Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 26 (2023), No. 3, pp. 629–640
DOI : 10.47974/JDMSC-1734

Modern Metrics (MM): Software size estimation using function points for artificial intelligence and data analytics applications and finding the effort modifiers of the functional units using indian software industry

John T Mesia Dhas *
Department of Computer Science and Engineering
Lincoln University College
Malaysia

J. Midhunchakravarthy †
Faculty of Computer Science and Multimedia
Lincoln University College
Malaysia

Abstract

SPM, Software Project Management is an engineering process for scheduling, developing, testing, and maintaining a software system. The size of a software is one of the important attributes in determining all the activities of SPM. Price, time, and effort are the basic factors for developing software in a successful and profitable manner. The fast explosions in technological development of software systems attracted all the fields that existed in the universe. The modern software system uses many programming languages, domains, operating systems, technologies, development cycles, topologies, etc. So, the industry is expecting a new, versatile, and highly updated sizing technique for modern software. Modern Metrics (MM) is a new anticipated size estimation technique for present software systems with updated functional values and metrics using Function Point (FP). This article analyzed the functional unit metric values (effort modifiers) of the MM with real world applications. MM is a novel method for determining the solution to the SPM challenges.

Subject Classification: *91B32.*

Keywords: *Software project management, Software size estimation, Modern metrics, Software complexity factors, Software effort, Software schedule, Modern metrics size.*

* *E-mail:* **jtmdhasres@gmail.com (Corresponding Author)**
† *E-mail:* **midhun@lincoln.edu.my**

1. Introduction

Software engineering is an engineering process for developing software projects in an international standard manner, giving the guidelines for completing the project on time, testing and debugging in user as well as developer centric approaches, and giving the complete procedure for maintaining the developed software. Modern software systems use more than one programming language, application software, utility tool, architecture, platform (operating system), type of user, and databases. So, the development of a modern software system is highly complex. Modern applications are based on software as well as hardware. The systems (both hardware and software) are generating a huge amount of data. That data must be processed at the same time for the next process [1].

Software is an intangible logical product. So, the estimation of size is highly risky. The software industry is using many traditional size estimation techniques like Lines of Code (LoC), Feature Point, Delphi, Expert Judgment, Function Point (FP), Pattern Matching, etc. In terms of software sizing techniques, the FP method is the most reliable, programming language independent, user-centric, and understandable to beginners. The FP method estimates the size of the software at the analysis front itself. So, the estimation of effort, cost, and time is possible at the initial phase itself. Around 90% of the software industries in the world are following the FP method for determining the software size [2-4].

In the 1970s, Alan Albrecht introduced the FP method for defining the software size [5-10]. The key features of the FP method are: Independent of technology, platform, or language; Excellent as a starting point for estimating development costs and timelines; It provides the ability to normalize metrics for consistent analysis.

The FP method is using five important function points for measuring the size of the software. They are, Inputs, Outputs, Inquiries, Internal Logical Data and External Interfaces.

International Function Point Users Group (IFPUG) is an autonomous agency responsible for handling the advancement of FP. The current version of IFPUG FP is 4.3.1.

2. Literature Review

The early eras of the software industry considered only the lines of code present in a software project. According to Capers Jones [1-5],

counting the lines of code was the best practice for finding the software size because of the homogeneity of the software advance process from 1947 to 1957. The advent of new software like COBOL and FORTRAN (1957 to 1967) increased the hands of software development and started to develop many new applications in procedure-oriented and intermediate programming language domains. This extension is multipurpose and the complexity of the code differs from application to application. So that the existing lines of code are not used to find the size of software products.

Expert judgement is another software sizing method introduced by Helmer of RAND Corporation in the 1950s. The support of an expert is required to find the software size [2].

The proportions of scientific software applications is estimated using operands and operators used in the application. This concept was proposed by M.H. Halsted in 1969 in his software science matrix [2].

The advent of programming languages increased the code complexity and LoC in the software development process. also increased the effort level of the software development process [1].

In 1965, a new metric named "Function Points" was introduced by one of the IBM researchers, Allan Albrecht [1-5]. The function points are considered inputs, outputs, inquires, logical files, and interfaces. It is the first method that determines the size and complexity based on functional values.

Mehwish Nasir and H. Farooq Ahmad [6] suggested that an individual sizing method is not chosen for finding the software size. The initial identification of size leads the software development process in time, cost, and effort.

Mahir Kaya et al., [8] The application size is the base factor to find all other metrics of the software development process. Estimating the size early on will reduce the time, cost, and effort complexities.

