TEST CASE PRIORITIZATION FOR REGRESSION TESTING

1) Mr. MANOJ KUMAR SAHU,

ASSISTANT PROFESSOR,

NIIS GROUP OF INSTITUTIONS, BHUBANESWAR, ODISHA.

2) Mr. ALOKA NATHA

ASSISTANT PROFESSOR,

NIIS GROUP OF INSTITUTIONS, BHUBANESWAR, ODISHA.

Abstract- **Regression testing is the process of testing changes made to software to make sure that the newer version still works correctly with the new changes. Regression testing is an important part of the software development process and is done by testing specialists. Before a new version of software is released, the old one is run against the new version to make sure that all the old changes still work. The reason behind this is because changing or adding new code to a program can easily introduce errors into code that is not intended to be changed. But due to time and cost constraint, retesting cannot be performed. Thus, it becomes necessity to reduce the test case suite and select a subset of test cases from test suite that can be executed in minimum time and has ability to cover all the faults. Faults in regression testing artifacts may cause the application to extract wrong data from messages, leading to failure in service composition. Surprisingly, current regression testing researches hardly consider these artifacts. In this thesis, a new novel approach is proposed to generate prioritized test case suite using Business Criticality Value in regression testing. The algorithm will be validated and its performance will be compared with the existing ones.**

Keywords- Regression Testing, Test Case Prioritization, Business Critical Value

I. INTRODUCTION

Regression testing [12] is selective retesting of the system; or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

Testing with a Purpose

Software testing [12] is performed to verify that the software package functions according to the expectations defined by the software requirements specifications (SRS). The overall objective is not to find every software bug free, but to uncover situations that could negatively impact the customer, reliability and usability.

From the module level to the application level, this report defines the different types of testing techniques. Depending upon the testing purpose for the software requirements, a combination of testing techniques is applied. One of the most overlooked areas of testing is regression testing and fault tolerant testing.

The ideas and techniques of software testing are the essential knowledge for all software developers those who are going to developed a new product. Here the concepts of software testing are introduced by describing the activities of a test suite, test cases, defining a number of key terms and terminologies, and then explaining the main aim of test coverage. Software is considered as an important medium for many of the devices and systems that pervade our society. Software defines the uses and applications of network components, financial networks, and telephone switching devices, the World Wide Web, and other important concept of modern life. Software is an important component of many more applications that control real time applications such as aircraft management, space shuttles, and traffic control systems, as well as electronic appliances such as watches, ovens, buses, cars, DVD players, cell phones, and remote controller devices. Modern household's appliances have over 1000 micro processors, and some new vehicles have over 100; all of them running software that optimistic consumers assume will never fail! Although many factors affect the engineering of valuable software, careful design and sound processing management, testing is the primary technique which an industry uses to evaluate software products under development. Fortunately, a very few basic software testing methodologies can be used to design tests for a large variety of software applications. A goal of this thesis is to present these concepts in such a way that the student or practicing engineer can easily apply them to any software testing situation.

II. MOTIVATION

- As model based test case prioritization is more effective because of its faster execution and less expensive than code based test prioritization so

model based test prioritization is used in proposed methodology.

- When we retest our test suite it takes more time to re execute, so prioritization helps in rescheduling the test cases so that higher priority test cases will execute before low priority test cases.
- Prioritization helps in early fault detection. As model based test prioritization is more effective than code based test prioritization so we motivated to use model based test prioritization in my subsequent research.

III. OBJECTIVE

• A prioritization technique for test cases will be developed according to the weight of the different test cases of the project. Prioritization can be done according to the following criteria:

• For each function assign a *business critical value* (BCV), depending on the contribution of the function towards the quality of the product, which should act as one of the criteria for prioritization?

• The information will be collected from past data i.e. the various attributes in the test suites database for a particular type of change made, what are the corresponding *risk_factor* occurring. Accordingly the test cases will be prioritized.

• Depending upon the *test case weight* (TCW) prioritization can be done.

In this chapter we discuss about a novel approach to prioritize test case in case of regression testing.

Test Case Prioritization for Regression Testing using Business Critical Value

Regression testing is the process of testing a modified system using the old test suite. As the size of the test suite is large, retesting of the system acquires large amount of time and computing resources. This issue of software systems retesting can be handled using a good test case prioritization technique. A prioritization technique schedules the test cases for execution so that the test cases with higher priority executed before lower priority. The objective of test case prioritization is to detect fault as early as possible so that the debuggers can begin their work earlier.

