International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 3, Issue 5 (September-october 2015), PP. 15-22

INDUSTRY BASED REGRESSION TESTING USING IGRTCP ALGORITHM

Ms. K. Hema Shankari[1] , Dr. R. Thirumalai Selvi[2]
[1]Research Scholar , Bharath University and Assistant Professor WCC
[2] Assistant Professor, Department of Computer Science, Govt. Arts College, Nandanam, Chennai
[1]hems_banu@yahoo.com
[2]sarasselvi@gmail.com

***Abstract*— Regression testing is a testing to test the modified software during the maintenance level. Regression testing is a costly but crucial problem in software development. Both the research community and the industry have paid much attention to this problem. The paper try to do the survey of and current practice in industry and also try to find out whether there are gaps between them. The observations show that although some issues are concerned both by the research community and the industry. This research discusses the problems about current research on regression testing and quality control in application of regression testing in the engineering practice, and proposes a practical regression method, combing with change-impact-analysis, business rules model, cost risk assessment and test case management. This paper presents an approach to prioritize regression test cases based on the factors such as rate of fault deduction, percentage of fault detected and the risk detection capability. The proposed approach is compared with previous approach using APFD metric. The results represent that propose approach outperforms the earlier approach.**

***Index Terms* Regression testing; Change-impact-analysis; Cost-risk-assessment; Industry application; Business rules, Test case prioritization, APFD metric.**

I. INTRODUCTION

In software production the software maintenance activity is an expansive phase which cost nearly 60% of total cost. Regression testing is an important phase in software maintenance activity to ensure the modification caused by debugging. It provides confidence that changes do not harm the existing behavior of the software. With the wide application of IT in various industries, software systems have became extremely important. The reliability of the system is playing plays a key supporting role for the application business development. Development and maintenance are always accompanied the entire life cycle of application systems. As a result, there is a growing demand for regression testing.

Industry application regression testing has its inherent characteristics. The first is business-related. Application changes mainly come from business development. Because new business and old business are inextricably linked, which bring great difficulties to define the scope of regression testing. The second is iteration. Regression testing is an iterative process, a new round of testing has a great similarity with the pre-test, so how to reuse historical accumulation of test resources, and improve testing automation to efficiently complete testing are worthy of further study.

The Test case prioritization techniques [4] intend to arrange test cases for regression testing in such a manner, with the goal of amplifying some criteria. Rothermel et al. [1] and Elbaum et al. [3] proposed a variety of test case prioritization techniques to the boost fault detection rate. Test case prioritization can address to boost a diversity of objective functions such as rate of fault detection, rate of detection of high-risk faults, likelihood of revealing regression errors, coverage of coverable code, and confidence in the reliability of the system under test [1]. Numerous techniques have been investigated to arrange test cases for regression testing, with an attempt to test modified software, nine different test case prioritization techniques have been explained by Rothermel et al. [1]. We have presented an approach for prioritizing regression test cases on the basis of three factors which are rate of fault detection (RFT), percentage of fault detected (PFD) and risk detection ability (RDA). RFT is defined as the average number of defects found per minute by a test case [7]. PFD is the percentage of fault detected by a test case. RDA is

defined as the ability of test case to detect severe faults per unit time.

A. . Problems for Regression Testing in Industry Application

Problems Faced

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 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.

In regression testing, reusing of used cases can greatly improve test efficiency, and reduce time and duplication of effort. Therefore, there is a huge test case library at the supportive platform. It indexes all the used cases in catalogue to associate the specific cases with related businesses, and facilities the reference of cost-assessment model and the automatic generation of the test scripts.

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. . 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.

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.
In the age of manual testing, test requirements analysis is based on the analyst's personal understanding of applicable rules, so the analysis of the results is different from one to another. Furthermore, because the rules are not explicitly expressed, there are some problems on integrity and consistency of the rules in practice. At the same time, those rules are the accumulation of a senior analyst's the experience, with time passing by, it formed an industry tradition of manual testing times.
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 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 (Rbn)
(2) Rules derived from the theoretical testing principles (Rtp)
(3) Rules from the industrial tradition (Rit)
(4) Rules from the common sense of industry (Rcs)

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.

D. . CASE STUDY

Seapine Software has a process for requesting and managing changes to an application during the product development cycle. 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. Regression testing is system-wide to insure that all areas of the system still function as expected. This validates that the change caused no unexpected side effects and that the system still has the overall functionality it had before the change.

Step 6. Release.

What need to emphasize here is that the regression testing is required to be system-wide, also mentioned that in the ndustry, multi level regression testing is required to insure the software quality. Since in practice, large systems may be developed in different stages and by different teams. During the process, testing is involved in almost every stage, such as unit testing, functional testing, acceptance testing and field testing. So in practice, regression testing is required to be embedded in every mature software development process, since for every change of the software, no matter how tiny it is, regression testing must be applied.

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.