As Daniel V. Feren et al. [10] state, a new technique is required for finding the modern software size. It must be independent of its environment and must be measured in pre phases of the Software Development Life Cycle (SDLC).

Dr. Barry Boehm [11] states that the size estimate is the biggest problem for modern dynamic software systems.

Zia et al. [12] state that for doing software development analysis, project planning, risk analysis, and budgeting, software size is a key factor.

Md. Forhad Rabbi et al. [13] state that the software industry has matured. The earlier estimation using the existing techniques may have been wrong in the later stages. To estimate the actual size of modern software, a dynamic effective cost estimation technique is required.

2.1 *Findings*

The concluded findings are: Productivity is assessed based on software size; Software size is an inevitable factor in the software development process; The existing sizing techniques are particularistic and give distinct results with distinct software; Software size as a basic factor for scheduling, risk analysis, and project planning; If the software development methodology is modified, then the sizing technique must also be changed; A single sizing technique is not enough for finding the software size of modern applications; The existing sizing techniques are environment-centric; Existing sizing techniques are not adopted for modern software systems.

2.2 *Challenges in FPA*

Based on the studies with the existing FPA methods and the International Function Point Users Group's (IFPUG) FPA method, suggests following challenges. They are [14-18], The defect per function point is 4.5 in the IFPUG-FPA method.; Internal operations like multi-layered procedures and substantial designs that are a portion of a operation's processing rationality are not distinctly measured as a portion of the functional sizing; The choice-based selection (e.g., if structure, case structure) does not get extra size; FP merely reflects communication among the (external) people and the project. Communications among several inside portions of project are not determined using FP method; Relocation of Graphical User Interface (GUI) essentials deprived of count/removing/adjusting some of it is not encompassed in the sizing method; If a similar result is generated in several presentations or models (e.g., Text and Tables), no extra value is considered for the several models (i.e., only one model is comprised for the size estimation); The database or text files are not present in the FP count; The trial versions and model versions of the software are not included in the FP count. It reduces the effort of the developer; Internal Inputs (II), Indexed and List data do not have any importance in the FP count; The Software Development Life Cycle

(SDLC), standardizations, social, economic, and topographical features; distinct topologies; distinct languages; distinct data bases; web tools; and interfaces are not taken into consideration in the FP count; Trivial Function Points (TFP) like Multi Valued FP, Dependent FP, and Composite FP are not present in the FP calculations; The cost of estimating FP is high. So, the very large-scale projects are not estimated using the FP method.

3. Objectives and Motivation

The objectives are, to identify the missing FPs in the modern AI and Big Data Analytics (BDA) software system; To identify all the complexity adjustment factors (CAF) affecting the SDLC; To derive a methodology for finding the size of the software.

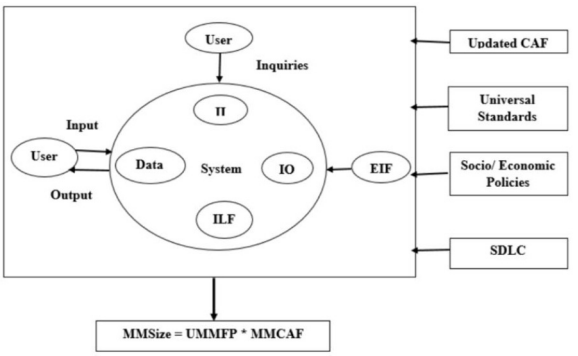
The motivation of the research: the software industry is a highly dynamic and variable industry that depends on time and situations. The size estimation techniques do not match with the current industrial flow. So many defects are present in the estimations based on existing techniques, which is affecting the delivery of the software product. To overcome the current scenario, the industry requires an updated estimation technique for identifying the software size.

4. Modern Metrics (MM)

MM is a proposed new methodology to identify software size like AI and BDA applications. It follows Alan Albrecht's FPA method with new and updated function points and complexity adjustment factors (CAF). This method is language, platform, and technology independent. The resultant value of this method is an important factor for estimating time, effort, and cost of the software product. The estimation process is fully documentable and repeatable. This method is capable of identifying the software size at any stage of the development phase. The estimation process gives quantitative results which are easily understandable by any kind of user.

4.1 *Architecture of MM*

The MM is comprised of FPs and CAFs. The MM architecture is illustrated at Figure 1.



