Volume 2, No. 4, July-August 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Simulator to Calculate Test Efforts by Using Reusability of Code

Princy Garg
dept. of Computer Science & Application
Kaithal, India
garg.princy9@gmail.com

Abstract: In computer science reusability is segment of source code that can be used again to add new functionalities with slight or no modification in a software. Software Testing is the process of executing a program or system with the intent of finding error. So,Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results vIn This paper we use the concept of estimation of software testing efforts based on test case point analysis [TPA] as a fundamental project estimation measure. We also discuss The concept of function point analysis[FPA]technique.In this paper, simulator will calculate the testing efforts with and without reusability. To achieve all these goal ,we implement the simulator in high level language. Our results show that the software testing efforts becomes less in case of using TPA technique with reusability as compared to TPA technique without reusing the software.

Keywords: FPA, TPA, White Box, Black Box

I. INTRODUCTION

Development of software has initiated the new role of software testing. At the beginning of software products development the majority of the testing was performed by the developer himself due to the simplicity of the product. As the complexity of software products has increased, the role of all parts in the software development process has been modified including with the role and importance of the testing process[5]. Besides code, other technical artifacts (design, requirements, testing) and management artifacts (such as plans and processes) have also been reused. Reuse in software engineering resulted in reduction of development and maintenance time, effort, and in improved software quality.

II. TESTING TECHNIQUES

There are basically two techniques of testing

A. White Box testing
B. Black Box testing

Black Box testing::It is also known as *functional testing*. In this testing technique, the internal workings of the item being tested are not known by the tester. The tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

White Box Testing: It is also known as *glass box, structural testing* .In this testing technique, explicit knowledge of the internal workings of the item being tested are used to select the test data. white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do.. White box testing does not account for errors caused by omission, and all visible code must also be readable.

III. TEST EFFORTS SCHEMES

There are basically two Types of Test Effort Schemes

A. Function Point Analysis (**FPA**): white-box test activities are included in the size calculation produced by FPA.
B. Test Point Analysis (**TPA**): TPA is one such method which can be applied for estimating test effort in black-box testing

IV. BASIC CONCEPT OF FPA TECHNIQUE

The FPA technique estimates the development function points, which also include white-box testing effort. FPA is a method for measuring the size of the software on the customer's point of view and describes a unit of work suitable for measuring the size of business application software. FPA can be used to measure productivity across various tools and environments. A basic knowledge of the FPA method is necessary to understand test point and maintenance analysis. The white box testing includes unit testing and integration testing. While white-box test activities are included in the size calculation produced by FPA, the black box testing activities are not included in size computation of FPA.

V. BASIC CONCEPT OF TPA TECHNIQUE

TPA is one such method which can be applied for estimating test effort in black-box testing. The goal of this technique is to outline all major factors that affect testing projects and to ultimately do accurate test effort estimation. If one has a predetermined estimate of test hours As per TPA method, there are two kinds of test points-Dynamic and Static. As FPA is doing white box testing only, we need the TPA model to find the black box testing. The FP count we use to calculate the TPA is estimated earlier in the FPA technique. As per the FPA technique, there are two sets of elementary processes-transaction function points (data in motion), data function points (data in rest).TPA is one such

method which can be applied for estimating test effort in black box testing. It is a 6-step approach to test estimation and planning.[2].