In this chapter we propose a new prioritization technique to prioritize the test cases to perform regression testing for model based approach.

In this section we discuss our proposed approach to generate a prioritize test cases. The below figure depicts the idea of a new model approach. The key factor of our model based approach is based upon the factors: *business critical value (BCV), risk_factor*, and *TCW (Test Case Weight)*. The below diagram depicts our model based approach in detail.

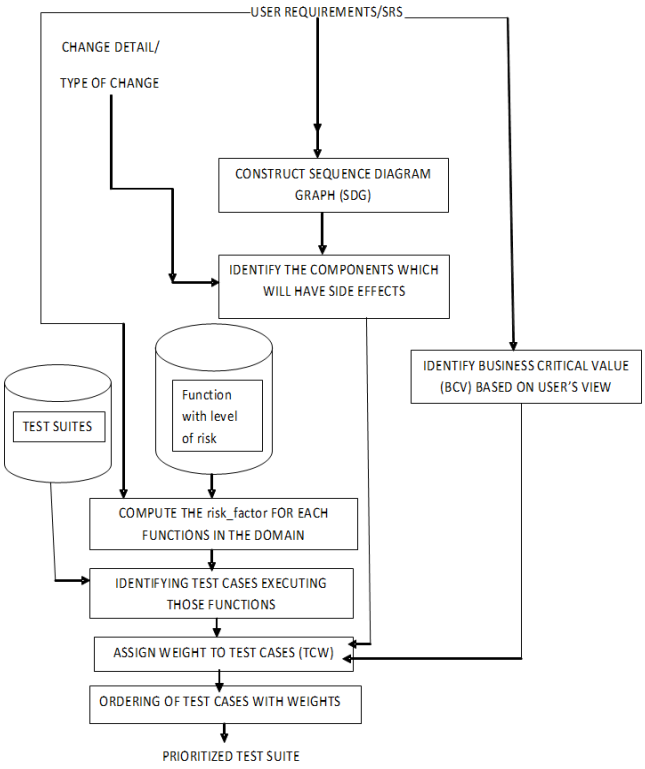

Figure 1: A Model for Test Case Prioritization

Our approach consists of following steps:

1. Construct the *Sequence Diagram Graph (SDG)*
2. Identify the components which will have side effects
3. Test suites and Function with level of risk
4. Compute the *risk_factor* for each function in the domain
5. Identifying test cases executing those functions
6. Assign weight to test cases
7. Identify *business critical value*
8. Ordering of test cases with weights

The above steps are defined as follows in details:

1. Construct the Sequence Diagram Graph (SDG)

Based upon the user requirements or often called as Software Requirement Specifications (SRS), the input is to be considered. The requirements must be of more beneficial towards the design or to propose a new model. Based upon the requirement analysis the newly invention is to be proceed. The SRS is module using the UML sequence diagram. In the sequence diagram, the objects are interacting with each other by passing messages. A Sequence Diagram Graph (SDG) is constructed from the sequence diagram. Then the output is going to fed to the next level, i.e. to identify the components which may have side effects due to the changes.

2. Identify the components which will have side effects

The type of change or change in detail is also considered is this stage. Here in the regression testing, there are many test cases are using. Among of them there are several components and among of them we have to find out the components which will have side effects.

3. Test suites and *Function with level of risk*

In the figure we are maintaining two databases, first is for storing the *test suites* i.e. originally the old test suites and second is for storing the functions which is having the various risk levels. From taking the inputs *SRS* stage and *Function with level of risk*, we have to move towards the next stage where we have to measure the *risk_factor*.

Table 1: Testing issues and their respective attributes and linkage

Issues	**Attribute**	**Related linkage**
Test suite	• Creation data • Execution data • Execution summary	• Test case to test case • Test suite execution summary
Function with level of risk	• Risk analysis • Risk_factor	• Risk_factor calculated • Risk_factor summary
BCV	• Business Critical Value analysis	• Business critical value factor
TCW (Test Case Weight)	• Test case data • Execution status • Execution history	• Test case priority • Test case execution summary

4. *Compute the risk_factor for each function in the domain*

Risk analysis should be performed as part of the risk management process for each project. The outcome of the risk analysis would be the creation or review of the risk register to identify and quantify risk elements to the project and their potential impact. Taking the inputs from the database test suite and the previous stage, we then move to the next step i.e. to identify test cases which are executing these functions.