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

(ii) Percentage of Fault Detected

The percentage of fault detected (PFD) for test case Tk can be computed by using number of faults found by test case Tk and total number of faults, expressed as follows.

$$PFD_k = (N_k / N) * 6 \qquad (2)$$

(iii)Risk Detection Ability

It can be defined as the ability of test case to detect severe faults per unit time. Testing efficacy could be progressed by emphasizing on test cases which detect greater percentage of severe faults (RDA). 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.

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

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:

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

IGRTCP (Industry oriented Genetic algorithm for Regression Case Prioritization)

The proposed prioritization technique expressed as follows.
Input: Test suite T, 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 T‟ empty
Step 3. For each test case Tk ε T do
Step 4. Calculate test case ranking using equation (4)
Step 5. end for
Step 6. Sort T according to descending order of TCR value
Step 7. Let T‟ be T
Step 8. end

E. EXPERIMENT and analysis

The Improved Genetic Algorithm is well suited for solving problems where solution space is huge and time taken to search exhaustively is very high. Another advantage of this algorithm is that it has the ability to solve problems with no previous knowledge. For the purpose of motivation this example assumes a priori knowledge of the faults detected by T in the program P.

TABLE 1: Sample data of Test cases

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 {T_1, T_2, T_3, T_4, T_5. T_6} as described in Table 1.

TABLE 2: Binary representation of Test cases

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

The binary representation of test cases is shown in Table2. The bit id is 1 (high) if it covers the respective fault starting from the leftmost position otherwise it is 0 (low).

TABLE 3: 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 3 for the purposes of motivation, this example assumes a priori knowledge of the faults detected by *T* in the program *P*.

Table 4. RFD, PFD, RDA for test cases T1..T6

Test cases	RFD	PFD	RDA
T1	1.66	2	1.333
T2	2.142	3	2.142
T3	0.9	1	0.3636
T4	4.0	4	8
T5	2.0	2	2.4
T6	1.538	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 5. Test case ranking for T1..T6 respectively

Test cases	**Test case ranking TCR=RFD+PFD+RDA**
T1	4.993
T2	7.284
T3	2.263
T4	16
T5	6.4
T6	4.461

For test cases, T1..T6, TCR value computed from equation (4) as given below. Table 5 shows test case ranking for each test case.

Table 6: 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

F. Comparison with the previous work

In this section, the proposed prioritized order is compared with previous work Table 7 represents proposed order of test cases and the prioritized order proposed

Table 7: APFD % for no prioritization, and proposed prioritization techniques

Prioritization Technique	**APFD %**
No order	64
Proposed order	85.4

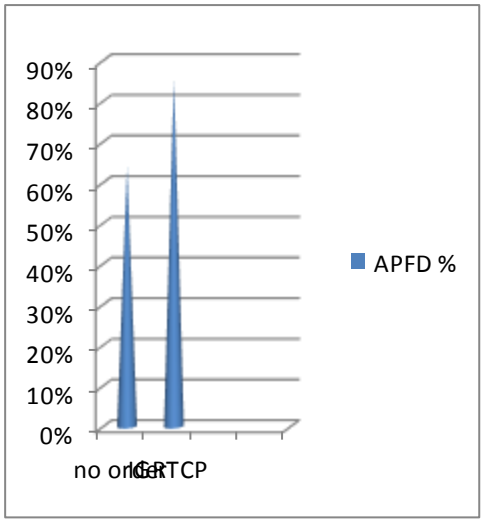

Fig 1 : APFD Percentage for no order and the IERTCP

In Fig 1 the percentage of APFD for both no order and the IGRTCP . APFD % for no prioritization and proposed prioritization techniques

II. Conclusion

Software tests, especially software represented by regression testing, it accompanies the whole life cycle of industrial application system. This paper presents a regression testing method for industry-oriented applications to solve issues, such as the low degree of automation of large-scale business systems and difficulty of defining test coverage. For every test case all the three factors are calculated and test case ranking is computed by adding these factors for each test case. To solve the problem of test case prioritization we prioritize test cases, according to decreasing order of test case ranking value, and we obtain the prioritized order of test cases. The proposed approach is compared with different prioritization techniques such as no ordering, using APFD metric. The APFD is calculated by taking the weighted average of the number of faults detected during the execution of the test suite. The results represent that proposed approach outperformed all approaches mentioned above. Benefiting from this method, decision-support platform of regression testing has been applied to a number of large financial systems and made great achievements.

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. on Softw. 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] Alex Groce, Todd Kulesza, Chaoqiang Zhang, ShaliniShamasunder, Margaret Burnett, Weng-Keen Wong,SimoneStumpf, Shubhomoy Das, Amber Shinsel, Forrest Bice, and Kevin McIntosh," You Are the Only Possible Oracle: Effective Test Selection for End Users of Interactive Machine Learning Systems," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 40, NO. 3, MARCH 2014

[7] 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.

[8] 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.

[9] 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.

[10] R. Kavitha and N.Suresh Kumar, "Factors Oriented Test Case Prioritization Technique in Regression 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, pages 329-338