A Novel Tool for Reducing Time and Cost at Software Test Estimation: An Use Cases and Functions Based Approach

Shaiful Islam[1], Bishwajit B. Pathik[2], Manzur H. Khan[3], Md. Mamun Habib[4]
[1, 3] Department of Computer Science, American International University-Bangladesh, Dhaka, Bangladesh
[2]Department of Electrical and Electronic Engineering, American International University-Bangladesh, Dhaka, Bangladesh
[4]School of Quantitative Sciences, Universiti Utara Malaysia (UUM), Malaysia
[1]jshaiful@gmail.com, [2]bishwajit.b.pathik@gmail.com, [3]manzur@aiub.com, [4]md.mamun@uum.edu

Abstract – **This paper represents a software test estimation tool, which is clearly defined to understand, that provides the estimated time as well as the cost of any software test project. There are different estimation tools for software development process [8] and those are well recognized. But there are lacks of standard tools for estimation of Software Test phase. Therefore, the researchers designed and applied this novel tool on 5 case projects, namely Climate Resilient Ecosystems and Livelihood (CREL), Management Accounts Consolidation System for Line Director (LDMACS), PBI Works Management System Software, PBI Mobile Apps and Human Resources Management System (HRMS), in order to determine the time and cost of those projects. Since few software test estimation tools are available in the market, however, this paper would be fruitful for the software industry to explore the time and cost of any real software test project.**

Keywords – **Test Estimation, Use Cases, Functions, Cost Estimation, Time Estimation.**

I. INTRODUCTION

Estimation is always important and it plays a vital role with resource, schedule and budget. It is desired to use an appropriate and accurate estimation technique for the software test projects to show the right value of the system and the right payment for the developers. Software Development Life Cycle (SDLC) consists of requirements analysis, design, coding, testing, deployment and maintenance. Though testing phase comes later but work starts from the beginning by test planning and test case preparation. If a system analyst could use an effective and efficient software test estimation technique and implementing the estimation in real project then the project cost will be calculated easily. Moreover, project will cost less in long run if better design is used for system development and maintenance. An accurate estimation technique would be suggested by the authors in this paper.

This paper illustrates few topics sequentially in the following sections. Related study and Technique for Software Test Estimation have been presented in the next sections which consist of some fundamental theory of test estimation practices, formula and their calculation methods, advantages, disadvantages and other features. Consequently, analysis of software test estimations as well as findings, results and conclusion were depicted in the consecutive sections.

II. RELATED STUDY

Software project failures are an important subject in the last decade. Software projects usually do not fail during the implementation and most project fails during the planning and estimation steps. Despite going to over time and cost, approximately 30% to 40% of the software projects are completed and the others fail [10]. According to the the Standish group's CHAOS reports, failure rate is 70% for the software projects. Also the cost overrun has been indicated 189% in 1994 CHAOS report [10]. In addition, Ref. [10] indicates that the CHAOS report may be corrupted. Nevertheless the mentioned statistics show the deep crisis related to the future of the software projects [10].

During last decade several studies have been conducted in order to find reasons of software projects failure [11]. An intensive search has been performed between 2100 internet sites and found 5000 reasons for the software project failures. Among the found reasons, insufficient requirements engineering, poor planning the project, suddenly decisions at the early stages of the project and inaccurate estimations were the most important reasons. The other researches regarding the reason of project fails show that inaccurate estimation is the root factor of failure in the most software project fails [10]. Despite the indicated statistics may be pessimistic, inaccurate estimation is a real problem in the software production's world which should be solved. To resolve the above problems we need to apply efficient techniques and reliable models. The conditions of the software projects are not stable and the state is continuously changing. Therefore, different methods should be presented for estimation so that each method is appropriate for a special project.

However, as it is really hard to find any reliable software test estimation tool, therefore, the intention of the researchers in this article is to present a tool to resolve the present bottleneck of estimation of software testing that will lead to success in terms of time and costing.