5. Identifying test cases executing those functions

There are several test cases are used. For better consideration and for executing the test cases we have to identify the test cases which are executing those functions. Here the input is coming from the databases which we are using as *test suites*.

6. Assign weight to test cases

Here we have to assign the weight to the test cases. In this stage also the inputs are to be considered i.e. from identify the components which will have side effects and identify business critical value (BCV) which is based on user's view. The assigning of the weight to the test cases is also known as *TCW (Test Case Weight)*.

7. Identify business critical value based on user's view

A Business Critical Value (BCV) is defined as the "amount of the function contribution towards the success of the project implementation." For a Stock Inventory System, a company is giving more and more emphasis to manufacturing the product, product's quality rather than the product's availability. It gives more focus on the production. Here the product's quality is having much more business value than others. Also the *risk_factor* for manufacturing the product is also highly defined. Likewise customer's feedback and complaint is having more business critical value than customer's enquiry.

After assigning the weights to the test cases, whatever the output is coming, it is going to the next stage i.e. ordering of test cases with weights.

8. Ordering of test cases with weights

The test cases are already assigned the TCW. The test cases are now goin to be ordered so that the prioritization comes to work and lastly we get the prioritized test suite.

IV. Proposed Prioritization Algorithm

After computing the test case weight for each of the test case, on the basis of TCW each test case will be prioritize with the help of proposed Prioritization technique is presented in an algorithmic form here under: The input of the algorithm is test suite T, test case weight of each test case is computed and the output of the algorithm is prioritized test case order.

Proposed Algorithm for calculating TCW:

Algorithm 1:

(BCV- Business Critical Value, TCW- Test Case Weight**)**

Input: - Business Critical Value, risk_factor.

Output: - test case weight in ascending order.

1. Begin
2. Set T' empty
3. For each test case t ε T do
4. Calculate TCW = BCV + risk_factor
5. End for
6. Sort T in descending order on the value of test case weight
7. Let T' be T

Proposed Algorithm for Test case Prioritization:

Algorithm 2:

(TC - Test Case List, BCV- Business Critical Value, TCW- Test Case Weight)

Input: - Test suites Ti, number of functions with level of risk.

Output: - Prioritized test cases.

1. Merge all test cases
2. The test cases Ti associated with the *risk_factor* and the test case Ti, which has association with consecutive maximum coverage of faults.
3. Calculate the *BCV* from the test case.
4. Select the test case Ti that has highest *risk_factor*.
5. If Ti has association with *risk_factor* then go to Step 6.
6. Assign *TCW* to the test case Ti, by calculations.
7. Sort the TC in descending order based on the maximum coverage by each test case.

Proposed formula for calculating BCV (Business Critical Value):

The BCV is used to measure out the criticality factor of each function. The consideration is taking or being carried out by the following formula:

$$BCV = \frac{\text{Number of faults}}{\text{Total number of functions}}$$

The proposed algorithms 1 and 2 along with the formula given in equation 1 will help in prioritizing the test cases in case of regression testing.

V. GSM BASED ATM SYSTEM: A CASE STUDY

Here we consider an example of a GSM based ATM system. In the ATM system we are focusing some major functions depending upon which we have to find out the BCV values. In the below we discuss about the use case analysis, activity diagram and test case specifications.

Use Case Analysis

The information gathered as a part of the analysis activity is used to create an analysis model for the interface. Use-Case is developed to show how an end-user performs some specific work-related task. Use-Case provides a basic description of one important work for computer aided design system.

Use Case Specification

- A control flow of events from one to another is created for each use cases
- Written from an actor point of view
- The system details must provide to the actor when the use cases is executed
- Typical contents
  - How the use case starts and ends
  - Normal flow of events
  - Alternate flow of events
  - Exceptional flow of events

