Cost Effective Model Based Regression Testing

Prabhakar K., A. Ananda Rao, K. Venu Gopala Rao, S. S. Satyanarayana Reddy, M. Gopichand

Abstract: **Regression testing is a software quality assurance activity performed frequently on modified software in maintenance. This re-testing costs a lot for maintenance in terms of effort and computing resources. The existing approaches for test case-optimization, prioritization, reduction, to improve the cost effectiveness of the regression testing are not sufficient for handling the mentioned problem. Therefore, this research proposes a holistic approach to derive test cases from behavioral models for regression testing which estimates test effort and detection of all the errors. The use cases are considered primarily for identifying defects and for reducing the number of test cases. This idea has been evaluated using Automatic Teller Machine (ATM) and satisfactory results are obtained. It is also observed that this method reduces the time and cost of the regression testing considerably.**

Index Terms**—** Test-suit, Use case point, Behavioral model, Software maintenance, Quality assurance.

I. Introduction

REGRESSION testing validates the modified software to confirm that the modifications are not badly affecting the unchanged parts of software [1]. The model based techniques have been used to generate test cases for the behavioral model of a software system. Execute test cases automatically or manually enables early detection of requirement errors [2]. In this automated test design, regression test suite design is challenging and important task.

This paper proposes an approach for cost effective regression testing. In contrast to prevailing approaches its main focus is attempts to maximize the test coverage. This method also facilitate effort estimation where Use Cases are used to derive test cases and applied in Re-testing in any kind of software maintenance. The use case model is taken form the behavioral models of unified modeling language. The use case model will identify all functionalities of a software system like *<Main flows>*, *<Alternative flows>*, *<Includes>*, *<Extends> and other <Relations>*. From all these use cases, complete test cases are generated. Quality Assurance (QA) team needs run all these test cases to ensure that the software product is stable.

Prabhakar K. Research Scholar, Dept. of CSE, JNTUA. nanthapuramu, India., Mobile: 9963039900, Email Id: prabhakarcs@gmail.com

Ananda Rao, Professor Dept. of CSE, DAP, JNTUA. Ananthapuramu, India, Mobile Id: 9440990090, Email Id: akepogu@gmail.com

K.Venugopala Rao, Professor Dept. of CSE, GNITS, Hyderabad, India, Mobile Id: 9849025342, Email Id: kvgrao1234@gmail.com

S. Sai Satyanarayana Reddy, Professor Dept. of CSE, Principal,VCE, Hyderabad, India, Mobile Id: 9502653333, Email Id: saisn90@gmail.com

M. Gopichand, Professor & Head, Dept. of IT, VCE, Hyderabad, India, Mobile: 9849042448, Email Id: gopi_merugu@yahoo.com

This paper structured as follows. The background and related work is given in Sec.-II. Model based regression testing and effort estimation is presented in Sec.-III. Finally the results and discussions are placed in Sec.-IV and conclusions & future enhancements are given in Sec.-V.

II. Related Work

According to L. Erlikh, 85-90% of the projects are under maintenance. So, it shows the importance of regression testing in software maintenance [3]. Jim Heumann, generated test cases from use cases [4]. This paper explains the process of generating test cases from the basic behavioral model called use cases but did not address regression testing. Bogdan korel used state machines for test reduction [5]. Yanping, explained regression test suit reduction using dependency analysis with state machines [6]. Selvakumar, explained extended dependency analysis for test suit reduction [7]. The state machines are used to reduce the test suit, but their main focus is on data dependencies and control dependencies only [5, 6, 7].

III. Model Based Regression Testing

To develop any quality software, test cases play a vital role. As per the existing techniques of testing, Model Based Testing (MBT) techniques are mostly used for system testing. i.e. (66% of all the techniques) [8]. As using MBT in testing is very complex in reality, it is very less used in regression testing i.e. only 5% compared to all the other techniques.