III. TECHNIQUE FOR SOFTWARE TEST ESTIMATION [1], [3], [4], [5]

This paper represents software test estimation techniques step wise with the method. This would cover the formulas, calculations, ratings, weight factor and the explanation of each area. It could be express as:

Total Hours = PT * PC, **Total MM** = (Total Hours / 7) / 22
Project Cost = Total Hours * Per Hour Cost + Overhead Cost

(a) Function-dependent (D_f) [1]

$$D_f = ((U_e + U_y + I + C) /16) * U$$

Where, D_f = weighting factor for the function-dependent factors, U_e = user-importance, U_y = usage-intensity, I = interfacing, C = complexity and U = uniformity

User Importance (U_e): Expert judgment techniques involve consulting with software. A useful rule of thumb is that about 25% of functions should be placed in the "high" category, 50 in the "normal" category and 25% in the "low" category.

Usage Intensity (U_y): The usage intensity has been defined as the frequency with which a certain function is processed by the users and the size of the user group that uses the function. As with user-importance the usage-intensity is being determined at a user-function level.

Interfacing (I): Interfacing is an expression of the extent to which a modification in a given function affects other parts of the system. The degree of interfacing is determined by ascertaining first the logical data sets (LDSs) which the function in question can modify, then the other functions which access these LDSs.

An interface rating is assigned to the function by reference to a table (Table 4) in which the numbers of LDSs affected by the function are ranged vertically and the numbers of other functions accessing the LDSs are ranged horizontally. When working out the number of "other functions" affected, a given function may be counted several times over if it accesses several LDSs, all of which are maintained by the function for which the interweave calculation is being made. If a function does not modify any LDSs, it is given a low interface rating. A CRUD table (Table 3) is very useful for determining the degree of interfacing.

Complexity (C): Complexity of functions depends on conditions in any functions. Depending on conditions rating and weight has set for the calculation.

TABLE 1
TABLE FOR RATING AND WEIGHT OF CRUD

LDS\Functions	*1*	*2-5*	*>5*
1	L	L	A
2-5	L	A	H
>5	A	H	H

L: Low interfacing, A: Average interfacing, H: High interfacing

(b) Dynamic Quality Characteristics (TP_f)

$$TP_f = FP_f * D_f * Q_d$$

Where, TP_f = number of test points assigned to the function, FP_f = number of function points assigned to the function, D_f = weighting factor for the function-dependent factors, Q_d = weighting factor for the dynamic quality characteristics = per characteristic 0.02/4*Weighting Factor, then added together (explicitly measurable quality characteristics)

In Test Point Analysis (TPA), four dynamic, explicitly measurable quality characteristics are recognized. [1], [3-4].

Uniformity (U): Under the following circumstances 60% of the test points assigned to the function under analysis count towards the system total. A uniformity factor of 0.6 is assigned in cases of the kinds described above; otherwise 1 is allotted.

(c) Total number of Test Points

$$TP = \sum TP_f + (FP * Q_i) / 500$$

Where, TP = total number of test points assigned to the system as a whole, $\sum TP_f$ = sum of the test points assigned to the individual functions (dynamic test points), FP = total number of function points assigned to the system as a whole (minimum value 500), Q_i = weighting factor for the indirectly measurable quality characteristics.

TABLE 2
TABLE FOR FUNCTION DEPENDENT AND DYNAMIC QUALITY CHARACTERISTICS

D_f & TP_f	*Rating*	*Factors*	*Weight*
User Importance	3	Low: the importance of the function relative to the other functions is low	0.5
	6	Normal: the importance of the function relative to the other functions is normal	0.75
	12	High: the importance of the function relative to the other functions is high	1
Usage Intensity	2	Low: the function is only used a few times per day or per week	0.1
	4	Normal: the function is being used a great many times per day	0.2
	12	High: the function is used continuously throughout the day	0.3
Interfacing	2	The degree of interfacing associated with the function is low	0.5
	4	The degree of interfacing associated with the function is normal	0.75
	8	The degree of interfacing associated with the function is high	1
Complexity	3	The function contains no more than five conditions	0.5
	6	The function contains between six and eleven conditions	0.75
	12	The function contains more than eleven conditions	1
Dynamic Quality Characteristics	3	Suitability: Quality requirements are not important and are therefore disregarded for test purposes.	0.05
	4	Security: Quality requirements are relatively unimportant but do need to be taken into consideration for test purposes.	0.10
	5	Usability: Quality requirements are of normal importance. (Appropriate where the information system relates to a support process.)	0.25
	6	Efficiency: Quality requirements are very important. (Appropriate where the information system relates to a primary process.)	0.50

(d) Primary Test Hours

$$PT = TP * P * E$$

Where, PT = total number of primary test hours, TP = total number of test points assigned to the system as a whole, P = productivity factor, E = environmental factor

$$\textit{Total test hours} = PT * PC$$

Where, PT= Primary test hours formula, PC= Total number of test hours.

