May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

Modern Metrics (Mm): Function Point based Size Estimation Technique for Modern Software

[1]John T Mesia Dhas, [2]T.S. Shiny Angel, [3]J. Sheeba
[1]Associate Professor, Department of Computer Science and Engineering, Audisankara College of Engineering and Technology, Gudur, Andhrapradesh, India, Mail: jtmdhasres@gmail.com
[2]Assistant Professor Sr. grade, Department of Software Engineering, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India, Mail: shinyangeldavid@gmail.com
[3]Associate Professor, Mohamed Sathak A.J Academy of Architecture, Chennai, Tamil Nadu, India sheeba4archi@gmail.com

Article Info
Volume 83
Page Number: 235 - 246
Publication Issue:
May - June 2020

Abstract:
About fifteen distinct programming languages, operating system, development tools and utility software are used for developing a new software system. The programming language independent, operating system neutral, highly extensible and dynamic are the behaviors of modern software system. The existing particularistic approached software sizing techniques are not good for estimating the size of versatile modern software. Modern Metrics (MM) is a novel method for estimating the size of modern software system. MM is independent of computer languages, operating system, development methodology, application domain and technology behind the development. MM can be estimated early in the analysis and design phase of the System Development Life Cycle (SDLC) and is prepared based on the user, developer and environmental perspectives. This novel method Modern Metrics (MM) analyses all possible functional units and complexity factors of modern software. So, the defects present in the Function Point Analysis (FPA) is reduced. MM considers internal inputs, internal operations, database, SDLCs, output formats, international standards and multiple software usage. It increases the accuracy of the results and also reflects good results in cost, size and time constraints.The performance of MM is accurate in industrial results in developing the software compared with existing FPA method. The result analysis of MM an FPA with Software Project Management (SPM) metrics like size, effort, cost and time implies, MM is more accurate than FPA and MM is suitable approach for calculating the size of modern software system. So, this research concludes that MM method is a successful approach to determine size of modern software system and it leads to the success of project management activities of modern software system development.

Article History
Article Received*: 11August 2019*
Revised*: 18November 2019*
Accepted*: 23January 2020*
Publication*:07May2020*

Keywords*: Modern Metrics, Modern Software, Software size, Software Project Management, MMSize, Function Points.*

I. Modern Metrics Sizing Technique

Modern Metrics (MM) is the proposed sizing technique for modern software which is based on new metrics and values. MM is a novel approach, that estimates the size of the software with less cost and time. The modern software mainly does the extraction, processing of data and value based on decision making. Apart from the traditional function points like External Input (EI), External Output (EO), Internal Logical Files (ILF), External Inquiries (EQ) and External Interface File (EIF), it includes Internal Input (II), Internal Operations (IO) and Data and Text (DT). It also recognizes System Development Life Cycle (SDLC), updated Complexity Adjustment Factors (CAF), Trial versions of the software, Indexed data, Multiple forms of output, user developer views on system and

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

Social, Economic and Political laws of the Nation. Therefore, the defects per function point is reduced by the novel Function Point Analysis (FPA), using MM technique.

A. Modern Metrics

Modern Metrics (MM) is an Indian metrics which will measure the size of a software with the help of updated functional units of modern software. MM has some simple calculations for finding the size of modern software. It is not considering programming language, operating system, development tools, working environment and other technical factors. Hence, a novice or non-software professional can easily estimate the size of software.

Architecture of MM

The functional diagram of Modern Metrics (MM) includes all the internal and external function points of a software system. The traditional IFPUG function point estimation technique has only five functional units (External Input, External Output, External Inquiries, External Interface files and Internal Logical Files). But the MM has eight functional units (External Input, External Output, External Inquiries, External Interface files and Internal Logical Files, Internal Inputs, Internal Operations and Data and Text). The MM also includes twenty two Complexity Adjustment Factors (CAF) but the traditional IFPUG function point calculation has only eighteen CAF. The architectural diagram of Modern Metrics (MM) is shown in the following Figure 1:

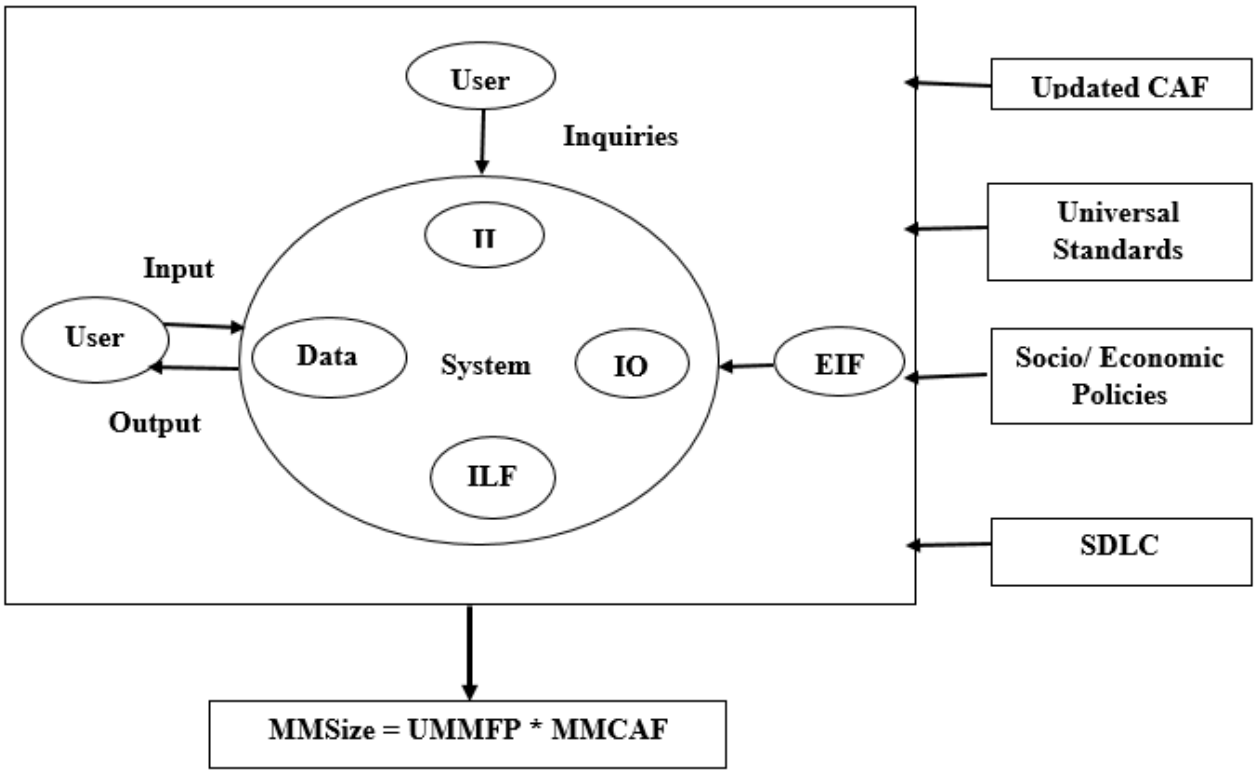

Figure 1: Architecture of MM

II – Internal Inputs
IO – Internal Operations
ILF – Internal Logical Files
EIF – External Interface Files
CAF – Complexity Adjustment Factors
SDLC – System Development Life Cycle
MMSize – Modern Metrics Size

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

UMMFP – Unadjusted Modern Metrics Function Points
MMCAF – Modern Metrics Complexity Adjustment Factor

Functional Units of MM

The functional units of a software is the basic element for estimating the size of a software. The functional units are divided into two categories based on its functional view. They are, internal functional units, external functional units and hybrid functional units. The internal functional units are influencing the system internally and which will not interact with the external actors. Like that, external functional units are influencing the system by external actors or communications from system to an external actor. In Modern Metrics, internal inputs, internal operations and internal logical files are internal functional units. Other functional units like, external inputs, external outputs, external inquiries and external interface files are external functional units. The data and text is having the behavior of both internal and external functional units. So it is a hybrid functional unit.

Internal Functional units:

a) Internal Inputs (II): The defined constants and internal assignments of variables are internal inputs.
b) Internal Operations (IO): A complete cycle of operations in the system but which is not present under any other functional units.
c) Internal Logical Files (ILF): It is a supporting software or data present in the system for executing the system successfully.

External functional units:

a) External Inputs (EI): Inputs given to the system through input devices by an external actor.
b) External Outputs (EO): The results received from the system through output devices for an external actor.
c) External Inquiries (EQ): The external questions raised from the actor during the execution time for checking the accuracy of the system.
d) External Interface Files (EIF): It is a supporting software or data present in the external system for executing the software successfully.