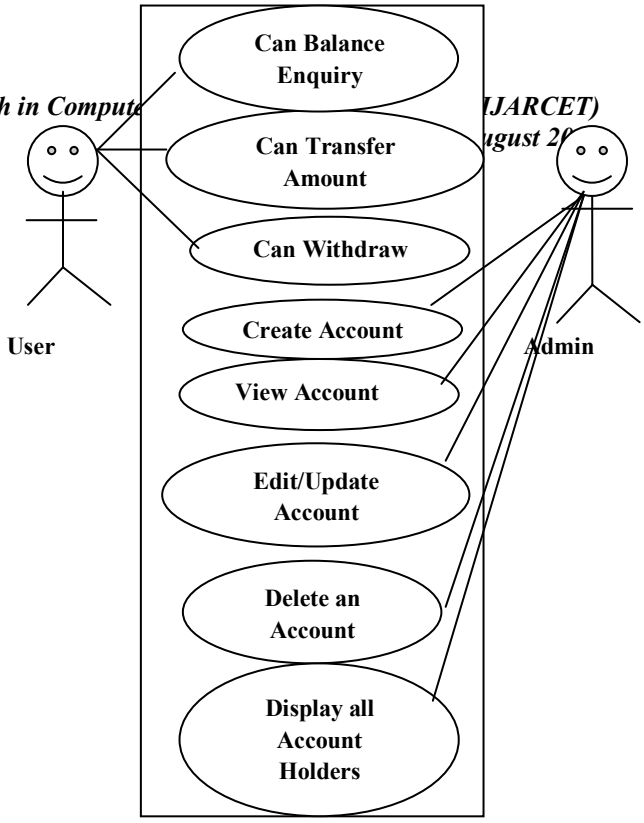

Figure 2: Use- Case specifications of GSM based ATM system

Now construct the Activity diagram for the GSM Based ATM system.

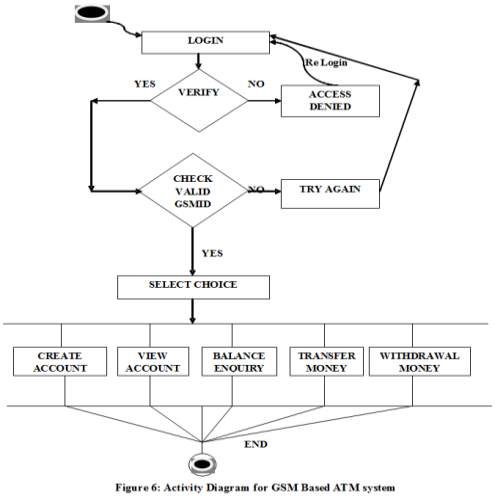

Figure 3: Activity Diagram for GSM Based ATM system

VI. Test Case Specification

The success of testing in revealing errors in programs depends critically on the test cases. During testing, the program to be tested is executed with a set of test cases, and the output of the program for the test cases is evaluated to examine if the program is performing as expected

Table 2: The test case specifications

Test Case ID	**Test Case Name**	**Input**	**Expected output**	**Actual output**
T1	Login	Click on the Login button with proper input	Display the user form/admin form	Display the user form/admin form
T2	Exit	Click on the exit button	All Application will be closed	All Application will be closed
T3	GSM PIN	Click on the button and wait for some time	To get the valid GSMID	To get the valid GSMID
T4	Create Account	Click this option after login	It will create a new account for a new user	It will create a new account for a new user
T5	Delete account	Click on this option for deletion	It will delete an account	It will delete an account
T6	Edit account	Click on this button for updating	It will modify/update the account holder details	It will modify the account holder details
T7	Users	Click on this option to show all the account holders	It will display all the current account holders	It will display all the current account holders
T8	Logout	Click on this button	It will close the current application and goes back to Login form	It will close the current application and goes back to login form
T9	Balance enquiry	Click on this button	It will display the current account holder's balance amount	It will display the current account holder's balance amount
T10	Withdraw	Click on this option	Here we can withdraw our required amount of money	Here we can withdraw our required amount of money
T11	Transfer	Click on this button	This transfer money from one account to other	This transfer money from one account to other

The numbers of faults are detected during the examining of different test cases. There are some functions are taking into consideration as follows:

Table 3: the BCV values and Faults Detected with Functional Values

Test Case ID	Functions	Faults Detected	BCV Value
T1	2	FT1, FT20	1.000
T2	5	FT2, FT5, FT6, FT8, FT11, FT15, FT16, FT20	1.600
T3	2	FT3, FT4, FT6, FT7, FT18	2.500
T4	2	FT1, FT5, FT6, FT9. FT17	2.500
T5	3	FT4, FT6, FT7, FT10, FT17, FT19	1.333
T6	10	FT3, FT4, FT6, FT8, FT9, FT13, FT14, FT16	0.800
T7	20	FT3, FT5, FT7, FT8, FT10, FT12, FT13, FT14, FT16	0.450
T8	3	FT2, FT5, FT9, FT17, FT18, FT20	1.000
T9	3	FT1, FT3, FT4, FT8, FT9, FT11, FT13	3.333
T10	5	FT2, FT5, FT11, FT13, FT14, FT16, FT19	1.400
T11	6	FT5, FT7, FT8, FT9, FT11, FT13, FT15	1.166