(e) Static Test Points

One has to determine whether the static measurable quality characteristics are relevant for test purposes. A static test can be carried out using a checklist. In principle all ISO 9126 quality characteristics [4] can be tested using a checklist. E.g. security can therefore be measured dynamically, using a semantic test, and/or statically, by evaluating security measures with support of a checklist.

Method of Calculation (Q_i): If a quality characteristic is tested by means of a checklist (static test), the factor Q_i get the value sixteen. For each subsequent quality characteristic to be included in the static test, another sixteen is added to the Q_i factor rating.

Primary Test Hours (PT): The formula gives the total number of test points assigned to the system as a whole.

This total number of test points is a measure of the volume of the primary test activities. The primary test hour count is the number of hours required for carrying out the test activities involved in the test life cycle phases Preparation, Specification, Execution and Completion.

Productivity Factor (P): The productivity factor indicates the number of test hours required per test point. The productivity factor is a measure of the experience, knowledge and skill of the test team. It can be calculated by analyzing completed test projects. The higher the productivity factor, the greater the number of test hours required. In practice, the productivity factor has shown to have a value between 0.7 and 2.0.

Environmental Factor (E): The number of test hours required for each test point is also influenced by the environmental factor. Different environmental variables are defined for calculation of the environmental factor.

(i) Test Tools: The test tools variable reflects the extent to which testing is automated, or the extent to which automatic tools are used for testing. For the purpose of calculating this variable, the term "test tools" covers tools that are used for the primary test activities.

(ii) Development Testing: The development testing variable reflects the quality of earlier testing. If the estimate under preparation is for an acceptance test, the earlier testing will have been system testing; if the estimate is for a system test, the earlier testing will have been white-box testing. The quality of such development testing influences the amount of functionality that may be require less thorough testing with less coverage and the duration of the test activities.

(iii) Test basis: The test basis variable reflects the quality of the (system) documentation upon which the test under consideration is to be based. The quality of the test basis influences the amount of time required for the Preparation and Specification phases.

(iv)Development Environment: The development environment variable reflects the nature of environment within which the information system was realized. Here, the degree to which the development environment will have prevented errors and inappropriate working methods is of particular importance. If errors of given type cannot be made, it is not necessary to test for them.

(v) Test Environment: The test environment variable reflects the extent to which the test infrastructure in which the testing is to take place has previously been tried out. In a well-tried test infrastructure, fewer problems and delays are likely during the execution phase.

(vi) Testware: The testware variable reflects the extent to which the tests can be conducted using existing testware. The availability of usable testware mainly influences the time required for the Specification phase.

(vii) Method of Calculating Environment Factor: The environmental factor (E) is calculated by adding together the ratings for the various environmental variables (test tools, development testing, test basis, development environment, test environment and testware), then dividing the sum by twenty-one (sum of nominal ratings).

