing that optimises resources of time and labor will result in a better software product.

REFERENCES

[1] G. Rothermel, R. Untch, C. Chu and M. Harrold, “Test case prioritization: An empirical study,” *In Software Maintenance, 1999. (ICSM’ 99) proceedings. IEEE International conference, on pages 179-188 IEEE, 1999.*

[2] A.Pravin and Dr. S. Srinivasan,”An Efficient Algorithm for Reducing the Test Cases which is Used for Performing Regression Testing,” 2nd International Conference on Computational Techniques and Artificial Intelligence (ICCTAI'2013) March 17-18, 2013

[3] S. Elbaum, A. Malishevsky, and G. Rothermel, “Prioritizing test cases for regression testing,” *Proc. The 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis, Portland, Oregon, U.S.A., August 2000, 102–112.*

[4] W. Wong, J. Horgan, S. London and H. Agrawal, “A study of effective regression testing in practice,” *In Proc. of the Eighth Intl. Symp. onSoftw. Rel. Engr., pages 230–238, Nov. 1997.*

[5] R.Beena, Dr.S.Sarala, “CODE COVERAGE BASED TEST CASE SELECTION AND PRIORITIZATION,” International Journal of Software Engineering & Applications (IJSEA), Vol.4, No.6, November 2013.

[6] R. Kavitha, N. Sureshkumar, “Test Case Prioritization for Regression Testing based on Severity of Fault,*” College of Engineering and Technology Madurai, Tamilnadu, India (IJCSE) International Jthenal on Computer Science and Engineering 2010.*

[7] Samaila Musa, Abu BakarMd Sultan, Abdul Azim Bin AbdGhani, SalmiBaharom,“A Regression Test Case Selection and Prioritization for Object-Oriented Programs using Dependency Graph and Genetic Algorithm” Research Inventy: International Journal of Engineering And Science Vol.4, Issue 7 (July 2014), PP 54-64 Issn (e): 2278-4721, Issn (p):2319-6483

[8] Sujatha, Mohit Kumar and Varun Kumar, (2010) "Requirements based Test Case Prioritization using Genetic Algorithm", International Journal of Computer Science and Technology, Vol.1, No, 2, pp.189-191.

[9] Q.-u.-a. Farooq, M. Z. Z. Iqbal, Z. I. Malik, and A. Nadeem. An approach for selective state machine based regressiontesting. In Proceedings of the 3rd International Workshop on Advances in Model-based Testing (A-MOST 2007), pages44–52, New York, NY, USA, 2007.ACM.

[10] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold. “Test case prioritization: an empirical study”. Testing “European Journal of Scientific Research ,ISSN 1450-216X Vol.55 No.2 (2011), pp.261-274

[11] S. Elbaum, A. G. Malishevsky and G. Rothermel,(2001), “Incorporating varying test costs and fault severities into test case Prioritization” ,23rd International Conference of Software Engineering, pages329-338

[12] Automated Order Management: http://www.infosys.com/industries/banking/automated_order_management.asp?page=iemsl

[13] Andreas Hagerer, TizianaMargaria, OlverNiese, Bernhard Steffen, Georg Brune and Hans-Dieter Ide, Efficient Regression Testing of CTI-Systems: Testing a complex CallCenter Solution, In Annual Review of Communication Volume 55, pages 1033-1040, Int. Engineering Consortium (IEC), 2001

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)