MODERN METRICS (MM):

The Functional Size Estimator for Modern Software

Dr. John T Mesia Dhas

MODERN METRICS (MM): The Functional Size Estimator for Modern Software

Dr. John T Mesia Dhas

Title: Modern Metrics (MM): The Functional Size Estimator for Modern Software

Author: Dr. John T Mesia Dhas

Publisher: Self-published by Dr. John T Mesia Dhas

Address of Publisher: No-1, MGR Street, Charles Nagar, Pattabiram

Chennai – 600072

India

Email: jtmdhasres@gmail.com

Printer: The Palm

Gangai Amman Nagar, Mogappair West

Chennai -600037

India

ISBN: 978-93-5408-510-9

ABSTRACT

The modern software system is programming language independent, operating system neutral, highly extensible and dynamic. About fifteen distinct programming languages, operating system, development tools and utility software are used for developing a new software system. The existing particularistic approached software sizing techniques are not efficient 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 MM analyses all possible functional units and complexity factors of modern software. So, the defects present in the existing Function Point Analysis (FPA) are 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 and FPA with Software Project Management (SPM) metrics like size, effort, cost and time implies, MM is more accurate than existing FPA and it is a suitable approach for calculating the size of modern software system.

The proposed MM method is a successful approach to determine the size of modern software system and it leads to the success of project management activities of modern software system development.

ACKNOWLEDGEMENT

First and foremost, I express my deep debt of gratitude to the Founder Chancellor & President **Col. Prof. Dr. Vel. R. Rangarajan** and Foundress President **Dr. Mrs. Sagunthala Rangarajan** for their immense contribution in making this organization grow and providing me the state of the art facilities to do this research work.

My heartfelt gratefulness to Chairperson & Managing Trustee **Dr. Mrs. Rangarajan Mahalakshmi Kishore** and Vice President **Mr. K. V. D. Kishore Kumar** for their deep commitment and dedication, to bring this institution to the peak in terms of discipline and values.

I would like to acknowledge my boundless feelings of thankfulness to the Chancellor **Dr. Beela Satyanarayana** for his visionary guidelines on this research work.

I voice my great pleasure and happiness to the respected Vice Chancellor **Dr. V. S. S. Kumar** for his in depth contribution for this research work.

Above all, my successful completion could not have been accomplished without our Registrar **Dr. E. Kannan** for his generous output and exemplary leadership in guiding my research in all aspects.

I owe my full satisfaction in my research work to the extensive hands of cooperation of **Dr. Anne Koteswara Rao**, Director Academics.

I would like to express my deep appreciation and sincere gratitude to my research supervisor, **Dr. C. R. Bharathi**, Associate Professor, Department of Electronics and Communication Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India for her guidance, support, encouragement, understanding and patience. I have been honored to work under her supervision and learn from her advice and useful insights throughout my Ph.D programme.

I extend my warm feelings of gratitude to **Dr. S. Koteeswaran**, Dean (Research Studies) for his support and help with respect to my research work.

I warmly thank my Doctoral Committee member, **Dr. P. Sakthivel**, Associate Professor, Department of Electronics and Communication Engineering, Anna University, Chennai for his valuable advice and extensive discussions about my work.

I extend my gratitude to **Dr. N. Malarvizhi,** Professor and Head, Department of Computer Science and Engineering for her constant support to complete the research work.

I express my sincere thanks to all the Professors and non-teaching staffs of **Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai** who cooperated with me which helps me to realize my dream.

I express my sincere thanks to the **Chairman, Trustees, Principal, Professors and non-teaching staffs of Audisankara College of Engineering and Technology, Gudur, Andhra Pradesh** who cooperated with me in all climates for successful completion of this book.

I convey my special gratitude to my beloved family for their loving support and encouragement which enables me to complete this research work successfully.

JOHN T MESIA DHAS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	ABBREVIATIONS
ADT	Average functional units of Data and Text
AEI	Average functional units of External Inputs
AEQ	Average functional units of External Inquiries
AEIF	Average functional units of External Interface Files
AEO	Average functional units of External Outputs
AI	Artificial Intelligence
AII	Average functional units of Internal Inputs
AILF	Average functional units of Internal Logical Files
AIO	Average functional units of Internal Operations
COSMIC	Common Software Measurement International Consortium
CAF	Complexity Adjustment Factors
COCOMO	Constructive Cost Model
DBMS	Data Base Management System
DT	Data and Text
EI	External Inputs
EQ	External Inquiries
EIF	External Interface Files
EO	External Outputs
FPA	Function Point Analysis
FP	Function Points
FDT	Functions in Data and Text
FEI	Functions in External Input
FEQ	Functions in External Inquiries
FEIF	Functions in External Interface Files
FEO	Functions in External Outputs
FII	Functions in Internal Inputs
FILF	Functions in Internal Logical Files
FIO	Functions in Internal Operations
GUI	Graphical User Interface
II	Internal Inputs
ILF	Internal Logical Files

ACRONYM	ABBREVIATIONS
IO	Internal Operations
IFPUG	International Function Point User Group
ISO	International Standard Organization
ISPA	International Society of Parametric Analysis
IT	Information Technology
LOP	Learning Object Points
LOC	Lines of Code
MFP	Modern Function Points
MIS	Management Information System
MM	Modern Metrics
MMCAF	Modern Metrics Complexity Adjustment Factors
MMC	Modern Metrics Cost
MMD	Modern Metrics Duration
MME	Modern Metrics Effort
MMPF	Modern Metrics Productivity Factor
MMSize	Modern Metrics Size
NESMA	Netherlands Software Metrics Association
SLIM	Software Life cycle Management
SPR	Software Productivity Research
SPM	Software Project Management
SDLC	System Development Life Cycle
UDT	Unadjusted Data and Text
UEI	Unadjusted External Inputs
UEQ	Unadjusted External Inquiries
UEIF	Unadjusted External Interface Files
UEO	Unadjusted External Outputs
UII	Unadjusted Internal Inputs
UILF	Unadjusted Internal Logical Files
UIO	Unadjusted Internal Operations
UMMFP	Unadjusted Modern Metrics Function Points
UML	Unified Modelling Language
UCP	Use Case Points
WDT	Weightage of Data and Text

ACRONYM	ABBREVIATIONS
WEI	Weightage of External Inputs
WEQ	Weightage of External Inquiries
WEIF	Weightage of External Interface Files
WEO	Weightage of External Outputs
WII	Weightage of Internal Inputs
WILF	Weightage of Internal Logical Files
WIO	Weightage of Internal Operations
WWW	World Wide Web

LIST OF SYMBOLS

SYMBOL	MEANING
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Percentage
=	Assignment
>	Greater than
&	Address of
∑	Sum of

CHAPTER 1

SOFTWARE PROJECT MANAGEMENT

The Software Project Management (SPM) is one of the fields of Computer Science under Software Engineering. It is a management process which leads planning, designing, implementing, testing, sizing, monitoring and controlling the software and software development process. The perfect initial planning is a key for success of completion and quality development of the software system. The project planning is an initial process for all software management activities. The systematic planning gives all the parameters of software development like actual size of the system, effort and skills required for system development, technology and hypotheses used for decision making, proper schedule of system development and price of software. The perfect plan leads delivery of the product on its predicted time (Capers 2008).

The software industry is using distinct sizing techniques for determining the size of the software as Lines of Code, Expert Judgement, Constructive Cost Model (COCOMO), Function Points, Future Points, Object Points, etc. These techniques are giving distinct results for same software. Based on programming language, type of application, estimator and technique used for estimation the result vary accordingly. So, the effectiveness of size estimation techniques for modern software is complex and critical (Capers 2007).

The modern software system includes all the applications like, Webpages, networks, internet, database, Artificial Intelligence (AI) , designing, modelling, animation and expert systems. It is particularistic in domain and dynamic in behaviour. The existing software sizing techniques are not efficient for determining the actual size of complex modern software system. The incorrect sizing of software system affects quality of the software, customer satisfaction and System Development Life Cycle (SDLC). The improper size of the software leads delay in completion of the project and increases the development cost of the software.

The Modern Metrics (MM) is a novel software size estimation technique for modern software using modern dynamic function points. The MM is independent of programming language, development tool, operating system, database and all other internal, external factors of system development process. This proposed size estimation technique considers the user, developer and social perspectives of software system. Therefore, the defects and wrong estimations in the size of modern software system are resolved through this proposed MM technique.

The software size is a key factor for determining all planning activities of software development process (Kenneth and Rogardt 2009). The modern software is a merger of software and other Engineering disciplines. It includes all application programs, embedded systems, data mining, data warehousing and big data, AI, enterprise resource planning, service oriented architecture, E-Commerce, design, modelling and animation. This dynamic behaviour of modern software system leads confusion in size estimation using existing software size estimation techniques. The imperfect size estimation generates crisis in software development process of modern software. This proposed novel technique called MM which gives new strategy for determining size of modern software.