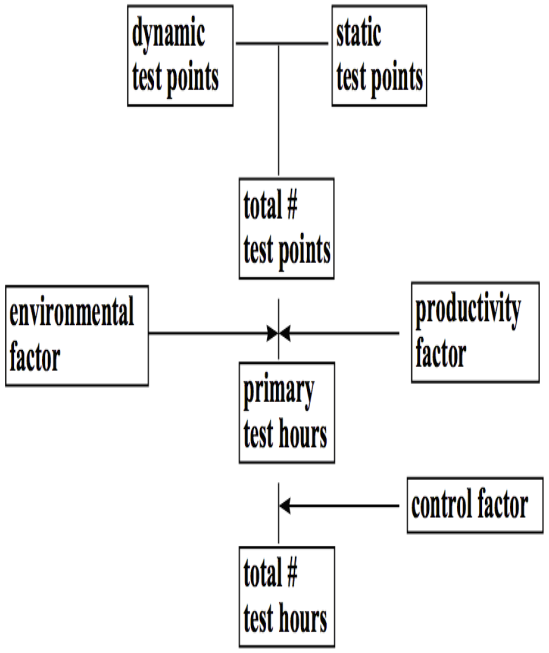

Figure 1-TPA Technique

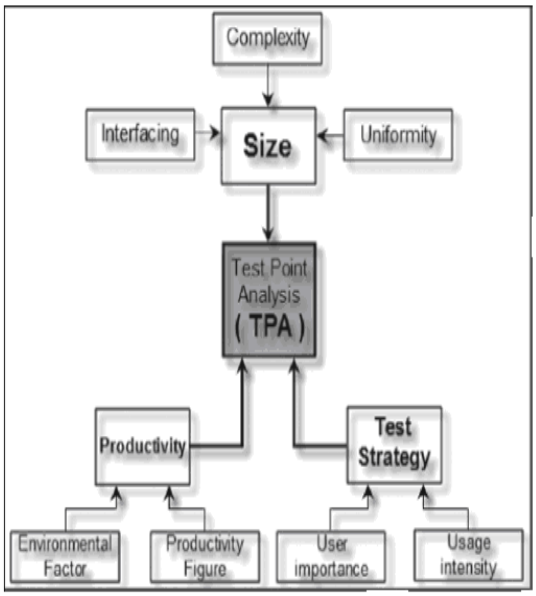

Figure 2- Parametres of TPA Technique[1].

A. Tpa Technique For Estimating Efforts:

a. Computing Dynamic Test Points (Tps):

Dynamic test points are related to individual function and are based on FPA transaction function points. Dynamic test points are computed by summing the product of Transaction Function points (FP_t), Dependency Factor (D_f), and Dynamic Quality Characteristics (Q_d) for individual function points.

b. Dependency factor (D_f):

A rating is assigned for the individual functions points. A useful heuristics is to have 25% functions in low, 50% in medium and 25% in high category.

i. User Importance of the functions: Rating—3-low, 6-medium, 12-high.[1].
ii. Intensity of the functions: Rating— Usage 2-low, 4-medium, 12-high.
iii. Interfacing with other functions: Rating—2-low, 4-medium, 8-high.
iv. Complexity of function: Rating—3-low, 6-medium, 12-high.

These ratings are added and divided by 20 (sum of medium rating) to arrive at weighted rating, and uniformity factor could be 0.6 or 1. The uniformity is taken at 0.6 in case of second occurrence of unique function, where test specs can be reused else, uniformity factor is taken at 1.
Dependency factor is calculated by multiplying weighted rating with uniformity factor.

c. Dynamic Quality Characteristics (Q_d):

This calculation is based on rating and weighing factor for the variables-suitability, security, usability, efficiency. Weighing factors for these four variables are 0.75, 0.05, 0.10, and 0.10 respectively. For each of these variables the rating is (0-not important, 3-relatively unimportant, 4-medium importance, 5- very important, 6- extremely important).

Total dynamic test points equal sum of FP_t* D_f*Q_d for individual functions.

d. Computing Static Test Points:

Test points are related to overall FP of the system and static quality characteristics of the system. Overall FP of the system is assumed at minimum 500(in case it is below 500)recommends functionality, usability, reliability, efficiency, portability and maintainability as quality characteristics and several sub- characteristics within these as desirable. For each quality characteristics statistically tested, a value of 16 is added to Qi.

B. Total Test Points:

Total test points are equal to sum of Dynamic and Static test points.
TP = (Sum of FP_t* D_f *Q_d for individual functions) + (Total FP* Qi/500)[1].

C. Productivity Factor (P) :

Indicates tests hours required per test point. It ranges from 0.7(if test team is highly skilled) to 2(if test team has insufficient skills) hours per test point. Productivity factor requires historical data of the projects and it can very from one organization to another organization. So, this factor can be called organization dependent factor.

D. Environmental Factor (E):

The number of test hours required for each test point is not only influenced by productivity factor but also by the environmental factor.[1] The following environmental factor might affect the testing effort: test tools, development testing, test basis, test ware, development environment, and test environment. Environmental factor is calculated by adding the rating on all the above environmental factors and divided by value 21(the sum of nominal ratings).

E. Primary Test Hours:

The number of primary test hours is obtained by multiplying the number of test points by productivity factor (P) and environment factor (E).