After calculating the BCV, now we have to calculate the TCW i.e. Test Case Weight. Before going to calculate the TCW now we have to find out the risk_factor of each test case. The risk_factor value can be assigned to different test cases depending upon the client's perspective and expert entry and the value is defined within the range 0 to 6. The GSM based ATM System is having more different types of functionalities. We are assuming some critical issues which are arising upon the user's specifications. Based upon the function's critical point, the value of risk_factor is taking into consideration. The below table shows about the detailed values of BCV and risk_factor.

Table 4: risk_factor values along with BCV Value

Test Case ID	BCV Value	risk_factor Value
T1	2.000	2
T2	1.200	3
T3	2.000	2
T4	2.000	3
T5	1.000	1
T6	0.800	5
T7	0.450	6
T8	1.000	1
T9	3.333	1
T10	1.400	5
T11	1.166	4

Now taking the values of BCV and risk_factor, we have to calculate the TCW (test case weight) as following manner:

Table 5: TCW Values against Test Case ID

Test Case ID	TCW
T1	4.000
T2	4.200
T3	4.000
T4	5.000
T5	2.000

T6	5.800
T7	6.450
T8	2.000
T9	4.333
T10	6.400
T11	5.166

From the above calculations, we observed that the test cases are prioritized based upon the final value of TCW. After getting the value of TCW now we get the prioritized order of the above test cases as follows:

T7, T10, T6, T11, T4, T9, T2, T3, T1, T5, T8

Now the efficiency of prioritized and non-prioritized test suite are compared using APFD metric.

Comparative Study of Proposed Work

Now we have to make a comparison on the given data values with an existing data values. The below table shows the number of faults detected by the test cases in the test suite against the corresponding time. Based upon the calculated values we have to analyze the conclusion.

Table 6: Prioritization Table Calculation

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
FT1	*					*					
FT2				*			*	*	*		
FT3						*				*	
FT4							*				
FT5						*		*		*	*
FT6						*		*			
FT7										*	
FT8		*							*		
FT9						*		*			
FT10										*	*
FT11				*					*		*
FT12							*				
FT13						*	*		*	*	*
FT14						*	*			*	
FT15			*								*
FT16						*	*			*	
FT17				*	*			*			
FT18			*					*			
FT19				*	*					*	
FT20							*	*			
No. of Faults	1	1	2	4	2	8	7	7	4	8	5
Time	5	7	11	4	8	12	6	4	8	9	5

APFD Result:

Here we are set the priority according to the decreasing order values of TCW, since more the priority is to be considered as it covers maximum number of faults.

Hence the prioritized order is:

T7, T10, T6, T11, T4, T9, T2, T3, T1, T5, T8

In the above table

m= number of faults = 20

n= number of test cases = 11

So now putting the values of m, n, TF_i in the equation

$$APFD = 1 - \frac{(TFT_1 + TFT_2 + \ldots\ldots + TFT_m)}{(n*m)} + \frac{1}{2*n}$$

APFD value for prioritized test case:

$$APFD = 1 - \frac{9+5+3+1+3+3+2+7+3+2+5+1+3+3+8+3+5+8+5+1}{(20*11)} + \frac{1}{2*11}$$

$$= 0.69$$

APFD value for non-prioritized test case:

$$APFD = 1 - \frac{1+4+6+7+6+6+10+2+6+10+4+7+6+6+3+6+4+3+4+7}{(20*11)} + \frac{1}{(2*11)}$$

$$= 0.55$$

Analysis of APFD

The comparison is drawn between prioritized and non-prioritized test case, which shows that value obtained for prioritized case is more as compared to non-prioritized case. Also we compared the APFD value with the existing approach of APFD and get the better results.

VII. CONCLUSION

The regression testing is a very important and powerful testing technique among several testing methodologies. It is taking into consideration that the testing is done after coding and before implementation. The test case prioritization techniques are used for prioritized the regression test suites. There are techniques which are categorized into two types named as following: a) Code Based Test Case Prioritization and b) Model Based Test Case Prioritization.