MM is an Indian metrics, which is used to find size and complexity of software in analysis phase of SDLC. The MM is determining size of software based on user and developer views in Function Point Analysis (FPA) and International standards. MM is an ad hoc method for measuring the size of modern software system irrespective of its programming language, methodology, organisation and other physical parameters. The MM gives a successful way of measuring size of modern software.

1.1 SOFTWARE DEVELOPMENT PLANNING

The success factors for software are faster in development, cheaper in cost and better in quality. The success of software development process depends on good planning and dynamic management (Cigdem et al 2009). The systematic software development process follows analysis, design, coding, testing and maintenance phases in sequential, concurrent or divide and conquers fashion. The analysis phase captures all requirements, then construct initial business model and finalize plan to develop software project. The process to plan a project starts with an assessment of the constraints that affects the project (Barry 1981). It requires a delivery date, overall budget, staff, etc. These requirements are carried out by combining the project parameters like its structure, size and distribution of functions.

The following algorithm shows the sequence of steps followed for project planning (Galorath and Evans 2006).

To identify the project requirements.

i. To do the feasibility study of the project variants.
ii. Explain all the intermediate steps and outcomes of the project.
iii. The following loop is executed until the project is completed.
    a. Define the schedule of the project.
    b. Perform the activities based on schedule.
    c. To check the progress of the project.
    d. Update the parameters of project.
    e. Revise the schedule.
    f. To check with requirements and outcomes.
    g. If (not an actual solution) then

        Start the technical reviews.

      End if

    End Loop

The above stated project planning algorithm gives the importance of initial assessment of project parameters, which are used for setting realistic targets towards project delivery. The failure of many large software projects highlights the problem of poor planning and estimation of project parameters (Robert et al 2008).

1.1.1 Activities during Software Planning

The major activities in a project planning stages are assessing or estimating project parameters, resources capturing and project scheduling (Mehwish and Farooq 2006). The Figure 1.1 shows these activities in detail.

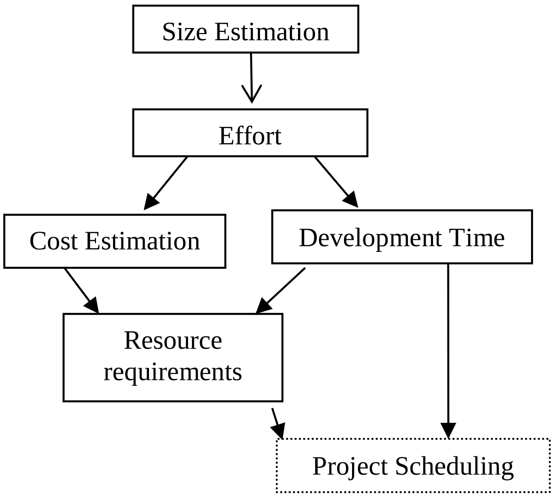


Figure 1.1: Major Activities in Project Planning

Estimation is the process of goal setting, which forms the basis of quantifying the resources to accomplish certain goals based on the clearly identified assumptions (Henry 2008). Size estimation is the predetermination of the size of final work product. Size is the basic measure to calculate other project factors.

1.1.2 Specific Quantities to Estimate and Measure during the Life Cycle of Project

The COCOMO of Software Engineering Institute developed for software systems is recorded that the following quantities to be measured during the lifecycle of the project (Angelica 2004).

- Effort(Events)
- Staff(Count, Expertise and Knowledge, Business)
- Time(Period, Agenda, Progress)
- Costs(Workforce)
- Hardware and software resources used for development and test
- Performance (Ability, Correctness, Speed, Time)
- Quality (Conformance to necessities, Reliability, Security, Data Veracity)
- Price and total proprietorship cost.
- Size or Amount (Generated, Altered, Acquired)

The primary quantity of the list mentioned above is size. It is directly or indirectly employed with other measures of software development process. The software industry has a lot of software sizing methods and techniques. The methods are giving ways to do the estimates, whereas the techniques state the procedure to do the estimates of a particular method. During project planning, the above said parameters are quantified other than performance and quality.

1.2 ABOUT SIZING APPROACHES

Software size is a key factor in determining the quantity of time, cost and effort that are needed to develop software systems. The success of any software project mostly depends on the efficient estimation of project effort, cost, and time. Estimation helps a software developer in setting realistic targets for completing the project in a successful way (Mehwish and Farooq 2006).The software industry uses various sizing techniques. They are Lines of code, Function points, Feature points, Use case points, Object points, Internet points; expert based Expert judgment, estimation by analogy, Delphi technique, etc. (Richard 2005). These techniques do not effectively support to determine the size of Modern Software system and leads to affect all the estimates. The wrong estimates lead imperfectness, loss and customer dissatisfaction.

1.3 MODERN SOFTWARE SYSTEM SIZING TECHNIQUES

Innovation in software technology has tremendously shaped our modern human life at every place. Every day, the new software technologies are emerging and millions of software is developed. The modern technologies are giving abundant to the people for their fertile living. In this digitalized living environment, software and internet are playing vital role in the dynamic face of the world.

The calculating machine is enhanced into governance machine. Millions of people are working with Information Technology (IT) and IT enabled services. Regulations, standardization and authentication are required in this field for harmonious growth of software industry. Many organizations and protocols are available for monitoring those things. But the size estimation of the software is one of the challenging issues in software industry.

Many empirical methods are available to measure the size of the software. But, it denies giving actual size of modern versatile software applications (Abran 2006). The existing sizing methods and its metrics are not sufficient for finding actual size of modern software like web based-database linked-multi environmental-multi faced- application systems, embedded system, grid and cloud computing, data mining and data warehousing, big data, scientific and AI, enterprise resource planning, service oriented architecture, design, modeling, simulation and E-commerce systems. Because, the modern software is the amalgamation of software and other Engineering disciplines. So, a multipurpose technology is needed for calculating size of software.

The FPA is a sizing technique which is independent of programming language, development tools, or software development lifecycle methods used for application (Erika 2012). To uplift the functional values of FPA will give actual size of modern software. This is a new technique to measure the size of the modern software known as MM.

MM, is an Indian Metrics (IM) which gives size of modern software through some basic calculations based on Modern Function Points (MFP). All the functional parameters are analyzed based on user and developer perspectives. The cost, size and time are rationally less to the traditional FPA and it is very simple to calculate.

1.3.1 Architecture of Modern Software System

Modern Software system is a task based analytical package. It gives solutions to living and nonliving, scientific and super natural, practical and theoretical, movable and immovable, dynamic and static, variable and constant, wisdom and folly, imaginable and unimaginable, etc. The Figure 1.2 shows the major domains of modern software system.

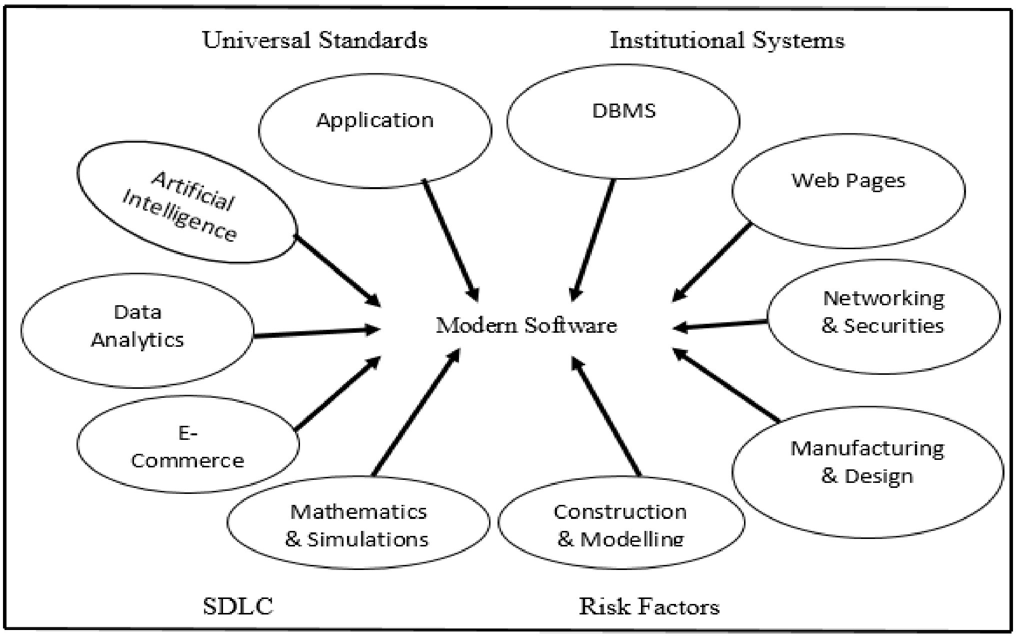


Figure 1.2: Major Domains of Modern Software System

The modern software system is not a single domain application; it is the combination of more than one domain. The web servers and internet facilities developed the efficiency of the software and increased boundary nil services. The software are used in creating applications, Data Base Management System (DBMS) packages, websites and services, networking and internet services, AI, Data Analytics (text, knowledge, scientific and etc.), E-Commerce, Mathematics and Simulations, Construction and Modeling, Manufacturing and Design, Training and Sharing, etc. In the modern world, all the applications must follow some universal standards, institutional system codes, SDLC, social, economic and political codes (Ferchichi 2006).