II – Internal Inputs, IO – Internal Operations, ILF – Internal Logical Files, EIF – External Interface Files, CAF – Complexity Adjustment Factors, SDLC – Software Development Life Cycle, MMSize – Modern Metrics Size, UMMFP – Unadjusted Modern Metrics Function Points, MMCAF – Modern Metrics Complexity Adjustment Factor.

Figure 1

MM Architecture

4.2 *Functional Units with Metrics and its Values*

In MM, based on the importance, the eight functional units are ordered accordingly. The effort required to complete each functional unit, it is a base factor to classify the effort levels of each functional unit as Low, Average, High and Very High. The rigid standard values, size of application, influence of functional units in the function, period of effort required in the growth of a functional unit; the metrics are categorized based on the industrial projects in Table 1.

Table 1

Functional Units with Metrics

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

The metric values of functional units of MM are categorized in Table 2.

Table 2

Metrics and 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

4.3 *Calculating Functional Units of a Software Application using MM*

The software application must be analyzed module-wise and list out the count of all the functional units present in each function of the application. The values are arranged in Table 3.

Table 3

Calculating Functional Units

S. No	Name of the Function	EI	II	EO	IO	DT	EQ	ILF	EIF
1									
2									
Total number of Functions referred (TF)									
Total Functional Units (TFU)									

The above table is used for tabulating and finding all the functional units in an application, the total number of functions referred to for each functional unit, and the total count for each functional unit in a category. The entire functions is referred to in the count for the number of functions associated with each functional unit. The total functional units are the total number of each functional unit present in the entire application.

4.4 *Complexity Adjustment Factor of a Software using MM*

The rate of impact of the complexity factors in the application is estimated using the complexity influence factors and their values. The following are the most influential factors and their values: 0-Nil, 1-Secondary, 2-Moderate, 3-Average, 4-Important, 5-Essential.

The list of complexity adjustment factors is, The backup is essential for the system; Data communication in-between the systems is essential; The system is running under distributed processing; The system has

a complex architecture; The system is working in a multi-topological environment; The system requires parallel updating; The application has connected input, output, and operations; The system requires any file on online updating; The system operates in a multi-platform environment on a variety of devices; The system's internal operations are more scientific, highly complex and critical; The codes are reusable; The software is extensible for versions and services; The software is generic and applicable to different organizations with some customized changes; The system uses indexed, tabled, comma-separated, unstructured, and semi-structured data; The system permits dynamic user interactions to change the results; The application has complex SDLC models; The application uses about one programming language, DBMS, web tools, drivers, etc.; The system uses multiple topologies for network architecture for distinct devices; The system is used in diverse countries and also accepts different social values and laws; The application gives complex types of output; The experimental and classical versions of the application affects the progress of the application; The user interfaces are influencing the application.

All the complexity metrics are identified and valued using influential factors. The Modern Metrics Complexity Adjustment Factor (MMCAF) is identified using the formula,

$$MMCAF = 0.25 * \Sigma_1^{22} CFi \tag{1}$$

The CFi (where i = 1 to 22) is the quantity of effect and involvement of CF in the software based on the responses received from the Complexity Adjustment Factors (CAF). The MMCAF is not exceeding 35%.

4.5 *Unadjusted Modern Metrics Function Points*

The Unadjusted Modern Metrics Function Points (UMMFP) is the sum of functional values existing in the application derived based on the number of functional values present in each function of the system. The calculation of UMMFP uses the following Table 4.

Table 4

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							
2							
Total UMMFP							

The UMMFP is the product of the total number of functions (TF) and metric value (W). The average functional units (AFU) are the ratio of total functional units (TFU) and the total number of functions (TF). The weight or matric value is based on the number of functional units present in AVF for each functional unit for the entire system. In MM, the average value of the entire system is calculated instead of each function or module. The calculation of function points with each function and module is also possible with MM.

The UMMFP is calculated based on the parameters in table 4. It is one of the parameters for finding the size of the software.

4.6 *MMSize*

The MMSize, or Modern Metric Size, is the actual size of the application based on MM function points (MMFP). The MMSize is the product of unadjusted modern metrics function points (UMMFP) and modern metrics complexity adjustment factors (MMCAF). That is,

$$\text{MMSize} = \text{MMCAF} * \text{UMMFP} \quad (2)$$

The MMSize not only considers the functional values present in the software but also considers the complexity of the software in all the possible technologies. So, the accuracy of the software is increasing compared with traditional FP methods.