Primary test hours = Test points (TP)*P*E

F. Planning and Control Allowance :

The standard value of this is 10%.this value may be increased or decreased depending on two factors

a. Team Size:

The bigger the team, the more effort it will take to manage the project. The ratings for this value are: 3- if team consists of up to 4 persons, 6- if team consists of up to 5 and 10 persons, 12- if team consists of more than 10 persons.

b. Management Tools:

More the number of tools used to automate management and planning less are the amount of effort required. The ratings for this value are: 2-both an automated time registration system and automated defect tracking system are available, 4- either an automated time registration system or automated defect tracking system is available, 8- no automated systems are available.

Planning and control allowance =Team size factor +Management tools factor

G. Total Test Hours:

The total number of test hours is obtained by adding primary test hours and the planning and control allowance.
Total test hours= Primary test hours+ Planning and control allowance In the many approaches to test effort estimation, the use of stubs and drivers may be one. This could become a robust method of estimation over a period of time. The estimation technique is not claimed to be rigorous, but the approach offers practical advantages over techniques currently in use.

VI. PROPOSED ALGORITHM OF TPA TECHNIQUE

A. Dynamic test point: $D_t = FP_f * D_f * Q_d$
Where, FP_f =Transaction FP
D_f = Dependency Factor = Weighted rating on Importance to user, usage intensity, interfacing of functions, complexity of functions.
Rating on user importance(U_p):
U_p= 3*20%+6*60%+12*20%
=0.6+3.6+2.4 =6.6
Rating on usage intensity(U_i):
U_i=2*10%+4*70%+12*20%
= 0.2+2.8+2.4 =5.4
Rating on interfacing (I):
I= 2*50%+8*50% =5
Rating on Complexity (C):
C= 6(nominal complexity)
$D_f = (U_p + U_i + I + C)/ 20* U$
U =Uniformity Factor= 60%*1+40%*0.6
= 0.6+ 0.24 =0.84
$D_f = (U_p + U_i + I + C)/ 20* U$
Q_d = Dynamic quality characteristics = weighted score on following 4 quality characteristics:
Suitability
(weight=0.75, medium importance—rate =4)
Security
(weight=0.05, extremely importance—rate =6)
Usability
(weight=0.10, highly importance—rate =5)
Efficiency
(weight=0.10, medium importance—rate=4)
weighted score = (0.75*4+0.05*6+0.10*5+0.10*4) Q_d = 3+0.3+0.5+0.4= 4.2
Hence, $D_t = FP_t * D_f * Q_d$

B. Static test point
S_t=total FP * Q_i/500
Total FP = Data FP+ Transaction FP
S_t=total FP * Q_i/500

C. Total test point
TP= D_t+ S_t

D. Productivity Factor (PF) = 1.4 tests hours per test point
Rating on test tools=1
Rating on development testing =4
Rating on test basis = 6
Rating on development environment =2
Rating on test environment =2
Rating on test ware =2

E. Environmental Factor
EF =1+4+6+2+2+2/21 =0.81

F. Primary test hours
P=TP* PF *EF
Planning control allowance =6%+2% = 8%

G. Total test hours = P+ 8% of P

H. Experimental Results:

All these calculations are done by simulator.
Output:

a. Calculated Test Efforts Without Reuse:

value of Transaction count 696.000000
value of Data Count 480.000000
Dt = Fp * Df * Qd
Dynamic test point is 2823.811035
Static Test point is 57.088001
Toatal Test Point is(tp) = Dynamic test point(dt) +Static test point(st)
Total Test Point is 2880.898926
Primary Test Hours(P) = tp*pf*ef
Primary Test Hours(P) is 3266.939453
Total Test Hours = Primary test hours(p) + 8% of Primary test hours(p)
Total Test Hours is 3528.294678

b. Calculated Test Efforts with Reuse:

value of Transaction count 696.000000
value of Data Count 480.000000
Dt = Fp * Df * Qd
Dynamic test point is 2734.723145
Static Test point is 57.088001
Toatal Test Point is(tp) = Dynamic test point(dt) +Static test point(st)
Total Test Point is 2791.811035
Primary Test Hours(P) = tp*pf*ef
Primary Test Hours(P) is 3165.913574
Total Test Hours = Primary test hours(p) + 8% of Primary test hours(p)
Total Test Hours is 3419.186768