The market for modern software system shows a tremendous growth every year. But this growth rate varies based on the countries, which are economically and technically developed and developing. India is showing an enormous growth in software development and IT based services.

1.3.2 Necessities for a Sizing Approach in Modern Software System

The Standish Group (Lynch 2009) states that 44% of IT projects were delivered late and over budgeted. It indicates that the role of project management has become increasingly more important (Demirors and Gencel 2004). The International Society of Parametric Analysis (ISPA) identified the main reasons behind project failures (Eck et al 2009). These reasons can be summarized as follows:

i. Lack of valuation of the workforce’s talent level
ii. Lack of understanding the necessities
iii. Improper software size valuation

On the whole, many software projects failed because of the inaccuracy of software estimation and misunderstanding in requirement gathering from the customer or incompleteness of the requirements. These motivated researchers conduct research on software estimation for better software size and effort assessment. One of the initial stages of project management activity is planning. In this stage, the software developers perform the software size, effort estimation; calculate the budget, schedule and the number of people required for developing the software.

Modern software system development is also under crisis because of the unavailability of appropriate sizing technique (Filip 2007). It leads to improper size estimation, which affects the project planning process. Improper planning affects project management in all stages, and that leads customer dissatisfaction, which affects the goodwill of the organization. So, the software industries need an appropriate early stage sizing approach for estimating the size of modern software system. This proposed work introduces MM approach to the world for estimating the size of modern software system. It resolves the problems faced during modern software system development.

1.7 SUMMARY

Software size estimation is one of the most important phases in the software project management. To estimate the size of the software at the time of project planning gives good budgeting and delivering. The modern software system is an amalgamation of applications, Database Management Systems, web pages, networking and its securities, manufacturing and designing, construction and modeling, mathematics and simulations, E-commerce, data analytics and artificial intelligence.

So, the existing techniques are not opt for finding the actual size of the modern software.

A novel method, Modern Metrics is proposed for finding the size of the modern dynamic software system. It will overcome all the pitfalls of existing software size estimation techniques.

CHAPTER 2

SOFTWARE SIZE ESTIMATION TECHNIQUES

To analyze the productivity of software and developing team; it is a major issue for International Software Engineering research community. Because of the size of software, it is playing a great role in the estimation of productivity. Many researches are carried over by different scholars in different time and environment for estimating the size of software. They developed many innovative techniques and published. In it, some are domain centric others are generalized for software size estimation. The "literature review", in this chapter is presented in three stages based on their applications. All existing sizing techniques and their determined capabilities are reviewed in first stage. In the second stage, the significance of software size estimation and the software project planning activities such as effort, time and cost is observed. The third stage of literature review highlights on modern software system and its application domains.

2.1 SOFTWARE SIZING TECHNIQUES

After the Second World War, in-between 1945 and 1955, the first generation computers were emerged for doing the scientific calculations. The programs developed at this time are mainly in machine language and some of the programs in assembly language. The size of the software was few hundreds to some thousands of lines of code. The format and style of all the programs were same. Hence, lines of code were the main factor for determining the size of software. According to Capers Jones (2007), productivity and quality are measured based on lines of code. After 1950's some powerful procedure oriented languages like FORTRAN, COBOL and BASIC emerged. It has changed the history of computer science. These high level languages replaced machine language and assembly language bringing in the changes in software development field. File formats and syntaxes varied from one programming language to another. The lines of code varied from one developer to another and one language to another. Finally, at the end of 1950's, the lines of code are concluded as not the apt method for estimating the size of software.

To overcome the problems of lines of code, the Expert Judgment technique was proposed by Helmer of RAND Corporation. In this method, the size is estimated based on the views of an expert or group of experts (Richard 2005). The expert judgment method is good for scientific and analytical applications. The end user perspectives are not deliberated in this estimation technique.

In 1969, the Software Science Metric was developed by Halsted. It calculates the size of software based on number of operands and operators present in the application. It is good for mathematical and scientific applications. It is not good for general purpose software programs (Capers 2010).

After 1960's, computers entered into the commercial market like banking, manufacturing, etc. It has increased number of lines from few thousands to some millions. But quality and reliability of the software increased. Around 30% of effort and time is reduced in the development process; but more than 40% of effort and time increased in debugging and testing process (Capers 2008). A new engineering study for computer emerged in the entire world. Software developers and industries increased. Software industry turned into different fields like research, applications and entertainment. Thousands of new applications were developing all over the world with different programming languages. This, multi-faced environment challenged many irremovable factors like time, quality, size, cost, etc. To solve these industrial problems, many new proposals and solutions emerged in the industry. IBM was one of the leading industries that tried to solve the issues.

Allan Albrecht (Capers 2010), a well-known IBM researcher introduced a new metric for measuring the size of software known as function points in 1979. It has five functional units, they are, inputs, outputs, inquiries, logical files and interfaces. It allowed the interaction of user with the system at the development process itself.

To do the sizing process of software, many new and innovative techniques are emerged in general purpose and special purpose manner. They are classified in three different categories as: code based, function based and expert based (Gustavo 2011).

The lines of code and Halsted's Software Science are important code based techniques. In these methods, number of lines of source code is the key factor for determining the size of software. It was good for first generation programming languages.

The methods like Expert Judgment, Delphi, Pattern Matching, Linear Method and Estimation by analogy are important expert based techniques

(Hughes 1996). An expert or a group of experts will determine the size of the software. The interaction of user is restricted in this method.

Function Points, Feature Points, Use Case Points, Object Points, Internet Points, Common Software Measurement International Consortium (COSMIC), Backfiring Function Points, 3D Function Points, De Marco Function Points, Function Point Light, Full Function Points, International Function Point User Group (IFPUG) Function Points, Netherlands Software Metrics Users Association (NESMA), Total Metrics (Australian Metrics), Web Object Points and Story Points are the important function point based generalized and particularistic software size estimation techniques (Gopalaswamy 2013). These methods gave importance to the user in the software development process.

2.2 SOFTWARE PROJECT MANAGEMENT (SPM) ACTIVITIES

Mehwish and Farooq (2006) studied the concepts of software cost, effort and size estimation and suggested that the existing techniques are not giving 100% accuracy in estimation. But the proper way of estimation using the existing methods must increase some accuracy in measurement. The size estimation process and other findings like cost, effort, skill and time must be derived at the analysis phase. Then only the software will be developed on its allocated time period. The existing methods are not good for analysis phase size estimation.

Mahir Kaya et al (2011), signifies, “software size” is essential for estimating cost and effort of the system. Therefore, earlier the estimation of size and earlier the increase in efficiency of software management.

Daniel et al (1999) studied various issues of software size estimation and suggests that single method is not good for estimating software. Existing size estimation techniques are domain centric. It is not considering environment and social issues of the developing unit. So, innovative techniques must be developed for size estimation of software product.

Barry (1986) states that, “The biggest difficulty in using today’s algorithmic software cost models is the problem of providing sound size estimates”. That means parameters and metrics are not sufficient for doing the software sizing.

Zia et al (2010), in the study on Graphical User Interface (GUI) applications states, the current estimation techniques are not having the metrics to measure component based software applications and the existing methods produce wrong results in the estimation. So, new methods required for component based software applications.

Forhad Rabbi et al(2009), in the study on function point size estimation techniques suggests, software industry is not young, it is matured, it extends its wings to all the sectors. So, the existing standard FPA methods like ISO 19761: COSMIC FPA (2003), ISO 20968: Mk II (2002), ISO 20926: IFPUG 4.1 FPA (2003) and ISO 24570: NESMA (2005) are not good for modern software.

Edilson and Rosely (2003), suggests that the key factor determining the cost, time and effort is size of the software. Linda (2006) states, in software, effort, schedule and cost estimated based on size of software. The LOC and FPA based methods are not sufficient for measuring actual size of the modern software. Steven Fraser et al (2009) suggested good and poor estimation of size is affecting quality, cost, time and reliability.

Daniel (1999), in his study he specifies, many methods are available in the industry for measuring the size of the software but till now the accuracy in estimation is not giving by any existing methods.

Iman and Siew (2009) said, early stage of size estimation is the high performance of software size estimation. But the existing techniques are not doing it well.

The comparison of various sizing techniques and its features are listed in the Table 2.1(June 1992).