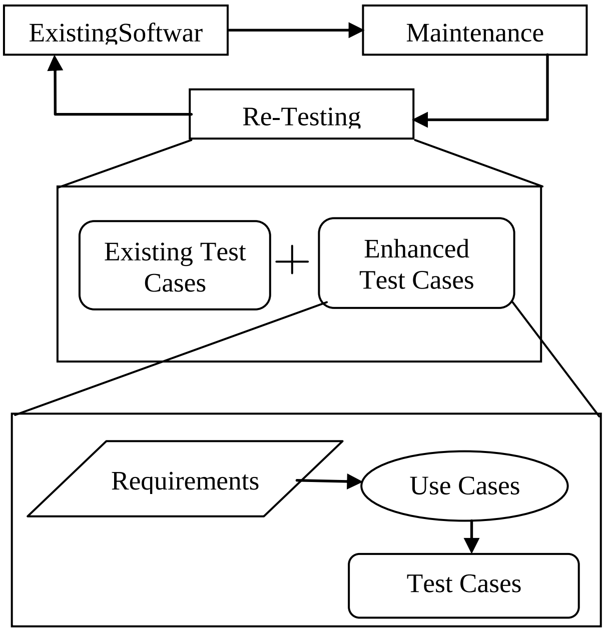

Fig.**1**. Model based regression testing approach.

The proposed approach is presented in Fig.1. Using this model enhanced test cases are generated from the behavioral

ISBN: 978-988-14047-4-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

models. Testing effort can be estimated very easily by this model.

Whenever a software system comes for maintenance, it has to fall under any one of four maintenance categories. That is maintenance may be a corrective, adaptive, preventive or perfective maintenance. After identification of maintenance type, the Change Impact Analysis (CIA) should be done on existing system. Then the maintenance people will do necessary modifications. Now it is the job of QA team to test the modified software by executing all test cases [9]. This testing is called Re-Testing or Regression Testing.

In fig.1, the retesting is done with the combination of existing test cases and Enhanced test cases. In the proposed model enhanced test cases are generated with help of behavioral models. That is by considering the changed, proposed, affected requirements and their related artifices [10]. After this retesting is performed by considering these generated test cases as well as existing test cases. This process will be continued for further maintenance also.

To estimate effort that is required to carry out the software maintenance activity, the flowing equation (Eq…1) is used.

Regression Testing Effort =Verification of fixed bugs + EUP.   Eq...1

In the above mentioned equation, first component i.e. verification of fixed bugs, generally 20 minutes time is required to run the script [1].

A. *Generation of Test-Case from Models*

Test Cases can be generated from different behavioral models like classes, use cases, state machines...etc. These test cases can be used in different testing activities like unit testing, integration testing, system testing and regression testing. These test cases can be applied for both development paradigms and execution environments. Here use case models are used for generating test cases, because these models are very much closed to the behavior of the software system and its related artifacts [4].

B. *Use Case Model: Case Study &Results*

The Unified Modeling Language (UML) is providing fourteen diagrams to model the software system. Among fourteen, the Use-Case diagrams represent behavior of the software system intended by the customer.

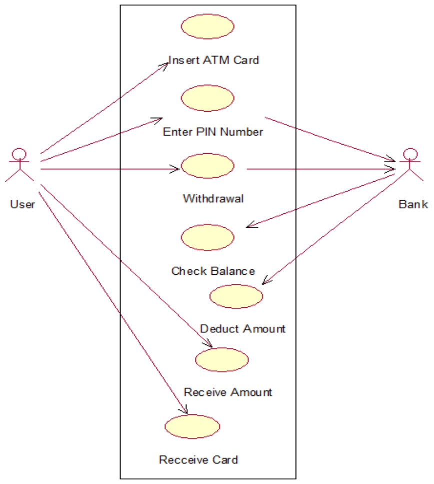

Fig.2. Use Case diagram for Money withdrawal from ATM.

By considering use cases, test cases can be generated very easily and can be executed automatically or manually. Here is an example use case diagram for money withdrawal from an ATM. The pictorial representation of ATM i.e., use case diagram is shown in Fig.2. Using this figure all test cases can derived.

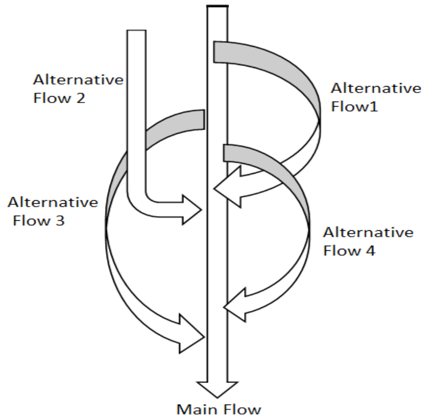

Fig.3. Shows Different flows of Use Case model for Money withdrawal module of ATM system.