Hybrid functional units:

a) Data and Text (DT): 8000 words (manual typing speed of a person per day) in a text document is a functional unit of DT. The DT may not take part any operation and it may be tables, historical data, help files, images or other text documents. It may be both internal and external.

The Metrics of the Functional Units of MM

The metrics of the functional units of modern software is difficult to find and classify it. So some important functional units of functions are identified and listed in the following Table 1,

S.No	Functional Unit	Metrics
1	Internal Inputs (II)	Constants, internal assignments, internal keys
2	Internal Operations (IO)	Choices, A complete operational cycle which is not taking part with any other functional calculations, dynamic effects of webpages, internal algorithms, Array input, output or calculations, the properties and events assigned to the GUIs, function calling in a program
3	Internal Logical Files (ILF)	The driver files for other software, header files, packages
4	External Inputs (EI)	Inputs given through input ports or input statements, input GUI's like text box, list box, combo box etc., Graphics coordinates for a complete diagram (example circle, line, ellipse etc.) with its properties
5	External Outputs (EO)	The results displayed using output devices, output GUIs like label box, list box, text box, combo box
6	External Inquiries (EQ)	The queries generated by the users for the better operations of the system
7	External Interface Files (EIF)	The driver files used for external devices and remote systems, anchor tags,
8	Data and Text (DT)	Tables, Text files, image files, help files, data files, Webpage contents

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

Functional Units with Metrics and Metric values of MM

The eight functional units are ordered according to their availability in a function. The metrics of the functional units are Low, Average, High and Very High based on the complexity and time required to complete the operations of each functional unit. These metrics are otherwise known as effort modifiers of the software sizing process. By using a set of inflexible standards the metrics are categorized following Table 2.

Table 2: Functional Units with Metrics

Metrics	Functional Units							
	EI	***II***	***EO***	***IO***	***DT***	***EQ***	***ILF***	***EIF***
Low	1 to 3	1 to 3	1 to 4	1 to 3	1 to 4	1 to 3	1 to 7	1 to 5
Average	4 to 5	4 to 5	5 to 6	4 to 5	5 to 6	4 to 5	8 to 14	6 to 9
High	6 to 8	6 to 8	7 to 9	6 to 8	7 to 9	6 to 8	15 to 21	10 to 13
Very High	>8	>8	>9	>8	>9	>9	>21	>13

If a function has 1 to 3 EI then, the metrics of EI is Low. Similarly, all the metrics are identified in a function and are tabulated. The metric values are effort modifiers of MM listed in the following Table 3.

Table 3: Metrics with its Values

Metrics	EI	II	EO	IO	DT	EQ	ILF	EIF
Low	3	3	4	3	4	3	7	5
Average	4	4	5	4	5	4	10	7
High	6	6	7	6	7	6	15	10
Very High	9	9	10	9	10	9	22	14

Calculating FUs of MM

All the classes and functions are analyzed and listed all the corresponding functional units using following Table 4 format. All the functional units are identified in each functions of software and tabulated. The total number of functions referred and total functional units of each type are calculated at the end of Table 4.

Table 4: Calculating Functional Units

S. No	Name of the Function	EI	II	EO	IO	DT	EQ	ILF	EIF
1									
2									
3									
4									
5									
Total number of functions referred (TF)									
Total functional units (TFU)									

Complexity Adjustment Factors (CAF) of MM

The project complexity and management process is one of the challenging task in the size estimation of modern software. In most of the projects, the complexity of a project will be measured in based on its degree of novelty, its interdependencies, and the technologies involved. The level of complexity may be the duties, the degree of autonomy and the scope of responsibilities.