Table 2.1: Sizing Techniques and its Features

Features	**Sizing Techniques**					
	LOC	**FPA**	**Feature Point**	**Use Case Point**	**Object Point**	**Internet Points**
Inputs and Outputs	No	Important Functional Units	Important raw Feature Point	Actor interaction Points	No	No
Logical and Interface Files	Counts the lines of code	Important Functional Unit	Important raw Feature Point	Not considers all logical files	All the logical files are considered	Considered in the form of hyperlinks
Web Pages	Considers the lines of code	No	No	No	No	Yes
GUI	No	Outputs only	No	No	Outputs only	No
Multimedia	No	No	No	No	No	Considers in an external file
Graphics	No	No	No	No	No	Yes
Reusability	No	Yes	No	No	Yes	No
Text	Considers number of lines	No	No	No	No	Considers number of lines
DBMS Support	No	No	Yes	No	No	No
Data Communication	No	Supports in Complexity Adjustment Factor	No	No	No	No
Internet and Securities	No	No	A little	No	No	Yes

The popular effort and cost estimation models are COCOMO (Barry 1981), Software Lifecycle Management Model (Putnam 1978), Function Point, Use Case Points (Karner 1993) and SEER-SEM (Galorath and Evans 2006). The Delphi technique is used to provide communication and cooperation among the experts (Dalkey and Helmer 1963). These models also used the size as the base factor.

From the above study, software size estimation is highly essential for software development process and the existing methods are not sufficient for measuring the software size perfectly.

2.3 MODERN SOFTWARE SYSTEM

The machine language codes of computer software are changed into high level language codes. The high level language codes are changed into object oriented language codes. The object oriented language codes are changed into GUI applications. The GUI applications are merged with network, internet and DBMS to form a new system known as modern software.

The evaluations of modern software based on its technological units are listed in the Table 2.2.

Table 2.2: Evaluation of Modern Software

S. No	Technological Unit	1980-1990	1990-2000	2000-2010	2010-till now
1	Procedure Oriented	Very High	High	Less	Very Less
2	Object Oriented	Less	Medium	High	Very High
3	Networking Support	Less	Medium	High	Very High
4	World Wide Web Support	Less	Medium	High	Very High
5	File Handling	Very High	High	Medium	Medium
6	DBMS	Very Less	Medium	High	Very High
7	GUI	Very Less	Medium	High	Very High
8	Object Linking and Embedding	Very Less	Less	Medium	High
9	Heterogeneous Environment	Very Less	Less	Medium	High
10	Distributed Computing	Very Less	Less	Medium	High
11	Parallel Computing	Very Less	Medium	High	Very High
12	Cloud Computing	Very Less	Less	Medium	High
13	Knowledge Based (Big Data, Data Mining, etc.)	Very Less	Less	Medium	High
14	External Input	High	Medium	Less	Very Less
15	External Output	High	Medium	Less	Very Less
16	External Inquiry	High	Medium	Less	Less
17	Internal Logical Files	Less	Medium	High	High

S. No	Technological Unit	1980-1990	1990-2000	2000-2010	2010-till now
18	External Interface Files	High	High	Medium	Medium
19	Internal Input	Less	Medium	High	High
20	Internal Operations	Less	Medium	High	Very High
21	Indexed Data	Less	Medium	High	High
22	Multiple form of Outputs	Less	Medium	High	High
23	Multi-valued Function Points	Less	Less	Medium	High
24	Dependent Function Points	Less	Less	Medium	Medium
25	Composite Function Points	Less	Less	Medium	Medium
26	Service Oriented Architecture	Less	Less	Medium	Very High
27	Enterprise Resource Planning	Less	Less	Medium	Very High
28	AI	Less	Less	Medium	Very High
29	Data Analytics	Less	Less	Medium	Very High
30	Standardization	Less	Less	Medium	Very High

The above study in Table 2.2 explains that the modern software is not a single unit, but it includes all the existing technologies of the modern world and the novel requirements of the end user.

MM, is an Indian Metrics which will give the actual size of modern software through modern function points. It analyses all the functional parameters based on user and developer perspectives. Its cost, size and time are rationally less to the traditional Function Points (FP). So, for calculating the size of modern software system, the existing popular sizing approaches are inefficient.

2.4 LIMITATIONS OF EXISTING SYSTEM: FUNCTION POINT ANALYSIS (FPA)

Based on the studies with the existing FPA methods, the following drawbacks are identified. They are,

1. The accuracy in function point calculation is very difficult for modern software. As on IFPUG study, defects per FP are 4.5.
2. Internal Operations like Multifaceted algorithms and heavy calculations that are portion of a transaction's processing rationality are not distinctly considered as portion of the functional sizing.
3. The choice based selection (e.g. if structure, case structure) does not get extra size.
4. FP merely reflects communications among the (external) user and the application. Communications between several internal portions of the application are not measured by the FP model.
5. Relocation of User Interface essentials without adding / removing / modifying some of them is not encompassed in the sizing method.
6. If the similar result is generated in several presentations or models (e.g. MS-Excel and PDF), no extra size is considered for the several models (i.e., only one model is comprised for the size estimation).
7. The database or text files does not present in the FP count.

8. The trial versions and model versions of the software does not present in the FP count. It is reducing the effort of the developer.
9. Internal Inputs, Indexed and List data is not getting importance in the FP count.
10. The Trivial Function Points like Multi Valued FP, Dependent FP and Composite FP are not present in the FPA calculations.
11. The cost for estimating FPA is high (estimation cost per function point is 4$ to 8$). Very large scale projects are not estimated using FPA method.
12. The CAF of the existing FPA must be updated.
    a) The indexed data, list values and choices must be considered and its influence must be calculated in CAF.
    b) The multiple forms of Outputs and its influences must be analyzed in the CAF.
    c) The number of Operating Systems, Programming Languages, DBMS, Web tools and drivers used in the system must be analyzed and to find out its influences.
    d) The various topologies, networks, servers and its software must be analyzed and measure the influence of it in the system.
    e) The various SDLC models must be analyzed and find the influence of it in the system.
    f) The political, economic and social condition of the nations which will be affected in the system must be analyzed.
    g) The influence of International Standards used in the system must be analyzed.

2.5 COMPARISON OF EXISTING SIZING TECHNIQUES

The comparison of all software size estimation techniques are present in the following Table 2.3.

Table 2.3: Comparison of Sizing Techniques

S. No	Sizing Approaches	Author and Year	Application Focus
1	Lines of Code	1950’s	Any Application but focusing on code
2	Expert Judgment	Helmer - 1959	Any kind of application but Expert centralized
3	Software Science Metric	Halstead M. H. - 1969	Scientific Application
4	Function Point Analysis	Allan Albrecht - 1979	MIS like business Applications
5	DeMarco “Bang” Function points	Tom DeMarco - 1982	System software, Scientific software
6	Mark II Function points	Charles Symons - 1983	System software
7	Backfiring Function points	Capers Jones - 1984	Mathematical conversion from source code statements to equivalent function points

S. No	Sizing Approaches	Author and Year	Application Focus
9	SPR Function points	Software Productivity Research - 1985	MIS like business Applications and is using Backfiring concept.
10	IFPUG Function points	International Function Point User Group - 1986	Business Applications. It is a regularized form of original function points developed by Albrecht of IBM
11	Feature Points	Alan J. Albrecht and his team - 1986	Real time systems
11	Engineering Function points	Donald Umholtz and Arthur Leitgeb -1994	Scientific Application
12	3D Function points	Scott Whitmire - 1994	Scientific and Real time software
13	Object Point method	Rajiv D.Banker - 1994	GUI based Applications
14	NESMA Function points	Netherlands Software Metrics Association - 1995	MIS like business Applications, Real time systems
15	Data point Method	1997	Database sizing
16	COSMIC Function points	Common software Measurement International consortium - 1998	Real time and Embedded software

S. No	Sizing Approaches	Author and Year	Application Focus
17	Story points	1999	Agile based software Development
18	Web object points	Donald Reifer - 2000	Web Applications
19	Use Case points	UML based software sizing approach introduced in 2003	Object Oriented Software
20	Function points ‘Light’	David Herron of David consulting group	MIS like business Applications

2.6 SUMMARY

The existing sizing techniques like Lines of Codes, Function Points, Feature Points, Use Case Points, Object Points, Internet Points and all other size estimation techniques are domain centric. But, the modern software is multi domain and it has complex architecture. So, a novel multifaceted and simplified sizing approach is required for finding the size of modern dynamic software system. The Function Point Analysis is an effective method for measuring the size of application software based on user perspectives. To update the existing FPA with some functional units, complexity adjustment factors, software metrics and values will give an effective approach for finding the size of modern software system.

CHAPTER 3

SIZING APPROACHES FOR MODERN SOFTWARE SYSTEM

Software size denotes the quantity of software. Software size, is a key factor in determining the amount of time and effort that is needed to develop software systems and the modern software system development also has no exception. The success of any software project largely depends on the effective estimation of these attributes (Juan 2010). Estimation helps in setting the realistic targets to complete the project. The basic element for estimating everything is size. Sizing is the prediction of coding that is needed to fulfill the requirements. Every object in the real world can be measured regarding some units. Software size is measured in terms of number of lines, counting functions, counting features, a number of pages of user documentation, etc. (Kjetil 2003). The software industry uses various sizing techniques. They are lines of code, function points, feature points, use case points, object points, internet points, etc. They do not support effectively to determine the size of Modern Software system which leads to inaccurate estimates. The inaccurate estimates lead to incompleteness, loss and customer dissatisfaction. This chapter presents the popular sizing techniques and their inabilities in sizing and also the necessities of new sizing approach for modern software system.