Table 1. Different flows of Use Case model for Money withdrawal module of ATM

S. No.	Main Flows of Use Cases	Alternative Flows of Use Cases	No. of Alternative Flows for each use case
1.	***Insert card***	User Interface ,Buttons, Keypad, Backspace entries, Account Information, Pin-code Generation, Transaction charge, Deposit limit, Limit per transaction, Deposit limit per day, Withdrawal limit per transaction, Withdrawal limit per day, Account status, Card active, Card inactive, Expired, Replaced, Reported stolen, Suspicious activity, Improper card, Proper card, Wrong way, Upside down, Correctly, Select language, Language acceptance, Display language,	28
2.	***Enter PIN***	Receive PIN, Verify PIN, Correct PIN move, Wrong PIN, Retype PIN	05
3.	***Money Withdrawal***	from Checking account, from Savings account, Upper limit (+.01 and -.01), Lower limit (+.01 and -.01), Nothing, Correct amount (Yes, No), Re-enter, Check Amount	08
4.	***Check Balance***	----------	00
5.	***Deduct Amount***	----------	00
6.	***Receive amount***	----------	00
7.	***Eject Card***	Another transaction (yes, No), Get receipt (yes, No), Transaction charge (amount + acc. type)	03
Total no. of Alternate flows of a use case			= 44
Total No. of Main Flows of Use Case			= 07
Total No. of Main & Alternative Flows			= 51

Test cases result is either true or false, it depends on the expected result and actual result.

```
 If <expected results> equal to < Actual results>
        Then <test case result> should be <True>
else
            <test case result> should be <False>
```

Test case priority is given based on the severity of the Bug.

```
If <test case> result=True <Priority> should be <Low>
else
If<test case>result=False && not affecting other module
                          <Priority> should be <Medium>
else
If<test case>result=False && affecting other module
                         <Priority> should be <High>
```

C. Effort Estimation

The use case point method can be used to determine the software system test estimation, with this technique people can also forecast the size of software system before development. This is widely used estimation technique for object oriented software systems [5].

⇨ The Unadjusted Use Case Weight (UUCW) = (Total No. of Simple Use Case *1)+ (Total No. of Average Use Case *2)+ (Total No. of Complex Use Case *3) -- Eq.--2

⇨ The Unadjusted Actor Weight (UAW) = (Total No. of Simple Actors *1)+ (Total No. of Average Actors *2)+ (Total No. of Complex Actors *3) -- Eq.--3

⇨ The Unadjusted Use Case Point (UUCP) = UUCW+UAW -- Eq.--4

⇨ The Adjusted Use Case Point (AUCP) = UUCP * [0.65+(0.01*TEF)] -- Eq.--5

⇨ The Total Effort Through Use Case Point (EUP) = AUCP*PWE --------------------------------------- Eq.--6

This method will primarily take no. of use cases and no. of actors into consideration and it will estimate the effort in Man-Hours [11]. The existing equations [Eq..1 to Eq..6] are used for effort estimation. In equation 6, the term PWE considered for Plan, Write, and Execute test cases and it is use case dependent, it varies from system to system [12]. The term TEF is Total Environmental Factor, If TEF is not provided, tester can assume as 0.5 [11].

In the given case study “Money withdrawal from ATM” there are two actors namely User and Bank. User can be treated as a simple actor because he is following only GUI whereas Bank is having API / low-level interactions, hence it is treated as complex actor. And this system has seven use-cases at three different verities (simple, average and complex) based on the number of transactions. Every verity of U/C will have waiting factor shown in Table 2.

The effort can be estimated in *man-hours for performing regression testing.* The total effort estimation for ATM money withdrawal module is *obtained as 6.17 man-hours and this value is presented in table 2.*

IV. CONCLUSIONS & FUTURE ENHANCEMENT

In this paper, the model based regression testing approach is presented. Primarily use cases are considered for generating test cases for ATM system. This approach achieved to deriving test cases from behavioral models, maximized test coverage, early detection of requirements errors, automatic test case prioritization, automatic test suit reduction/ optimization and effort estimation.

For this case study 55 test cases are derived and these test cases are used in effort estimation which is represented in man-hours. This information is useful in carrying out the software maintenance. It provides low test execution cost which leads to low project maintenance. Here the model itself will regenerates the test cases for new functionalities.