The complexity of modern software is derived based on the following reasons,

a) Technology used in the software.
b) Standardisation and development models associated to the software.
c) Distribution and processing of application.
d) The novelty and innovation of the developing system.
e) Uncertainty of the software system

The complexity of the software is determined using the following Complexity Factors (Fi). They are,

1. Whether backup is required to the system?
2. Whether data communication is important?

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

3. Whether it has any distributed processing?
4. Is representation complex?
5. Whether the system works in congested environment?
6. Is it requires any online updating?
7. Whether the system has online input, output and operations?
8. Is it require any major file on online updating?
9. Is it work in multi environment?
10. Is the internal operation critical?
11. Is it reusable?
12. Whether the software is extensible?
13. Is it good for different organizations?
14. Is it permit the user interactions?
15. Whether the system uses indexed or listed data (single index or multi index)?
16. Whether the system uses more than one SDLC models?
17. Is the system using more than one programming languages, Data Base Management Systems (DBMS), Web tools, Drivers, etc.?
18. Is the networking environment using more than one network topologies?
19. Is the system installed in different nations and uses different social, cultural, economic and environmental laws?
20. Is the system giving multiple forms of output?
21. Is the trial version and model version of software development affects the system?
22. Is User Interface influence the system?

The influence of the complexity factors of a software is measured using the influential values (Nil = 0, Secondary = 1, Moderate = 2, Average = 3, Important = 4, Essential = 5) assigned to the Complexity Factors. The following Equation (1) gives the value of Modern Metrics Complexity Adjustment Factor (MMCAF) of the software.

$$MMCAF = 0.25 + 0.01 * Fi \quad (1)$$

The Fi (i = 1 to 22 factors) is the amount of influence and are based on responses to complexity factors.

Calculating Unadjusted Modern Metrics Function Points (UMMFP)

The UMMFP is the number of raw function points present in a software. The following Table 5 is used to calculate the UMMFP

Table 5: Unadjusted MMFP

S. No	Functional Units	Total Number of Functions (TF)	Total Functional Units (TFU)	Average Functional Units (AFU = TFU / TF)	Metrics	Metric Value (W)	UMMFP (TF * W)
1	EI						
2	II						
3	EO						
4	IO						
5	DT						
6	EQ						
7	ILF						
8	EIF						
						Total UMMFP	

To find the value of UMMFP, we must calculate Total number of Functions (TF), Total Functional Units (TFU), Average Functional Units (AFU), weighting factor and weightage of the functional units.

The total number of functions is sum of the functions having the functional units of each type. It is calculated during the functional unit calculations of each functions of a software. The function having any functional unit, immediately the corresponding functions count is increased by one.

The distinct functional units of each function is calculated and tabled using Table 4. The total sum of

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

each functional units in all functions is the total functional units.
The ratio of total functional units and total number of functions is known as average functional units.
The value of average functional units is used to calculate weighting factor and weightage of the functional units using Table 2 and Table 3.
The Unadjusted Function Point (UFP) of each functional units is calculated. That is the product of total number of functions and weightage.
The Unadjusted Modern Metrics Function Point (UMMFP) is the sum of all the Unadjusted Function Points of each functional unit,

MMSize
MMSize is the size of the software based on Modern Metrics. The unit of Modern Metrics (MM) software size is MM. It is calculated using the Equation (2)
MMSize = UMMFP x MMCAF (2)
The Modern Metrics Size (MMSize) is the product of Unadjusted Modern Metrics Function Points (UMMFP) and Modern Metrics Complexity Adjustment Factor (MMCAF).

B. Other Estimations Based on MM

MM Productivity Factor
Modern Metrics Productivity Factor (MMPF) defines the amount of time required for completing one function point. The productivity factor may change from organization to organization. PF is calculated using the following Equation (3),
MMPF = Total Hours required to Complete a project / MMSize (3)

MM Effort
Software Effort denotes the amount of man hours required for completion of the project [9,66]. Software size is the primary independent variable affecting software development effort. The following Equation (4) is used for calculating effort using MM.
Modern Metrics Effort (MME) = MMSize * PF (4)
Where MMSize = Size of software using Modern Metrics
PF = Productivity Factor.
Productivity factor defines the amount of time required for completing one function point. The productivity factor may change from organization to organization. Our organization uses productivity factor as 16 because they took in and average 16 hours per Modern Metrics Function points.

MM Duration
Duration denotes the total time required for completing the project. The following Equation (5) is used for calculating Duration using Modern Metrics
Modern Metrics Duration (MMD) = Modern Metrics Effort (MME) / (176 * number of persons involved in the software development) (5)
Here 176, denotes working hours per month that means Indian software industry people work on 22 days per month and per day 8 hours, totally 22*8 = 176 hours.