3.1 ESSENTIALS OF SIZING APPROACHES

The process of quantifying software is called software sizing. Sizing and estimation play a major role in software development, which leads to complete the project in good fashion (Kotonya 1998). Figure 3.1 illustrates the estimating principle for project management.

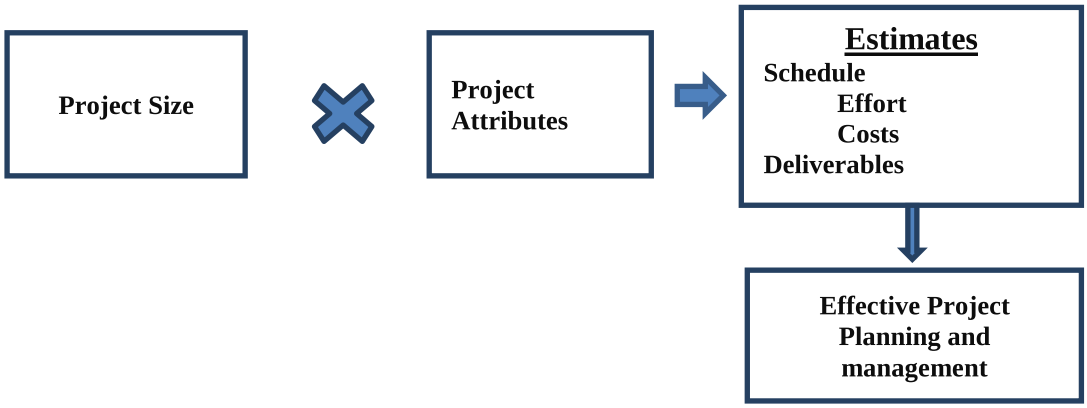


Figure 3.1: The Estimating Techniques for Project Management

The rate of software requirements may change depends on the following factors (Luigi 2011),

- The knowledge of the development group in similar applications.
- The process or methods used to develop the project.
- The programming language or languages utilized
- The presence or absence of reusable artefacts.
- To develop a project, whether the Development tools are used?

These attributes may orient to personal, technologies, tools or programming environment. By using size and attributes, effort, cost, schedule and other deliverables are estimated. These estimates are supported by effective planning and management of software projects. So, size and sizing approaches are essential.

3.2 BASIC PROCESS TO ESTIMATE THE SIZE

The following steps needed for estimating the size of software in a linear way (Richard 2005).

i. Define your size measure.
ii. Identify all items to be built.
iii. Estimate the size of each items using sizing approaches
iv. Add up sizes of each item.
v. Validate the results
vi. Repeat steps ii - v, if appropriate.

3.3 CHOOSING A SIZE MEASURE

While choosing or inventing a new sizing approach, the following characteristics should be considered. The characteristics of a good size measure are as follows (Humphrey 2004).

- It is correlated to the development effort expected by the engineers.
- It is autonomous of the knowledge used.
- It can be estimated at the beginning of the SDLC.
- The calculations must be simple.
- The user, developer and organisational perspectives must be present.

3.4 SIZING APPROACHES

Sizing approach denotes a method or technique, which is used to quantify the size of the software.

The sizing approaches are broadly classified into three categories. They are,

- Code based techniques
- Expert based techniques
- Function based techniques

These sizing approaches are represented in Figure 3.2.

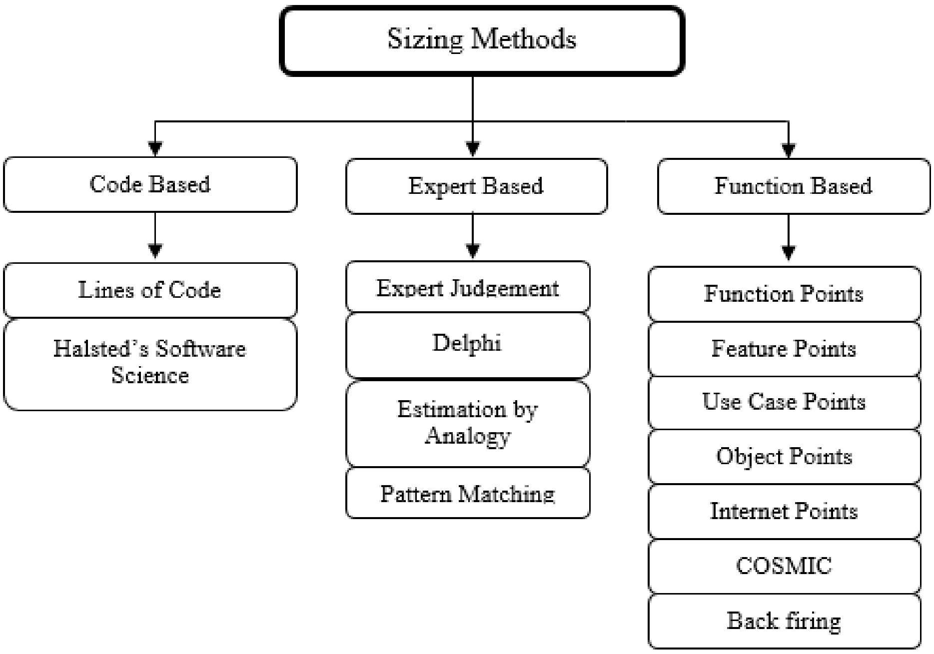


Figure 3.2: Classifications of Sizing Methods

The following section describes the popular sizing approaches in the Software Industry and their limitations in Sizing Modern Software system.

3.4.1 Code Based Techniques

Lines of Code

The Lines of Code (LOC) is used from the beginning stage of the evolution of programming languages. The main objective of LOC is to count each executable instruction including data definition and the size (Lavanya 2010).

Halstead's Software Science

The Software Science developed by Halstead attempts to estimate the programming effort (Luigi 2011). The measurable and countable properties are as follows:

- n_1 = number of unique or distinct operators that appear in that implementation
- n_2 = number of unique or distinct operands that appear in that implementation
- N_1 = total usage of all of the operators that appear in that implementation
- N_2 = total usage of all of the operands that appear in that implementation

From these, Halstead defines vocabulary length and other attributes. The vocabulary of the program is the summation of unique operators and unique operands. The formula for calculating vocabulary *n* is given in following Equation (3.1).

$$n = n1 + n2 \tag{3.1}$$

Similarly, the length of the program is the summation of the Total number of operators and total number of operands. The formula for calculating program length N is given by the following Equation (3.2).

$$N = N1 + N2 \tag{3.2}$$

3.4.2 Expert Based Techniques

Expert Judgment

Expert or group of experts uses their experience to understand the proposed project, and they make estimation (Najberg 1984). The original technique arose from work done at the RAND Corporation in 1950's and matured in the following decade. The following steps are used for estimation.

Steps

1. Coordinator gives each expert a requirement and a valuation form.
2. Coordinator organizes a group meeting in which the experts deliberate estimation problems with the coordinator and each other.
3. Experts fill out the forms anonymously.

4. Coordinator prepares and distributes the summary of the estimation on an iteration form.
5. Coordinator calls a group meeting to discuss the expert’s points, where the estimates varied widely.
6. Experts fill out the forms again anonymously and step 4 – 6 are repeated to get an appropriate conclusion.

Delphi Technique

Delphi cost estimation technique tries to overcome some of the short comings of the expert judgment. Using Delphi technique, the size and amount of effort that is required to perform the tasks are estimated properly. There are two Delphi versions. They are Narrow band Delphi and Wide band Delphi. In narrowband Delphi, estimators never meet. Every expert in the panel gives the opinion without discussing with other experts. In Wideband Delphi, estimators meet face to face. Every expert may discuss together and gives the opinion.

Estimating Size by Analogy

Based on the size of similar projects that is developed in the past helps to estimate the size of new software. For this estimation, historical data and experts are necessary. Sometimes scaling concept is also used. This kind of guessing not supported in modern software system sizing because of complex parameters (Noureldin 2010).

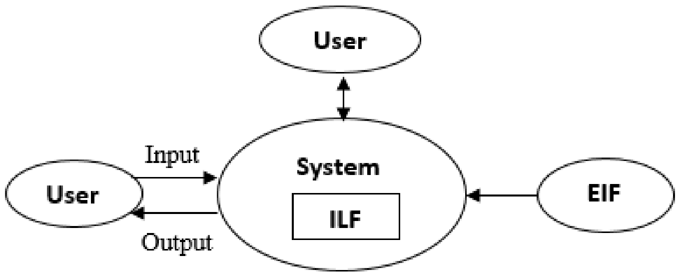
Pattern matching

The same analogy concept is used in functional size measurement called Pattern matching and function points. In pattern matching approach, the application to be sized is compared against the catalogue of historical projects and matched against similar projects. There are two critical topics requires for the pattern matching approach to be effective. They are the large collection of historical data and a formal taxonomy of software projects to guide the search. The taxonomy for pattern matching states that during pattern matching, elements like project nature, project scope, project class and project type in the function point approach has to be considered.