The code based test prioritization techniques are based only upon the source codes as it is giving more emphasis on coding. In the code based test prioritization, the idea is to use the source code of the system to prioritize tests. These code based prioritization techniques are mostly dependent on information relating the tests of the test suite. But in model based test case prioritization, the idea is used on system models to prioritize tests.

We implement here a new model approach for model based test case prioritization and also which gives the better APFD value as compared to the existing ones. We calculate BCV values, TCW values with a great extent for this model based approach to increase the APFD metric value. This report proposed an algorithm for test case prioritization in order to improve the regression testing. Analysis also is done for both prioritized as well as non-prioritized cases with the help of APFD metric and also we found out the result; hence the prioritized case is more effective. An effective regression testing approach saves the organizations both time and money.

VIII. FUTURE WORK

- We can further extend our proposed work to validate the proposed model.
- We can derive an algorithm for prioritized test cases by using soft computing techniques such as GA (Genetic Algorithm) approach, PSO approach.
- We can implement this course work in a real time as well as embedded systems.

REFERENCES

[1] N. Panda, A. A. Acharya, D. P. Mohapatra. Model based test case prioritization for testing component dependency in cbsd using uml sequence diagram. In *IJACSA*, volume 1, pages 108−113, December 2010.

[2]C. Chu, G. Rothermel, R.H. Untch and M.J. Harrold. Test case prioritization: An empirical study. *Proc. Int'l Conf. Software Maintenance*, pages 179−188, Sep1999.

[3] M. Harrold, G. Rothermel, R. Untch. Prioritizing test cases for regression testing. In *IEEE Transactions on Software Engineering*, volume 27, pages 929−948, Oct 2001

[4] Rajib Mall. *Fundamentals of Software Engineering*. Prentice-Hall of India, New Delhi, Third Edition, July-2010.

[5] G. Rothermel, S. Elbaum, A. Malishevsky. Test case prioritization: A family of empirical studies. In *IEEE Transactions on Software Engineering*, volume 28, No. 2, pages 159−182, Feb 2002.

[6] Praveen Ranjan Srivastava. Test case prioritization. In *Journal of Theoretical and Applied Information Technology, JATIT*, pages 178−181, 2005 – 2008

[7] Pankaj Jalote. An Integrated Approach to Software Engineering. Narosha Publishing House, New Delhi, Third Edition-2009.

[8] George Koutsogiannakis, Bogdan Korel. Experimental comparison of code-based and model-based test prioritization. In IEEE International Conference on Software Testing Verification and Validation Workshops, pages 77–84.

[9] Prem Parashar, Arvind Kalia, Rajesh Bhatia. How time-fault ratio helps in test case prioritization for regression testing, pages 25-35.

[10] Nilam Kaushik, Mazeiar Salehie, Ladan Tahvildari, Sen Li, Mark Moore. Dynamic Prioritization in regression testing, Department of Electrical and Computer Engineering, University of Waterloo Research in Motion, Canada. Fourth International Conference on Software Testing, Verification and Validation Workshops, 2011.

[11] Chhabi Rani Panigrahi, Rajib Mall, Model-Based Regression Test Case Prioritization, ACM SIGSOFT Software Engineering Notes Page 1, November 2010, Volume 35, Number 6

[12] Naresh Chauhan, Software Testing (Principles and Practices), Oxford Higher Education, Third Edition 2012.

[13] Srinivasan Desikan and Gopalaswamy Ramesh, Software Testing: Principles and Practices, 1/e, Pearson Education-2006.

[14] Gaurav Duggal, Mrs. Bharti Suri, Understanding Regression Testing Techniques, Guru Gobind Singh Indraprastha University, Delhi.

[15] Rajib Mall, Swarnendu Biswas, A Model-Based Test Selection Approach for Embedded Applications, SIGSOFT Software Engineering Notes, page 1-9, July 2009 Volume 34 Number 4.

[16] Chuanqi Tao, Bixin Li, Jerry Gao, A Model Based Approach to Regression Testing of Component Based Software.

[17] Lijun Mei, Zhenyu Zhang, W.K. Chan, T. H. Tse, Test Case Prioritization for Regression Testing of Service Oriented Business Applications, WWW 2009, ACM Press, New York, NY 2009.