TABLE 3
TABLE FOR UNIFORMITY

	Sl. No.	*Cases*
Uniformity	1	In the case of a second occurrence of a virtually unique function: in such cases, the test specifications can be largely reused
	2	In the case of a clone function: the test specifications can be reused for cloned functions
	3	In the case of a dummy function (provided that reusable test specifications have already been drawn up for the dummy).
Tools	1	Testing involves the use of a query language such as SQL; a record and playback tool is also being used.
	2	Testing involves the use of a query language such as SQL, but no record and playback tool is being used.
	4	No test tools are available.
Development Testing	2	A development testing plan is available and the test team is familiar with the actual test cases and test results.
	4	A development testing plan is available.
	8	No development testing plan is available.
Test Basis	3	During the system development documentation standards are being used and a template, in addition the inspections are organized
	6	During the system development documentation standards are being used and a template.
	12	The system documentation was not developed using a specific standards and a template.
Development Environment	2	The system was developed using a 4 GL (Generation Language) programming language with an integrated DBMS containing numerous constraints.
	4	The system was developed using a 4 GL programming language, possibly in combination with a 3 GL language.
	8	The system was developed using only a 3 GL programming language such as COBOL, PASCAL or RPG.
Test Environment	1	Environment has been used for testing several times in past.
	2	The test is to be conducted in a newly equipped environment similar to other well-used environments within organization.
	4	The test is to be conducted in a newly equipped environment which may be considered experimental within organization.
Testware	1	A usable general initial data set (tables, etc.) and specified test cases are available for the test.
	2	A usable general initial data set (tables, etc.) is available for the test.
	4	No usable testware is available.
Team Size	3	The team consists of no more than four people.
	6	The team consists of between five and ten people.
	12	The team consists of more than ten people.
Management Tools	2	Both an automated time registration system and an automated defect tracking system (including CM: Configuration Management) are available.
	4	Either an automated time registration system or an automated defect tracking system (including CM) is available.
	8	No automated (management) systems are available.

Total Number of Test Hours (PC): The number of primary test hour and the planning and control allowance together give the total number of test hour. The standard (nominal) allowance is 10 per cent. However, the allowance may be increased or decreased, in line with two factors, namely, Team size and Management tools.

- *Team Size:* The team size factor reflects the number of people making up the team (including the test manager and, where appropriate, the test controller).
- *Management Tools:* The planning and control tools variable reflects the extent to which automated resources are to be used for planning and control.

The planning and management percentage is obtained by adding together the ratings for the two influential factors (team size and planning and control tools). The allowance in hours is calculated by multiplying the primary test hour count by this percentage. Addition of the planning and control allowance to the number of primary test hours gives the total number of test hours.

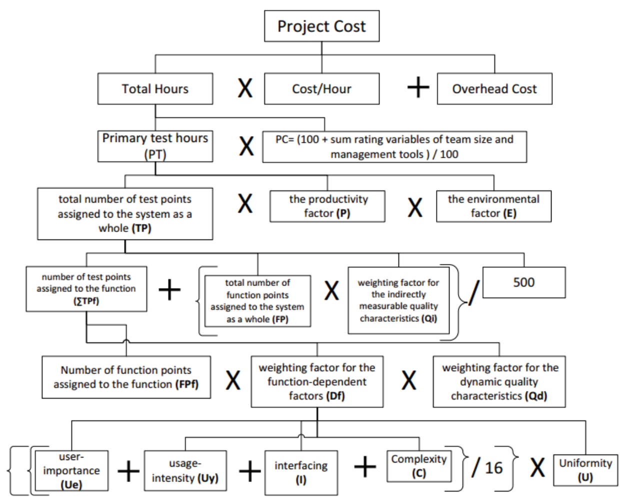
Project Cost
Total Hours
X
Cost/Hour
+
Overhead Cost
Primary test hours (PT)
X
PC= (100 + sum rating variables of team size and management tools) / 100
total number of test points assigned to the system as a whole (TP)
X
the productivity factor (P)
X
the environmental factor (E)
number of test points assigned to the function (∑TPf)
+
total number of function points assigned to the system as a whole (FP)
X
weighting factor for the indirectly measurable quality characteristics (Qi)
/
500
Number of function points assigned to the function (FPf)
X
weighting factor for the function-dependent factors (Df)
X
weighting factor for the dynamic quality characteristics (Qd)
user-importance (Ue)
+
usage-intensity (Uy)
+
interfacing (I)
+
Complexity (C)
/ 16
X
Uniformity (U)

FIGURE 1 Diagram of the Estimation Methodology

Method of Calculation: The planning and management percentage is obtained by adding together the ratings for the two influential factors (team size and planning and control tools). The allowance in hours is calculated by multiplying the primary test hour count by this percentage. Addition of the planning and control allowance to the number of primary test hours gives the total number of test hours.