3.4.3 Function Based Techniques

FPA

FPA is the standard metrics for measuring functional size of a software system (Paul 2007). The function point was first defined by A.J. Albrecht at IBM in late 1970's. The FPA is used to predict the effort estimation of the software project in the beginning stage of the life cycle. It measures the complexity of the functions and overcomes the difficulties of Lines of Code. FPA helps the developers and users to quantify the size and complexity of software application functions in a way that is useful to software users. The diagrammatic representation of functional units of FPA is in the Figure 3.3



ILF – Internal Logical Files
EIF – External Interface Files

Figure 3.3: Functional Units of FPA

There are two types of functionality in FPA: The first one is data functions to count the size of the data part of the project and the second one is transactional functions to count the size of the transactional functions of the project.

Unadjusted Function Point - UFP

UFP- Unadjusted function point specifies the total number of function points depending on the following two factors. They are Data functions and Transaction functions. It means the counting of all the five classes namely External Interface Files, Internal Logical Files, External Inputs, External Outputs and External Queries.

Data Functions

Internal Logical File (ILF): ILF is a user identifiable group of logically related data or control information that is maintained within the boundary of the application.

External Interface File (EIF): EIF is a user identifiable group of logically related data or control information referred to the application, but maintained within the boundary of another application.

Transaction Functions

There are three types of transaction functions. They are External Input, External Output and External Inquiry.

External Input (EI): EI are received by the user to the software, which provides the application-oriented data.

External Output (EO): Things are provided by the software that goes with the outside systems like screen data, report data, error message and so on.

External Inquiries (EQ): Inquiries may be the command or requests that are generated from outside. It is the direct access to a database that retrieves the information.

Table 3.1 shows the computing procedure for Unadjusted Function Points (UFP) for the five categories of data and transaction functions.

Table 3.1: Unadjusted Function Point Calculation

Function Type	**Weight by Functional Complexity**		**Total FP**
EI	Low	A * 3	
	Average	A * 4	
	High	A * 6	
EO	Low	A * 4	
	Average	A * 5	
	High	A * 7	
EI	Low	A * 3	
	Average	A * 4	
	High	A * 6	
ILF	Low	A * 7	
	Average	A * 10	
	High	A * 15	
EIF	Low	A * 5	
	Average	A * 7	
	High	A * 10	
Total number of UFP			

Where, A - Number of functional units of that category present in the software.

After calculating the unadjusted function point, the next step involves is gathering the information about the environment and complexity of the project or application. The General System Characteristics (GSC) are a scale from 0 to 5 (degree of influence) as shown in Table 3.2.

Table 3.2: General System Characteristics

S. No	General System Characteristics
1.	There are communication facilities to aid in transferring or exchanging the information with the application or system.
2.	Handling the distributed data and processing functions.
3.	The response time or output required by the user.
4.	The heavy use of the current hardware platform where the application is executed.
5.	The transactions that are executed daily, weekly, monthly, etc.
6.	The On-line percentage of the information is entered.
7.	The end-users efficiency to design the application.
8.	Updating the ILF's through On-Line Transaction.
9.	The application provides extensive logical or mathematical processing.
10.	The application is developed to meet one or many user's needs.
11.	The difficulties of the conversion and installation.
12.	The effective and automated are a start-up, back-up, and recovery procedures.
13.	The applications are specifically designed, developed, and supported to install at multiple sites for multiple organizations.
14.	The application is specifically designed, developed, and supported to facilitate change.

After, all the 14 GSC's, the Complexity Adjustment Factors (CAF) is calculated. The formula that is used to calculate the CAF using Equation (3.3)

$$\mathrm{CAF} = 0.65 + \left(0.01 * \sum_{i=0}^{n} C\,i\right) \qquad (3.3)$$

Where,

n = 14 GSC's.

Ci - Complexity Adjustment Factor of C1 to Cn.

After determining the value of UFP and CAF, it is necessary to compute FP. The formula is calculated in the final FP count, which is given in the Equation (3.4).

$$\mathrm{FP} = \mathrm{UFP} * \mathrm{CAF} \tag{3.4}$$

Advantages

i. It calculates the size in the users' perspective.
ii. The FP metric doesn't correspond to any actual physical attribute of a software system (such as lines of code or the number of subroutines).It is useful as a relative measure to compare projects, measure productivity, and estimate the amount, develop effort and time needed for a project.
iii. FP can be applied early in the software development lifecycle.
iv. It is independent of programming languages.
v. It is a good sizing technique for the application programs in 1980's.

Limitations of FPA in the sense of Modern Software System

i. FPA focuses on the computation part of an application. In 1980's, an application system had a full computational part. So, it is focused on external inputs, outputs, inquiries, internal logical files and External interface files. But Modern Software system has a huge volume of the document. The learning content may express in terms of video, audio, simulation, animation or textual document. Sizing of this part was not mentioned in FPA.

ii. It is a count-based method. If the count of each component is high then it states that the complexity is high.
iii. It is not well suited to non-Management Information System applications, especially modern software system like web applications.

Feature Points

It was the extension of FPA designated to deal with different kind of applications such as embedded system, real-time system, system software, etc. FPA never considers the complexity of algorithms involved in each application. To overcome that problem, feature point method was introduced (Ursula 2003). The complexity of algorithms defined in terms of the number of rules required to express that algorithm. The formula for calculating Feature Points (FuP) is given in Equation (3.5).

$$\text{FuP} = \text{Raw Feature Points} * \text{CAF} \tag{3.5}$$

Determination of Raw Feature points

Count all inputs, outputs, files, inquiries, algorithms and interfaces present in a software system and multiply with the average weighting factors of the future type. The sums of all values are known as raw feature points. The Table 3.3 assists for calculating raw feature points.

Table 3.3: Calculating Raw Feature Points

Feature Type	Average	Total
No. of Inputs	B * 4 =	
No. of Outputs	B* 5 =	
No. of Files	B * 7 =	
No. of Inquiries	B * 4 =	
No. of Interfaces	B * 7 =	
Count the number of Algorithms	B * 3 =	
Total Raw Feature Points:		

Where, B - Number of raw future points of that category present in the software

Determination of CAF

The complexity adjustment factor is calculated based on the two environmental factors. The range of influence of each factor is from 1 to 5. The environmental factors are the logic values and data values. Logical value is assessed based on the complexity of algorithm or logics used in the application. The data value is assessed based on the complexity of data used in algorithm or logics used in the application. The Table 3.4 assists to find the environmental factors of an application. Choose any one from each factor category.

Table 3.4: Environmental Factors

Environmental Factors and values	
Logic Values (select one)	
Simple algorithms and calculations	1
Majority of simple algorithms	2
Average complexity of algorithms	3
Some difficult algorithms	4
Many difficult algorithms	5
Data values(select one)	
Simple Data	1
Numerous variables but simple relationships	2
Multiple Fields, Files and Interactions	3
Complex file structures	4
Very complex files and data relationships	5

The sum of logical value and data value provide environmental factor. Environmental factor ranges from 2 to 10. For each range of environmental factor, specific CAF is assigned. The Table 3.5 shows the CAF value for each range of environmental factor.

Table 3.5: CAF Value for Environmental Factor

Environmental factor	CAF
2	0.6
3	0.7
4	0.8
5	0.9
6	1.0
7	1.1
8	1.2
9	1.3
10	1.4

The exact feature point of the system is the product of the new feature point and CAF.

Advantages

i. It is an excellent approach to size the algorithmically intensive system.

ii. FP can be applied early in the software development lifecycle.

iii. It is independent of programming languages. It performs well in the embedded system and the real time system sizing.

Limitations of Feature points in the sense of Modern Software System

i. It never considers other technical factors that influence the execution of Modern Software system.

ii. It never considers the database and networking support that is needed for the application.

iii. It considers only the simple entities and algorithms used by the system.

iv. It never consider video, audio, simulation, animation and their worth fullness.

Use Case Points (UCP)

Use case point was introduced in the year of 1993 by Karner of Objectory. It is an extension of FPA. It supports sizing in the early stage itself. The following Equation (3.6) is used for calculating UCP.

$$UCP = UUCP + TCF + EF \quad (3.6)$$

Where,

UUCP - Unadjusted Use Case Points.
TCF - Technical Complexity Factor
EF - Environmental Factor

Determination of Unadjusted Use Case Points (UUCP)

UUCP can be calculated based on the unadjusted actor weight and unadjusted use case weight. Identify actors and its complexity from each use case of an application system. Find the weight because the weight may be 1, 2 or 3 based on the actor complexity that is simple, average or complex. Sum the weight for the actors in all use cases to obtain the Unadjusted Actor Weight (UAW). Similarly, identify the use cases and assign weight 5, 10, 15 based on the complexity. Sum the weight for all use cases to obtain the Unadjusted Use Case Weight. The Equation (3.7) is used for calculating the UUCP.

$$UUCP = UAW + UUCW \quad (3.7)$$

Determination of Technical Complexity Factor (TCF)

The technical complexity of the product can be calculated based on the degree of influence of thirteen technical factors. The Table 3.6 describes the technical factors and their weight. It is similar to the CAF calculation of FPA.