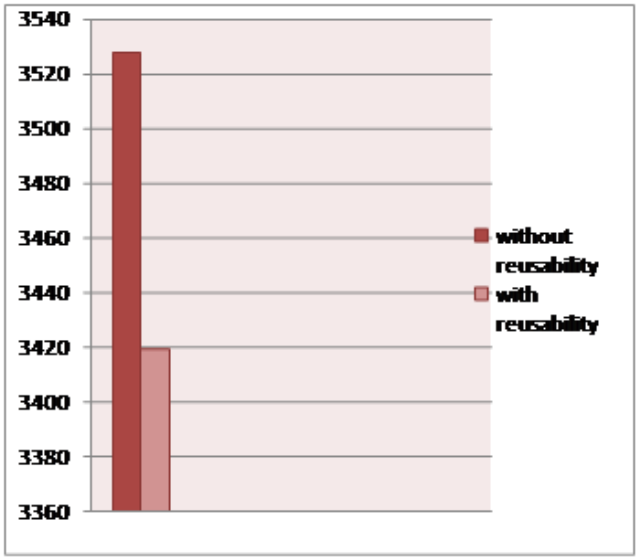

Figure: 3

VII. CONCLUSION

Testing effort is the number of hours that is required for the testing process of software that is being developed. Effective test effort estimation is one of the most challenging and important activity in software testing. There are many popular models for test effort estimation in vogue today. Ineffective test effort estimation leads to schedule and cost overruns. This is due to lack of understanding of development process and constraints faced in the process. But we believe that our approach overcomes all these limitations. In this paper we find out that how effectively we can minimize the test effort for a project. We used the TPA method for our proposed work. Test Case Point Analysis is a tool to estimate the effort required to test a software project, based on the number of use cases and the other features of object-orientation used in software development. Testing is an important activity that ensures the quality of the software.

Here is an area where further work is necessary, obviously. However, there are methods that make it possible to estimate effort required for executing Testing projects. Test Points are slowly emerging for sizing Software Testing projects. In the many approaches to test effort estimation, the use of stubs and drivers may be one. Drivers and stubs can be reused so the constant changes that occur during the development cycle can be retested frequently without writing large amounts of additional test code. In effect, this reduces the cost of writing the drivers and stubs on a per-use basis and the cost of retesting is better controlled. We are using this approach as the stubs and drivers are reused then the less coding is to be done, and less will be the test effort for test the code. Either it takes more code writing for stubs or drivers but the reusability of these minimizes the overall coding and the test effort also. So using the stubs and drivers approach is more beneficial than without them. This could become a robust method of estimation over a period of time. It leads to accurate estimation of test effort by this estimation we can easily calculate the test effort for the each phases of a testing life cycle. We can apply this estimation to find the estimated test plan and it is also a very powerful method to generate realistic test cases.

VIII. REFRENCES

[1] Raghuvirkamath, " TPA – Test Point Analysis – A method of Test Estimation". http://raghuvirkamath.wordpress.com/2010/06/08/tpa-test-point-analysis-a-method-of-test-estimation/

[2] Rajiv Chopra, (2009), Second Edition ,"Software Testing-Test Point Analysis" (TPA), (pp 309-321), S.K Kataria & Sons, New Delhi

[3] Renu Rajani, Pradeep Oak, (2004) "Software Testing-Software Test Effort Estimation Techniques",Tata McGraw Hill, New Delhi

[4] William E,(1999), "Effective Method For Software Testing", Perry, (pp-177-205), Wiley

[5] Nick Jenkins,"A Software Testing Primer",An Introduction to Software Testing,2008.

[6] Suresh Nageswaran, "Test Effort Estimation Using Use Case Points", (*Cognizant Technology Solutions)*. Quality Week 2001, San Francisco, California, USA. http://www.testexpert.com.br/files/Test%20Effort%20Estimation%20Using%20Use%20Case%20Points.pdf

[7] Sudip Misra, "An Empirical Framework For Choosing An Effective Testing Technique For Software Test Process Management", Journal of Information Technology Management ISSN #1042-1319,A Publication of the Association of Management

[8] Silverpath technologies inc, "Increasing Test Effort Estimate Effectiveness", trevor.atkins@silverpath.com, may 29,2008

[9] Willie L. Brown, **"**Function Point Analysis", Software Project Management, SE 510- Spring 2009

[10] Drs Erik P.W.M. Van Veenendaal CISA, Ton Dekkers, "Test Point Analysis: A Method for Test Estimation",1999.

[11] Pressman, "An approach to Software Engineering" 6th Edition, 1991.