Breakdown between Phases: The result of a TPA is an estimate for the complete test process, excluding formulation of the test plan. If a structured testing approach [6], [7] is used, the test process is divided into five life cycle phases; many clients will want to see estimates for the individual phases, as well as for the complete test process. The estimate for the Planning and Control phase will normally be the same as the planning and control allowance, i.e. the primary test hour count multiplied by the planning and control percentage. The primary test hours are then divided between the Preparation, Specification, Execution and Completion phases. The breakdown between phases varies from one organization to another or even from one organizational unit to another. Experts of TPA technique suggest the following percentages: preparation (10%), specification (40%), execution (45%) and completion (5%).

Total Hours = [{{∑(FPf * (((Ue + Uy + I + C)/16) * U) * Qd) + ((FP * Qi) / 500)} * P * E} * PC]

Total MM = [[{{∑(FPf * (((Ue + Uy + I + C)/16) * U) * Qd) + ((FP * Qi) / 500)} * P * E} * PC] / 7] / 22

Project Cost= [{{∑(FPf * (((Ue + Uy + I + C)/16) * U) * Qd) + ((FP * Qi) / 500)} * P * E} * PC] * Per Hour Cost + Overhead Cost

IV. RESULTS/ CASE ANALYSIS

Five real life projects were taken for software test estimation in this paper. In the first project the calculation has been carried out basis on *functions* [2] and the second one has been carried out using *use cases* [9].

Note: Calculated value has taken for MM and Project Cost has taken from automated result given by the tool. So, there might be fractional difference that leads to variance in the equation and actual result.

Case 1: LDMACS (Management Accounts Consolidation System for Line Director)

TABLE 4
FUNCTION DEPENDENT AND DYNAMIC WIDTH FACTOR OF LDMACS

Function Dependent	Function/ Use Cases	RATING	WEIGHT	POINTS	Total
Ue	14.75	3.00	0.50	7.38	44.25
	29.50	6.00	0.75	22.13	
	14.75	12.00	1.00	14.75	
Uy	30.00	2.00	0.10	3.00	9.70
	20.00	4.00	0.20	4.00	
	9.00	12.00	0.30	2.70	
I	24.00	2.00	0.50	12.00	40.50
	26.00	4.00	0.75	19.50	
	9.00	8.00	1.00	9.00	
C	21.00	3.00	0.50	10.50	44.00
	18.00	6.00	0.75	13.50	
	20.00	12.00	1.00	20.00	
U	59.00	1.00	0.60	35.40	35.40
Df					306.32
Qd	39.00	3.00	0.05	1.95	4.70
	15.00	4.00	0.10	1.50	
	5.00	5.00	0.25	1.25	

Total MM= [[{{(1 * 306.32 * 4.70) + ((500 * 16) / 500)} * 1.2 * 0.67} * 1.05] / 7] / 22 = 7.94[MM]

Project Cost= [{{(1 * 306.32 * 4.70) + ((500 * 16) / 500)} * 1.2 * 0.67} * 1.05] * 400 + 116689.89= 589,387.66 [BDT]

Case 2: CREL (Climate Resilient Ecosystems and Livelihood)

TABLE 5
FUNCTION DEPENDENT AND DYNAMIC WIDTH FACTOR OF CREL

Function Dependent	Function/ Use Cases	RATING	WEIGHT	POINTS	Total
Ue	19.50	3.00	0.50	9.75	58.50
	39.00	6.00	0.75	29.25	
	19.50	12.00	1.00	19.50	
Uy	70.00	2.00	0.10	7.00	8.80
	6.00	4.00	0.20	1.20	
	2.00	12.00	0.30	0.60	
I	68.00	2.00	0.50	34.00	42.00
	8.00	4.00	0.75	6.00	
	2.00	8.00	1.00	2.00	
C	72.00	3.00	0.50	36.00	41.00
	4.00	6.00	0.75	3.00	
	2.00	12.00	1.00	2.00	
U	78.00	1.00	0.60	46.80	46.80
Df					439.63
Qd	71.00	3.00	0.05	3.55	4.35
	3.00	4.00	0.10	0.30	
	2.00	5.00	0.25	0.50	