4.7 *Advantages of MM*

The highlighted advantages of MM are: The MM technique is a new method to find the software size in any phase of the SDLC and also finding the size after the completion of the software development; The MM technique finds the effort required for the development of the functional units in an individual and collective manner. So, the defects

present in the FP count are negligible; The complexity adjustment factors and effort metrics are modified and updated based on the requirements of the modern software system.

4.8 *Future Developments*

The future enhancements in MM are: The MM technique was developed based on Indian software industry values. So, the analysis must be done using international values; The modern software industry is using a greater number of functional units with a function. So, it is facing difficulties in fixing the limits of the very high metric of functional values.

5. Discussion and Conclusion

The modern software industry is waiting for an effective size estimator with updated parameters for solving software management-based technical, economic, and social issues. The proposed MM analyses all the issues associated with the software from a user and developer perspective. It gives the feature of estimating the size of the software from any of its SDLC phases. The earlier estimation helps the software industry to do the development process with the derived time, cost, and effort. It increases the efficiency of delivery. The analysis-based work of the MM is progressive with the industrial parameters.

References

[1] Capers Jones, "Applied Software Measurement-Global Analysis of Productivity and Quality", Tata McGraw Hill, Third Edition, pp 71-182 (2008).

[2] Richard D. Stutzke, "Estimating Software-Intensive Systems: Projects, Products and processes", *SEI Series in Software Engineering*, Addison Wesley Edition, pp 1-786 (2005).

[3] Capers Jones, "Estimating Software Costs: Bringing Realism to Estimating", Tata McGraw Hill, Second Edition, pp 3-629 (2007).

[4] Capers Jones, "Software Engineering Best Practices: Lessons from Successful projects in the top companies", Tata McGraw Hill Edition, pp 1- 643 (2010).

[5] Robert t. Futrell, Donald F. Shafer, Linda I. Shafer, "*Quality software project management*", Pearson Education, pp 1-600 (2008).

[6] MehwishNasir, H. Farooq Ahmad, "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, *IEEE* (2006).

[7] Kenneth Lind, RogardtHeldal, "Estimation of Real-Time Software Code Size using COSMIC FSM", *IEEE International Symposium on Object/Component/ Service-Oriented Real-Time Distributed Computing*, pp.244-248 (2009).

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

[9] Ursula Passing, Martin Shepperd, "An experiment on software project size and effort estimation", *International Symposium on Empirical Software Engineering* (ISESE'03), IEEE (2003).

[10] Daniel V. Ferens, "Software size estimation techniques", Air Force Institute of Technology (AFIT/ISY), Wriqht-Patterson AEB, Ohio 45433, pp 701-706.

[11] Kjetil Molokken and Magne Jorgensen, "A Review of Surveys on Software Effort Estimation", *International Symposium on Empirical Software Engineering* (ISESE'03), IEEE (2003).

[12] Z. Zia, A. Rashid, K. uzZaman, Software cost estimation for component based fourth-generation-language, IET Software, pp.103-110 (2010).

[13] Md.Forhad Rabbi, Shailendra Natraj, Olorisade Babatunde Kazeem, "Evaluation of convertibility issues between IFPUG and COSMIC function points", *Fourth International Conference on Software Engineering Advances*, IEEE, pp.277-281 (2009).

[14] John T Mesia Dhas, C.R. Bharathi," Relative Analysis of Sizing Methods in the sense of E-Commerce system", *International Journal of Applied Engineering Research* ISSN 0973-4562 Volume 10, Number 18, pp. 39808-39816 (2015).

[15] John T Mesia Dhas, C.R. Bharathi, "Risks Associated to Size Estimation of E-Commerce System using Function Point based Estimation Techniques", *Indian Journal of Science and Technology*, Vol 9(7), (February 2016). DOI: 10.17485/ijst/2016/v9i7/85148.

[16] John T Mesia Dhas, C.R. Bharathi, "E-Commerce System Size using User Based Function Points", *International Journal of Applied Engineering Research* ISSN 0973-4562 Volume 12, Number 16, pp. 6115-6122 (2017).

[17] John T Mesia Dhas, C.R. Bharathi, "Modern Metrics (MM): Size Estimation of Modern Software and its Metrics Analysis", *Journal of Web Engineering*, Vol. 17, No. 6, pp. 1851-1861 (2018).

[18] John T Mesia Dhas, Midhunchakravarthy, "The Functional and Storage Risks Associated to the Size Estimation of Parallel Computing Applications", *Advances in Parallel Computing*, ISSN 1879808X, Volume 41, doi:10.3233/APC220052, pp. 373-379 (November 2022).