Further research includes implementation of Model Based Testing (MBT) Techniques for test case optimization with an experimental setup.

Table 2 The effort estimation for the ATM system

Unadjusted Actor weights(UAW)				Unadjusted Use Case weights(UUCW)			
Actor Name	Actor Type	Factor	Weight	Use Case Name	Use Case Type	Factor	Total Factor
User	Simple	01	1*1=1	Check balance, Deduct amount Receive amount	Simple (Transactions <=3)	01	3*1=3
-------	Average	02	00	Enter PIN, Eject Card	Average (Transactions 4 -7)	02	2*2=4
Bank	Complex	02	1*3=3	Insert ATM card(UI, Acc. Info), Withdrawal	Complex (Transactions >7)	03	2*3=6
Total UAW			04	*Total UUCW*			13
Unadjusted Use Case Point (UUCP) = UUCW+UAW							17
Adjusted Use Case Point (AUCP) = UUCP *[0.65+(0.01*TEF)] AUCP =17*[0.65+(0.01*0.50)]							11.14
Total Effort through Use cases Pint (UPE) = AUCP* 0.5							***5.57***
Total Regression Testing Effort = verification of fixed bugs + UPE [20(minutes)+5.17(hours)]							***6.17***

Table 3. Test Cases for Withdrawal money form the ATM machine

TC ID	**TC Name**	**TC Description**	**Pre – condition**	**Input Fields**	**Expected Results**	**Actual Results**	**TC Result**	**TC Priority**
1.0.0	**user interface**	Check screens have proper format and text	ATM Should not be in out of order	-----	screens with proper format and text	screens displaying proper format and text	True	Low
1.1.0	***Buttons***	buttons correspond to proper items on screens	ATM Should have to have Touch screen or Buttons	Touch	Responds to finger touch	Responding to finger touch	True	Low
1.2.0	***Keypad***	keypad entries are properly displayed	Manual and Virtual Key pad should be available	------	keypad entries display properly	keypad entries are properly displaying	True	Low
1.3.0	***backspace entries***	can backspace to delete entries	There should be a delete option	Wrong Data Entry	We can delete entered data	We are able to delete wrong data	True	Low
2.0.0	***Account Information***	debit or credit card information	--------	Choose the option	Display the account types	Displaying Acc. Types	True	Low
2.1.0	***Pin-code Generation***	System should generate a PIN	Max. limit is 4- digits	------	User will get PIN	Getting PIN	True	Low
2.2.0	***transaction charge***	transaction charge per transaction	With draw amount	-------	Deduction from account	Trans. Amount Deducted from account	True	Low
2.3.0	***deposit limit limit per transaction***	deposit limit per transaction	Open Account	Choose deposit	Display trans. Limit	Displayed trans limit	True	Low
2.4.0	***deposit limit per day***	deposit limit per day	Open Account	Choose deposit	Display trans. Limit	Displayed trans limit	True	Low
2.5.0	***withdrawal limit per transaction***	withdrawal limit per transaction	Account should have money	Choose withdrawal	Display trans. Limit	Displayed trans limit	True	Low
2.5.0	***withdrawal limit per day***	withdrawal limit per day	Account should have money	Choose withdrawal	Display trans. Limit	Displayed trans limit	True	Low
2.5.0	***account status***	To know the account status	Open Account	Choose status option	Display Acc. Status	Displayed Acc. Status	True	Low
2.5.1	***card active***	Activation of new card for the first time	Receive the card from bank	Insert card	Card will activate	Card activated	True	Low
2.5.2	***card inactive***	Card inactivation	------------	Insert card	Card will activate	Card in-activated	False	Medium
2.5.2.1	***Expired***	Card gets Expired	Compare card date with current date	Insert card	System will display card expired	System will displayed card expired	True	Low
2.5.2.2	***Replaced***	Card is replace with new card	Lost/ Stolen the card	Insert card	System will display welcome message	System will displayed welcome message	True	Low
2.5.2.3	***reported stolen***	Card was stolen so receive the complaint	Lost Card	Choose the option	System will receive acceptance	System will received acceptance	True	Low
2.5.2.4	***suspicious activity***	Detecting the suspicious activity with transaction by card	Any Unknown activity	Any Wrong activity	---------------	Something has gone wrong	False	High
3.0.0	***insert card***	Enter the card in to ATM Machine	There should be a card acceptance path	Insert card	Card will go into the ATM Machine	Card inserted successfully	True	Low
3.1.0	***improper card***	The entered card is not ATM Card	There should be a card acceptance path	Insert card	Card will go into the ATM Machine	Unable to insert Card	True	Low
3.2.0	***proper card***	The entered card is an ATM Card	There should be a card acceptance path	Insert card	Card will go into the ATM Machine	Card inserted successfully	True	Low