Total MM= [[{{(1 * 439.63 * 4.35) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] / 7] / 22 = 7.66[MM]

Project Cost= [{{(1 * 439.63 * 4.35) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] * 500 + 117906.64 = 707439.84 [BDT]

Case 3: HRMS ((Police Bureau of Investigations (PBI) of Human Resources Management System))

TABLE 6
FUNCTION DEPENDENT AND DYNAMIC WIDTH FACTOR OF HRMS

Function Dependent	Function/ Use Cases	RATING	WEIGHT	POINTS	Total
Ue	30.75	3.00	0.50	15.38	92.25
	61.50	6.00	0.75	46.13	
	30.75	12.00	1.00	30.75	
Uy	80.00	2.00	0.10	8.00	18.60
	23.00	4.00	0.20	4.60	
	20.00	12.00	0.30	6.00	
I	90.00	2.00	0.50	45.00	73.00
	20.00	4.00	0.75	15.00	
	13.00	8.00	1.00	13.00	
C	115.00	3.00	0.50	57.50	64.00
	6.00	6.00	0.75	4.50	
	2.00	12.00	1.00	2.00	
U	123.00	1.00	0.60	73.80	73.80
Df					1143.21
Qd	85.00	3.00	0.05	4.25	8.80
	33.00	4.00	0.10	3.30	
	5.00	5.00	0.25	1.25	

Total MM= [[{{(1 * 1143.21 * 8.80) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] / 7] / 22 = 40.01[MM]

Project Cost= [{{(1 * 1143.21 * 8.80) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] * 500 + 616089.58 = 3696537.50 [BDT]

Case 4: PBI Website ((Police Bureau of Investigations (PBI) of PBI Website))

TABLE 7
FUNCTION DEPENDENT AND DYNAMIC WIDTH FACTOR OF PBI WEBSITE

Function Dependent	Function/ Use Cases	RATING	WEIGHT	POINTS	Total
Ue	17.50	3.00	0.50	8.75	52.50
	35.00	6.00	0.75	26.25	
	17.50	12.00	1.00	17.50	
Uy	65.00	2.00	0.10	6.50	7.80
	2.00	4.00	0.20	0.40	
	3.00	12.00	0.30	0.90	
I	63.00	2.00	0.50	31.50	37.75
	3.00	4.00	0.75	2.25	
	4.00	8.00	1.00	4.00	
C	59.00	3.00	0.50	29.50	39.00
	6.00	6.00	0.75	4.50	
	5.00	12.00	1.00	5.00	
U	70.00	1.00	0.60	42.00	42.00
Df					359.76
Qd	64.00	3.00	0.05	3.20	3.95
	5.00	4.00	0.10	0.50	
	1.00	5.00	0.25	0.25	

Total MM= [[{{(1 * 359.76 * 3.95) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] / 7] / 22 = 5.71[MM]
Project Cost= [{{(1 * 359.76 * 3.95) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] * 500 + 87864.56 = 527187.36 [BDT]

Case 5: EDCL (Essential Drugs Company Ltd. Sales Management Systems)

TABLE 8
FUNCTION DEPENDENT AND DYNAMIC WIDTH FACTOR OF EDCL

Function Dependent	Function/ Use Cases	RATING	WEIGHT	POINTS	Total
Ue	7.50	3.00	0.50	3.75	22.50
	15.00	6.00	0.75	11.25	
	7.50	12.00	1.00	7.50	
Uy	20.00	2.00	0.10	2.00	4.40
	6.00	4.00	0.20	1.20	
	4.00	12.00	0.30	1.20	
I	15.00	2.00	0.50	7.50	20.00
	10.00	4.00	0.75	7.50	
	5.00	8.00	1.00	5.00	
C	24.00	3.00	0.50	12.00	17.00
	4.00	6.00	0.75	3.00	
	2.00	12.00	1.00	2.00	
U	30.00	1.00	0.60	18.00	18.00
Df					71.89
Qd	22.00	3.00	0.05	1.10	2.35
	5.00	4.00	0.10	0.50	
	3.00	5.00	0.25	0.75	

Projects Cost ={(1*71.89*2.35)+((((500*16)/500)}*1.2* 0.48)*1.07)*500+21636.70 = 129,820.20 (BDT)

Total MM= [[{{(1 * 71.89 * 2.35) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] / 7] / 22 = 1.40[MM]

Project Cost= [{{(1*71.89*2.35) + ((500 * 16) / 500)} * 1.2 * 0.48} * 1.07] * 500 + 21636.70 = 129820.20 [BDT]

From table 9, we can easily understand that the estimation by tool is more accurate compare to estimation by project manager.