Table 3.6: Technical Factors and their Weight

Technical factor	**Weight**
Distributed system	2
Response or throughput performance objectives	2
End-user efficiency	1
Complex internal processing	1
Reusable code	1
Easy to install	0.5
Easy to use	0.5
Portable	2
Easy to change	1
Concurrent Processing	1
Include security features	1
Provide access for third parties	1
Special user training facilities are required	1

The degree of influence of each factor ranges from 0 to 5. For each factor, multiply the degree of influence by the weight, and sum the products to obtain the Technical Complexity Sum (TSUM). The Equation (3.8) is used for computing TCF.

$$TCF = 0.6 + 0.01 * TSUM \qquad (3.8)$$

Determination of Environmental Factor (EF)

It is calculated based on eight environmental factors, which addresses the skills and training of the staff and requirement stability. The rating of influence ranges from 0 to 5. Multiply the rate of influence with the weight and sum them to obtain Environment sum (Esum). The Table 3.7 shows the environmental factors and weight.

Table 3.7: The Environmental Factors and Weight

Environmental factors	Weight
Familiar with rational unified process	1.5
Application experience	0.5
Object oriented experience	1
Lead analyst capability	0.5
Motivation	1
Stable requirements	2
Part-time workers	-1
Difficult programming languages	-1

The Equation (3.9) is used for computing Environmental Factors (EF).

$$\mathrm{EF} = 1.4 - 0.03 * \mathrm{Esum} \tag{3.9}$$

Advantages

i. It supports for estimating the size of software in the first phase of development itself.

ii. It is good for the application that is generated by using object oriented methodology.

Limitations of Use case points in the sense of Modern Software System

i. Use case provides the initial view of the business model. But it is not much detailed and using this we can’t provide exact estimates.

ii. It counts the number of actors and use cases involved in an application system and identify the complexity. But it never identifies the implementation level difficulties.
iii. Use case complexity is assessed based on number of transactions. It never considers the weight of code or inner part of use case
iv. Sizing of the document part of Modern Software system is not mentioned.
v. Simulation, animation, video and audio specifications and their complexities are not assessed.

Object Points

Object points were introduced by Banker in 1991. It was object count instead of function count. Here the objects denote rule set, 3GL module, screens and reports. These objects are closer to work done by the developers. This approach meshes well with projects that use integrated computer aided software engineering environments to develop software (Renjeev 2007).

Determination of object points

Count all instances of each object type. Each object is assessed with the complexity weight. Sums up the complexity weight of all objects to get the Object Points (OP). Multiply OP by a Reuse Factor (RF). Reuse Factor is expressed in percentage 10% corresponds to the value of 0.1 and the New Object Points (NOP) is calculated using the following Equation (3.10).

$$\mathrm{NOP} = \mathrm{OP} * (1 - \mathrm{RF}) \qquad (3.10)$$

Advantages

i. Good for GUI based applications.
ii. It highly considers for reusability.
iii. It is suitable for object oriented applications.

Limitations of object points in the sense of Modern Software system

i. Modern Software system is also a GUI based application. Instead of screens, reports, and code list of special objects, there are animation, simulation, video, etc. Object point suggests no way for sizing those items.
ii. Modern Software system is a web-based application. It is accessed by a variety of students from the geographically distributed area. So, multiple system characteristics have to be considered. But, object point considers only reusability out of all technical and environmental factors that influence the system.
iii. This method is not suitable for research and analytics applications.

Other Sizing Approaches

There is few more sizing techniques are used by some companies based on their needs (Shukor 2009).

Web Points

Assessing the size of web pages, David Clary introduced this method in 2000. The size is assessed based on the complexity of web page. The complexity of each web page is considered based on the count of words and number of hyperlinks. Counting the size of each page and summing them gives the size of an application. A modern Software system has multiple algorithms and produces multiple reports. The database and different media files are also involved. So this sizing technique is not suited for Modern Software system. It supports only for assessing the size of the small web site.

Web Objects

It was introduced by Donald Reifer in 2000. Web Objects considers multiple objects of web pages like building blocks, web components, graphic files, multimedia files and scripts. It counts all objects and as well as FPA web objects. It is good for assessing the size of web site, but Modern Software system is highly more than a website. It is a document rich web application. So, it is not suitable for sizing Modern Software system.

Backfiring

Capers Jones of Software productivity research developed a technique in 1984 called “Backfire.” It estimates the size of existing legacy systems by counting the lines of code in the software product and then multiplying by a language-specific conversion factor. This technique provides moderate accuracy. It is based on LOC, which could not support in assessing the size of modern software system.

Object Oriented Size Measures

Entities that persist in the world are modeled on a software program, which includes both the application domain and solution domain (Rodrigo 2009). Application objects can be physical things, roles and events. Solution objects may be architecture elements and software components. The trick to obtain useful size measure is to stay near the application side. But, application object provides limited information for sizing. So, it mostly provides an inaccurate estimate in the early stages.

3.5 RISKS ASSOCIATED TO MODERN SOFTWARE SYSTEM'S SIZE ESTIMATION

The risks of the modern software system are classified into two broad categories. They are,

i. Functional Risks
ii. Social Risks

3.5.1 Functional Risks

The functional risks of size estimation are based on the ambiguity present in the identification of functional units.

The Versatile Behavior of Modern Software

The modern software is versatile using distinct programming languages, operating systems, file formats, topologies, SDLC and application tools. The functional unit of one software may differ from other software. It may lead to confusions in identifying functional units.

The Variable Behavior of Functional Units

In the same software, the same functional unit will behave differently in different modules. It also increases the difficulties for identifying functional units of modern software systems.

Difficult to Rank Function Points

Ranking of function points is differing from organisation to organisation. So it increases the confusions in estimating the size of modern software systems.

Insufficient CAF

The existing IFPUG Function Point size estimation technique uses only fourteen CAF. It is not competent for sizing modern software systems.

Dynamic Function Points

The dynamic behaviour of function points generates difficulties in the size estimation of modern software systems.

3.5.2 Social Risks

The social risks of size estimation and project management mainly depends on industrial policies, socio-economic policies, political system, and universal standards of the client, developer and domain.

Economic and Financial Risks

The economic and financial status of the institutions also affects the development of software in its estimated period of time as the fund flow is essential for managing the needs of software development process.

Social and Environmental Risks

The social issues in the society like employees, institutional policies, working hours and ecological policies also affects the development of software. It creates major impact on size estimation.

Security Risks

The secured transfer and storing of information is highly essential till the life time of software system. So, efficient algorithms must be developed for those issues.

National Policies

The national policies of the client and developer may also affect the development process of the software system.

Universal Standards

The universal standards of software development, employees and organizational standards also affect the growth of software development process.

3.6 FINDINGS IN FPA

The traditional function point estimation techniques were using only five functional units and fourteen CAF (discussed in 3.4.3). These are not sufficient for estimating the size of modern software system. The following are the new functional units and CAF for attaining the accurate sizing of modern software to an extent.

Internal Input

It is an important essential functional unit for modern software. For example,

```
void main()
   {
         const float pi = 3.14;
         int r = 10;
         float k;
         k=pi * r * r;
         printf (“Area = %f”, k);
   }
```

In the above function, the internal direct assignments (eg. variable r) and constants (eg. pi) are the examples for internal input. In the traditional FPA, the internal input is not considered as functional units. Therefore, it will reduce the size of the software product in FPA.

Internal Operations

The internal operations are not considered in the traditional FPA estimation. For example,

```
void main()
          {
```

```
        Int i,j,k,l;
        printf("Enter the value of a and b");
        scanf("%d%d", &i,&j);
        k=i+j;
        l=k-i;
        printf("Value of l=%d",l);
    }
```

In the above example, variables 'i' and 'j' are EI, variable 'l' is EO, but the internal operation 'k=i+j' is not considered as functional unit. The internal operations are playing very important role in scientific and AI software programs. To increase the accuracy of modern software size estimation, internal operations also can be considered as a functional unit for FPA size estimation process.

Indexed Data

The arrays and lists are very essential data variables for modern software. But all the indexed values are not getting importance. The indexed data variables also considered as the single valued variables. For example, in the existing FPA estimation 'int a[10]' will be getting equal weightage as that of 'int a'. It reduces the effort level of developer at the time of estimation.

Multiple Forms of Output

Nowadays, the modern software are capable to create many forms of outputs like data report, crystal report, excel formats, GUI formats, database reports, etc. But the existing FPA methods are considering only one format of output. The exclusion of different forms of output affects the size of the software system. Therefore, the time and cost constraints are not accurate in the estimation.