3.2.1	***wrong way***	Correct card inserted in wrong direction	There should be a card acceptance path	Insert card	Card will go into the ATM Machine	Card inserted successfully	True	Low
3.2.2	***upside down***	Correct card inserted in upside direction	There should be a card acceptance path	Insert card	Card will go into the ATM Machine	Card inserted successfully	True	Low
3.2.3	***correctly***	To check whether the system responding for valid car or not	monitor should be there for display	------	ATM displays the Welcome	ATM displayed Welcome	True	Low
3.3.0	***Select language***	To check whether the system displaying language option or not	There should be a language option	Choose the language	Language will be changed	Language changed Successfully	True	Low
3.3.1	***Language acceptance***	To check whether the system accepting selected language or not	------	------	Chosen language is first option	Chosen language is first option only	True	Low
3.3.2	***Display language***	To check whether the system displaying all thing in selected language or not.	------	------	ATM Will display all language related options	ATM Will displayed all language related options	True	Low
4.0.0	***Enter PIN***	To check whether the system asking for pin or not	------	Enter pin	ATM Will accept PIN	ATM accepted PIN	True	Low
4.1.0	***receive PIN***	To check whether the system receiving pin or not	------	------	Entered Data will be accept	Entered Data is accepted	True	Low
4.2.0	***Verify PIN***	To check whether the system is verifying pin with card information and database or not	------	------	Expecting PIN is correct	Expecting PIN verification decision	True	Low
4.2.1	***Correct PIN move***	To check whether the system moving to next activity for correct pin moves	------	------	screen moves to next level	Screen moved to next screen	True	Low
4.2.2	***Wrong PIN***	To check whether the system is able to identify wrong pin and asking for reenter the pin or not	------	wrong Card & PIN entries	PIN and the card is wrong	PIN and the card are wrong	True	Low
4.2.3	***Retype PIN***	To check whether the system is retaining the card for more no of wrong pin entries or not	------	Re type correct PIN	PIN and the card is correct	PIN and the card are accepted	True	Low
5.0.0	***withdrawal***	To check whether the system is showing withdrawal options or not (like current, savings acc.)	------	Choose withdrawal option	Choose savings or current a/c	Chosen savings	True	Low
5.1.0	***from checking account***	To check whether the system accepting checking account option or not.	------	Choose withdrawal option	Choose savings or current a/c	Chosen savings	True	Low
5.2.0	***from savings account***	To check whether the system accepting savings account option or not.	------	Choose withdrawal option	Choose savings or current a/c	Chosen savings	True	Low
5.2.1	***upper limit***	To check whether the system sending warning message if user entered amount exceeds max limit or not.	------	Enter the amount	Account having sufficient funds	Account having sufficient funds	True	Low
5.2.2	***lower limit***	To check whether the system sending warning message if user entered amount exceeds lower limit or not.	------	Enter the amount	Account having sufficient funds	Account having sufficient funds	True	Low
5.2.3	***Fund limit***	To check whether the system having sufficient funds or not, if not warning message is sending or not	------	Enter the amount	Account having sufficient funds	Account having sufficient funds	True	Low
6.0.0	***correct amount***	To check whether the entered amount is correct or not.	Have a look on entered amount	------	Get options Yes and No	Gat options Yes and No	True	Low
6.1.0	***Yes***	Proceed with withdrawal	Account having sufficient funds	Press YES	Counting machine (amount) starts	Counting amount started	True	Low
6.2.0	***No***	Cancel withdrawal option	------	Press No	Eject card	Ejected Card	True	Low
6.2.1	***Re-enter***	Enter the amount once again	Insufficient	Enter the	Transaction	Proceeding	True	Low