MM Cost
The cost of the software project is calculated based on the total expenditure for the development of the software. The following Equation (6) is used for calculating Cost of the project using MM.
Modern Metrics Cost (MMC) = Number of persons involved * Average remuneration of software developers + Management cost (6)
The management cost will be varied from organization to organization. The unit cost of Modern Metrics Size is calculated using the following Equation (7).
Modern Metrics Unit Cost (MMUC) = Modern Metrics Cost (MMC) / MMSize (7)

II. SIZE ESTIMATION USING MODERN METRICS

Modern Metrics (MM) is a novel technique for estimating the size of modern software system

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

based on its internal, external and hybrid function points. The previous chapter 4 analyses the procedure for implementing the MM. This chapter 5 is giving the practical implementation of MM.

A. Calculating the Functional Units

The functional units of each function is analyzed separately and tabulated using the following Table 6

Table 6: Functional Units calculation

S. No	Name of the Function	EI	II	EO	IO	DT	EQ	ILF	EIF
1	allsched1	5		6				1	2
2	cprocess1	5		7				1	2
3	cprocess2	3	3	4				1	2
4	cpwd1	2					1		1
5	cpwd	4	6				1		2
6	cregister	9		2	1	1			2
7	ctransit1	1		10	3			1	2
8	ctransit	1		1					2
9	czpro	3		1					1
10	dt1	1		2		1			1
11	dt2		6	2		1		1	2
12	dt3	2				1		1	2
13	fcitizen		5				1	1	2
14	lic2			2				1	2
15	licapp1	1		2	1			1	2
16	licapp2	1	3	6				1	2
17	licapp3		6	2				1	2
18	licapp11	1		2				1	2
19	licpro2		6	3				1	2
20	licst2	1		3			1	1	2
21	licst3	1	8	10				1	2
22	pinmast1	1	7	3	3				2
23	pinmast			2				1	2
24	pp1	1		2			1	1	2
25	ppst1	1	4	6					2
26	ppst11	1	1	1					2
27	prolic2	1		1	1			1	2
28	register					1		1	1
29	registerc					1	1	1	1
30	sappno		1	1					2
31	signin		3	4	3		1		2
32	sregister		3	2		1	1	1	2
33	tprolic	1	1					1	2
34	transit1	1		1			1	1	2
35	transit	1	1	1					2
36	tsched	1	6					1	2
37	updlic	4		1			1	1	2
38	vastaff		4	1					2
39	vcz1	1	10	7					2
40	vcz			1					1
41	vpp1	1	1	2	2		1		1
42	vpp2	1	11	12				1	1
43	vpp3	3		4					2
44	vpppro2	5		2				1	2
45	vpropp1	1	1	3				1	2
Total number of functions referred (TF)		33	23	38	6	7	11	29	45
Total functional units (TFU)		69	100	124	11	20	16	29	87

B. Unadjusted MM Function Point Calculation

The unadjusted Modern Metrics function points (UMMFP) of Aadhar processing system is calculated using the following Table 7

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

Table 7: Unadjusted MMFP calculation

S. No	Functional Units	Total Number of Functions (TF)	Total Functional Units (TFU)	Average Functional Units (AFU = TFU / TF)	Metrics	Metric Value (W)	UMMFP (TF * W)
1	EI	33	69	2.0909090	Low	3	99
2	II	23	100	4.3478260	Average	4	92
3	EO	38	124	3.2631578	Low	4	152
4	IO	6	11	1.8333333	Low	3	18
5	DT	7	20	2.8571428	Low	4	28
6	EQ	11	16	1.4545454	Low	3	33
7	ILF	29	29	1	Low	7	203
8	EIF	46	87	1.8913043	Low	5	230
						Total UMMFP	855

In the software, Aadhar processing system having total of 45 functions. 33 functions having 69 External Inputs, 23 functions having 100 Internal Inputs, 38 functions having 124 External Outputs, 6 functions having 11 Internal Operations, 7 functions having 20 Data and Text, 11 functions having 16 External Inquiries, 29 functions having 29 Internal Logical Files and 45 functions having 87 External Interface Files.

The average functional units are calculated based on the ratio of total functional units and total number of functions. Based on this value, the weighting factor and weightage is calculated based on Table 2 and Table 3 respectively.

Unadjusted Function point of each functional unit is calculated, which is the product of total number of functions and weightage of each functional units.