TABLE 9
COMPARISON OF TIME ESTIMATION

Project Name	*Estimated Time by Project Manager [MM]*	*Actual Time Required Required [MM]*	*Estimated Time with Research Tool [MM]*
LDMACS	6	8	7.94
CREL	4	6	7.66
HRMS	35	39	40.01
PBI Website	3	4	5.71
EDCL	2	1.5	1.4

V. CONCLUSION

The war against lowering the cost has not been over yet, however, the researchers have to implement better ideas and thoughts to reduce the software cost. This paper demonstrates the software test estimation tool, which is clearly defined as well as understandable, that provides the time and cost of any project. Finally, the researchers have successfully implemented the software test estimation tool over two real projects and it gives accurate results as per software experts. Following the guideline of the tool it is really easy task to generate the time and cost for any project. Now, the researchers can implement more projects to get statistics of accuracy to make it remarkable as other standard tools e.g. Use Cases Point to estimate any software development projects.

REFERENCES

[1] E.P.W.M. Veenendaal, T. Dekkers, "Testpointanalysis: a method for test estimation", *Project Control for Software Quality,* Shaker Publishing BV, The Netherlands, 1999.

[2] Y. Ahn, J. Suh, S. Kim and H. Kim, "The software maintenance project effort estimation model based on function points", *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 15, pp.71–85, 2003.

[3] Function Point Counting Practices, Release 4.1, International Function Point User Group (IFPUG) 1994, Internet:http://www.bilkent.edu.tr/~csevgi/courses/ctis359/IFPUG_Counting_Practices_Manual_4.1.pdf [Access: May, 2013]

[4] Pathik, B. B, Islam, S., Khan, M. H. and Habib, M. M., "Software Test Estimation Tools using Use Cases and Functions", Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEE IEEM), Bangkok, Thailand, 10-13 December 2013. ISBN: 978-1-4799-0985-8.

[5] Bevan Nigel, "Quality in Use: Meeting User Needs for Quality", *Journal of System and Software*, 1999.

[6] ISO/IEC TR 9126-2 (1997), Information Technology - Software Engineering Product Quality - Part 2: External metrics, International Organization of Standardization, Internet:http://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-2%20Standard.doc, [Access: April, 2013]

[7] The Application of Function Point Analysis in the Early Phases of the Application Life Cycle A Practical Manual, Theory and Case Study, Version 2.0, Manual of the Netherlands Software Metrics Users Association, Internet: http://www.nesma.nl/download/boeken_NESMA/N20_FPA_in_Early_Phases_(v2.0).pdf [Access: May, 2013]

[8] Leung, Hareton and Fan, Zhang, "Software Cost Estimation", *Handbook of Software Engineering And Knowledge Engineering*, World Scientific Pub. Co, River Edge, Nj, 2002

[9] Chris F. Kemerer, "An empirical validation of software cost estimation models", *Communications of the ACM*, Volume 30, Number 5, May 1987. ISSN: 0001-0782/87/0500-0416
Schneider, G. and Winters, J. "Applying Use Cases – A Practical Guide", Addison-Wesley. 1998, http://books.google.com.bd/books/about/Applying_Use_Cases.html?id=IKlQAAAAMAAJ&redir_esc=y [Access: May, 2013]

[10] Software Estimation Best Practices. Internet: http://www.galorath.com/wp/estimating-best-practices.php, [Access: April, 20113]

[11] Grimstad, S. and Jorgensen, M., "A framework for the Analysis of software cost estimation accuracy", *ISESE '06* Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, pp. 58 - 65, ISBN:1-59593-218-6.