			fund/ limit exceeds …etc.	amount	proceeds	with Transaction		
7.0.0	***Check Balance***	To check the account status	Account Existence	Enter Account Number	Sufficient amount is available in the Account	Amount is available	True	Low
8.0.0	***Deduct Amount***	To deduct the requested amount from the account.	Sufficient amount in the account	------	Amount will debit from Account	Amount was deducted from account	True	Low
9.0.0	***Receive amount***	To check whether the user is able to receive amount or not	------	------	ATM will send money	Received amount	True	Low
10.0.0	***another transaction***	To check whether the system is asking for another transaction or not	Complete previous transaction	Choose an option	System will display Yes or No options	System will displayed Yes or No options	True	Low
10.1.0	***yes***	To check whether the system Proceeding with other transaction or not	Account having sufficient funds	Press YES	Counting machine (amount) starts	Counting amount started	True	Low
10.2.0	***No***	To check whether the system proceeding with no transaction or not.	------	Press No	Eject card	Ejected Card	True	Low
10.3.0	***Get receipt***	To check whether the system is Asking the user to get the receipt on his/her account or not.	Transactions completed	Choose an option	System will display Yes or No options	System will displayed Yes or No options	True	Low
10.4.0	***yes***	To check whether the system receiving get the receipt option or not.	Printer should ready	Choose Yes	ATM will send printed paper	Received Printed paper	True	Low
10.5.0	***No***	To check whether the system receiving print receipt is not required or not	------	Choose No			True	Low
11.0.0	***transaction charge (amount + acc. type)***	To check whether the system displaying transaction charges or not.	------	Choose trans. Charges	Displays the list	Displayed the list	True	Low
12.0.0	***Eject Card***	To check whether the ATM machine card back or not.	------	Click on eject option	Card will be ejected by ATM	Received Card from ATM machine	True	Low
3.3.0	***Select language***	To check whether the ATM system is displaying different user continent languages or not	There should be a language option	Choose the language	Language will be changed	Language changed Successfully	True	Low

*TC- Test- Case.

REFERENCES

[1] Prof. A. Ananda Rao et al "An Approach to Cost Effective Regression Testing in Black-Box Testing Environment", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, ISSN (Online): 1694-0814,May 2011.

[2] Susanne Rösch, Sebastian Ulewicz, Julien Provost, Birgit Vogel-Heuser "Review of Model-Based Testing Approaches in Production Automation and Adjacent Domains—Current Challenges and Research Gaps", Journal of Software Engineering and Applications, , 2015, 8, 499-519.

[3] L. Erlikh, "Leveraging legacy system dollars for e-business", **IEEE**, IT Professional, Volume: 2, Issue: 3, May/Jun 2000.

[4] Jim Heumann, "Generating test cases from Use Cases" Rational edge, Copyright Rational Software 2001 | Privacy/Legal Information.

[5] Korel, B., Tahat, L. and Vaysburg, B. (2002) Model Based Regression Test Reduction Using Dependence Analysis. 2002 International Conference on Software Maintenance, Montreal, 3-6 October 2002, 214-223.

[6] Yanping hen, "Regression Test Suit Reduction Using Extended DependencyAnalysis"Pages62-69 ACM NewYork, NY, USA ©2007 table of contents ISBN:978-1-59593-724.

[7] S. Selvakumar et al ,"Extended Finite State Machine Model-Based Regression Test Suite Reduction Using Dynamic Interaction Patterns" Springer-Verlag Berlin Heidelberg-2010 10.1007/978-3-642-12214-9_82

[8] Arilo C. Dias Neto1 Rajesh Subramanyan2 Marlon Vieira2 Guilherme H. Travassos1, "A Survey on Model-based Testing Approaches: A Systematic Review", ACM November 5, 2007.

[9] Julien Courbe, "An ounce of prevention: Why financial institutions need automated testing," PwC, November 2014, www.pwc.com/fsi

[10] M. Gopichand, A.AnandaRao "Five Layered model for identification of software performance requirements" International Journal of Software Engineering and Applications(IJSEA),Volume 3,No.5, pp 47-61,September 2012.

[11] Gregory Tassey, "The Economic Impacts of Inadequate Infrastructure for Software Testing", Health, Social, and Economics Research Research Triangle Park, NC 27709, May 2002.

[12] Suresh Nageswaran, "Test Effort Estimation Using Use Case Points", Copyright(c) 2001, Cognizant Technology Solutions, Quality Week 2001, San Francisco, California, USA, June 2001.