The Unadjusted Modern Metrics Function Point is the sum of unadjusted function point of all the functional units. The UMMFP of Aadhar processing system is 855

C. Complexity Adjustment Factor

The complexity factors of the Aadhar Processing system is present in the following Table 8

Table 8: MMCAF

S. No	Factors	Scale of Factors						
		Nil (0)	Secondary (1)	Moderate (2)	Average (3)	Important (4)	Essential (5)	Value
1	Does the system need unfailing backup and recovery?						5	5
2	Is data communication necessary?				3			3
3	Are there distributed processing jobs?				3			3
4	Is act complex and critical?					4		4
5	Will the system work in an existing mainly utilized operational environment?						5	5
6	Does the system need on line data entry?						5	5

TEST Engineering & Management

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

S. No	Factors	Scale of Factors						
		Nil (0)	Secondary (1)	Moderate (2)	Average (3)	Important (4)	Essential (5)	Value
7	Does the on line data entry needs the input operation to be built over many screens or operations?						5	5
8	Is the original file updated on line?						5	5
9	Is the inputs, outputs, files, or inquiries complex?				3			3
10	Is the internal processing complex?				3			3
11	Is the code designed to be reusable?					4		4
12	Are change and installation included in the plan?			2				2
13	Is the system designed for many installations in different organizations?					4		4
14	Is the application designed to ease change and ease of use by the user?					4		4
15	Is the system using indexed or list data (single index or multi index)?			2				2
16	Whether the system using more than one SDLC models?			2				2
17	Is the system using more than one programming language, Data Base Management System (DBMS), Web tools, Drivers etc.?			2				2
18	Is the networking environment using more than one network topologies?				3			3
19	Is the system installed in different nations and uses different social, cultural, economic and environmental law?	0						0
20	Is the system giving multiple form of output?						5	5
21	Is the trial version and model version of software development affecting the system?				3			3
22	Is User Interface influencing the system?						5	5
							Total CAF	82

The complexity of modern software is derived based on the technology used in software, standardization and development models associated to software, distribution and processing of application, novelty and innovation of developing system and uncertainty of the software system. The complexity of Aadhar processing system is also derived based on these factors. The complexity adjustment factor of this software is 82.

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

D. Mm Complexity Adjustment Factor

The value of MMCAF is calculated using the Equation 1,

MMCAF = 0.25 + 0.01 * CAF
= 0.25 + 0.01 * 82
= 1.07

E. Modern Metrics Software Size

MMSize of the software is calculated using the Equation 2,

MMSize = UMMFP * MMCAF
= 855 * 1.07
= 914.85 MMFP

F. Productivity Factor

Total number of days required for completing the project = 120

Total number of persons involved for the development = 6

Total number of hours required to complete the project = 120 * 6 * 8
= 5760 Hours

MMPF = 5760 / 914.85
= 6.29

(6 Hours and 18 Minutes required for completing a MM Function Point)

G. Effort

= 914.85 * 6.29
= 5754.40

(5754 Hours and 24 Minutes) Man Hours required for completing the project Aadhar Card Processing System.

H. Duration

MMD = 5754.40 / (176 * 6)
= 5.18 Months

(5 Months and 7 Days) of time required to complete the project.

I. Cost

The average remuneration of a software developer per month = 22950.75 (Indian Rupee)

Total number of months required for completing project = 5.18

Average remuneration for a developer
= 22950.75 * 5.18
= 118884.88 (Indian Rupee)

Management Cost =210000 (Indian Rupee)

MMC =118884.88 * 6 + 210000
=923309.28 (Indian Rupee)

J. Unit Cost of MMFP

MMUC =923309.28/914.85
= 1009.24 (Indian Rupee)

K. Price of the Software

Let we assume, the maintenance cost is 40% of the MM cost and percentage of profit is 30%, then price of the software is

MM Price = 923309.28 + (923309.28+ (923309.28*40/100)) * 30/100
= 923309.28+ (923309.28+ (923309.28* 0.4)) *0.3
= 1311099.17(Indian Rupees)

II. Conclusion

This proposed innovative approach MM is used for calculating the size of the software at the early stages of SDLC. The difficulties with budgeting and delivery of the software product is overwhelmed. The traditional FPA based sizing techniques are considering only the user perspectives but, the proposed MM technique considers user and developer perspectives. So, the defects in functional units of MM technique is negligible.

The MM technique uses eight functional units over traditional FPA's five functional units. The MM technique uses twenty two complexity factors over traditional FPA's fourteen complexity factors. This updates are increasing the accuracy of the size of the software.

The MM technique reduces the inflated functional units of traditional FPA. Therefore, MM technique reduces around 20% to 30% of size in application software over FPA. The MM technique considers internal operations, multiple forms of outputs and database used in the application. Therefore, MM technique gives actual size of the scientific, artificial intelligence, webpages and game playing software.

May – June 2020
ISSN: 0193-4120 Page No. 235 - 246

The undefined functional units of design and modelling software like Computer Aided Designing, Computer Aided Modelling etc. shall be considered in the future studies.

References

[1] Abran A., Cuadrado J., and Desharnais J. M. (2006), "Convertibility of Function Points to COSMICFFP: Identification and Analysis of Functional Outliers", MENSURA 2006, Cadiz (Spain), pp. 6-8.

[2] Boehm B. W. (1981), "Software Engineering Economics", Prentice-Hall.

[3] Capers Jones (2007), "Estimating Software Costs: Bringing Realism to Estimating", Tata McGraw Hill, Second Edition.

[4] Capers Jones (2008), "Applied Software Measurement-Global Analysis of Productivity and Quality", Tata McGraw Hill.

[5] Capers Jones (2010), "Software Engineering Best Practices: Lessons from Successful projects in the top companies", Tata McGraw Hill.

[6] Dalkey N. and Helmer O. (1963), "An Experimental Application of the Delphi Method to the Use of Experts", Management Science, pp. 458-467.

[7] Daniel V. Ferens (1999), "The Conundrum Software Estimation Models", Air Force Research Laboratory (AFRLIIFSD), IEEE, pp. 23-29.

[8] Edilson J. D. Cândido and RoselySanches (2003), "Estimating the size of web applications by using a simplified function point method", IEEE.

[9] Galorath D. D. and Evans M. W. (2006), "Software Sizing, Estimation, and Risk Management", Boston, MA, USA: Auerbach Publications.

[10] Hughes R. T. (1996), "Expert Judgement as an Estimating Method", Information and Software Technology, pp. 67-75.

[11] ImanAttarzadeh and Siew Hock Ow (2009), "Proposing a New High Performance Model for Software Cost Estimation", Second International Conference on Computer and Electrical Engineering, IEEE, pp. 112-116.

[12] ISO/IEC 20968 (2002), "Software Engineering – Mk II Function Point Analysis – Counting Practices Manual", International Organization for Standardization – ISO, Geneva.

[13] ISO/IEC 19761 (2003),"Software Engineering – COSMIC-FP A Functional Size Measurement Method", International Organization for Standardization – ISO, Geneva.

[14] ISO/IEC 20926 (2003), "Software Engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual", International Organization for Standardization – ISO, Geneva.

[15] ISO/IEC 24750 (2005), "Software Engineering – NESMA functional size measurement method version 2.1 – Definitions and counting guidelines for the application of Function Points Analysis", International Organization for Standardization – ISO, Geneva.

[16] Karner G. (1993), "Resource Estimation for Objectory Projects", Objective Systems.

[17] Linda M. Laird (2006), "The Limitations of Estimation. IEEE Computer Society", IEEE, pp. 40–45.

[18] Mahir Kaya and OnurDemirörs (2011), "E-Cosmic: A Business Process Model Based Functional Size Estimation Approach", 37th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE, pp. 404-408.

[19] Md.Forhad Rabbi, ShailendraNatraj and OlorisadeBabatundeKazeem (2009), "Evaluation of convertibility issues between IFPUG and COSMIC function points", Fourth International Conference on Software Engineering Advances, IEEE, pp. 277-281.

[20] Mehwish Nasir and Farooq Ahmad H. (2006), "An Empirical Study to Investigate Software Estimation Trend in Organizations Targeting CMMISM", Proceedings of the 5th IEEE/ACIS

International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06), IEEE.

[21] Putnam L. H. (1978), "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, pp. 345-361.

[22] Richard D. Stutzke (2005), "Estimating Software-Intensive Systems: Projects, Products and processes", SEI Series in Software Engineering, Addison Wesley.

[23] Robert T. Futrell, Donald F. Shafer and Linda I. Shafer (2008), "Quality Software Project Management", Pearson Education.