Insufficient Metric Values

The size of the functions and the data handling with the functions are increasing by time. The existing metric values (low, average and high) are not sufficient. Hence, we have to add one more very high metric value.

Database and Text Files

The database and text files were not considered in the existing function point methods. The current technologies like machine learning, data mining, data analytics and big data are using a large amount of historical and primary data. The neglecting of database and text affects the actual effort level of the software system.

Multi-valued Function Points

A variable will act as one functional unit in one function and the same variable will act as another functional unit in another function is known as Multi-valued Function Points (MVFP). For example,

```
void get(void);
void add (int, int);
```

```
int a,b;
void main()
{
        get();
        add(a,b);
}
void get()
{
        a=10;
        b=20;
}
void add(int a, int b)
{
        int c= a+b;
}
```

In the above example, the variables '*a*' and '*b*' are as internal inputs in function get() and as EI in function add(). Similarly, ILF of one function becomes EIF of another function and EQ of one function is EI of another function. The importance of MVFP is not considered in FPA method.

Dependent Function Points (DFP)

Some functional units are identified based on some other functional units. The choice based functional segments are example for DFP. For example,

```
void main()
    {
        int a, b, big, small;
        printf ("Enter a and b values");
        scanf("%d %d", &a, &b);
        if (a > b)
         {
                big = a;
                small = b;
         }
        else
         {
                big = b;
                small = a;
         }
    }
```

In the above example, 'if' block and 'else' block will be chosen based on the variables '*a*' and '*b*'. If one block is chosen then all other blocks are omitted. The small applications won't give any impact on its size estimation. But in the large scale systems, the choices are playing great role in size of the software. The choices and dependent function points were not considered in FPA method. Similarly, case () structure also is an example for DFP.

Composite Function Points (CFP)

A variable will get the characteristics of different functional units in the same function is known as CFP. For example,

```
void main()
  {
        int a,b;
        printf("Enter a and b values");
        scanf("%d%d",&a,&b);
        a=a+b;
        b=a-b;
        a=a-b;
  }
```

In the above example, variable '*a*' and '*b*' are accepting the characteristics of External Inputs and Intermediate Results. Similarly, External Inquiries becomes Internal or External Inputs within the same function. The composite behaviors of functional units are not discussed in FPA method.

3.7 SUMMARY

The existing FPA has five functional units and fourteen CAF. These are not sufficient for measuring the size of modern software system. The functional risks in Function Point estimations are to rank function points, insufficient complexity adjustment factors, and dynamic function points. The economic, financial, environmental, security, national policies, and universal standards are social risks of modern software.

Internal input, internal operations, indexed data, multiple forms of output, insufficient metric values, database and text files, multi-valued function points, dependent function points, and composite function points are some additional factors affecting the modern software which is not reflected in the existing FPA method.

Updating the above factors with the existing FPA method will yield an opt method for finding the size of modern software system.

CHAPTER 4

MODERN METRICS SIZING TECHNIQUE

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 EI, EO, ILF, EQ and EIF, it includes Internal Input (II), Internal Operations (IO) and Data and Text (DT). It also recognizes SDLC, updated CAF, trial versions of the software, indexed data, multiple forms of output, user developer views on system and social, economic and political laws of the Nation. Therefore, the defects per function point are reduced by the novel FPA, using MM technique.

4.1 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.

4.1.1 Architecture of MM

The functional diagram of MM includes all the internal and external function points of a software system. The traditional FPA estimation technique has only five functional units (EI, EO, EQ, EIF and ILF). But the MM has added three more functional units (II, IO and DT) and it has eight functional units. The MM also includes twenty two CAF, whereas the traditional FPA has only fourteen CAF. The architectural diagram of MM is shown in the Figure 4.1

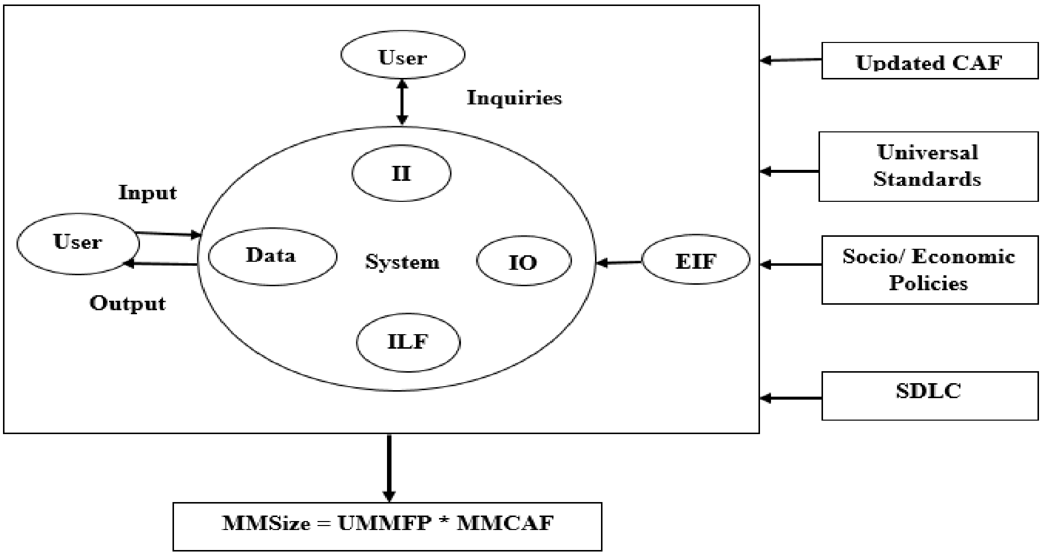


Figure 4.1: Architecture of Modern Metrics

4.1.2 Functional Units of MM

The functional units of software are the basic element for estimating the size of software. The functional units are divided into three 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 factors. External functional units are influencing the system by external factors or communications from system to an external factor. The internal inputs, internal operations and internal logical files are the internal functional units of the MM. 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: The defined constants and internal assignments of variables are internal inputs.

   b) Internal Operations: A complete cycle of operations in the system which is not present under any other functional units.

   c) Internal Logical Files: It is a supporting software or data present in the system for executing the system successfully.

- External functional units

   a) External Inputs: Inputs given to the system through input devices by an external factor.

   b) External Outputs: The results received from the system through output devices for an external factor.

c) External Inquiries: The external questions raised from the actor during the execution time for checking the accuracy of the system.

d) External Interface Files: It is a supporting software or data present in the external system for executing the software successfully.

- Hybrid functional units

a) Data and Text: 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 in any operation and it may be tables, historical data, help files, images or other text documents. It may be both internal and external.

4.1.3 The Metrics of the Functional Units of MM

The metrics of the functional units of modern software is difficult to find and classify it. Some important functional units of functions are identified and listed in the Table 4.1.

Table 4.1: Metrics of Functional Units

S. No	Functional Unit	Metrics
1	II	Constants, internal assignments and internal keys.
2	IO	Choices, A complete operational cycle which is not taking part with any other functional calculations, dynamic effects of web pages, internal algorithms, array input, output or calculations, the properties and events assigned to the GUIs, function calling in a program.
3	ILF	The driver files for other software, header files and packages.
4	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	EO	The results displayed using output statements, output devices, output GUIs like label box, list box, text box, combo box.
6	EQ	The queries generated by the users for the better operations of the system.
7	EIF	The driver files used for connecting external devices and remote systems, anchor tags.
8	DT	Tables, text files, image files, help files, data files and webpage contents.

The way of finding the functional units of modern software is explained in Appendix 1.

4.1.4 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. The calculations of effort modifiers are present in Appendix 2. By using a set of inflexible standards the metrics are categorized.

EI Functional Values

The EI of all the functions are identified and tabulated. Then, the EI functional values are categorized and valued based on its complexity. The metrics and its values of EI functional values are shown in the Table 4.2.

Table 4.2: EI Functional Values

S. No	EI Functional Values	EI Metrics	EI Metric Values
1	1 to 3	Low	3
2	4 to 5	Average	4
3	6 to 8	High	6
4	>8	Very High	9

If the EI functional value is in-between 1 and 3, the EI metric is low and its value is 3. If the EI functional value is in-between 4 and 5, the EI metric is

Average and its value is 4. If the EI functional value is in-between 6 and 8, the EI metric is High and its value is 6. If the EI functional value is greater than 8, the EI metric is very high and its value is 9.

II Functional Values

The II of all the functions are identified and tabulated. Then, the II functional values are categorized and valued based on its complexity. The metrics and its values of II functional values are shown in the Table 4.3.

Table 4.3: II Functional Values

S. No	**II Functional Values**	**II Metrics**	**II Metric Values**
1	1 to 3	Low	3
2	4 to 5	Average	4
3	6 to 8	High	6
4	>8	Very High	9

If the II functional value is in-between 1 and 3, the II metric is low and its value is 3. If the II functional value is in-between 4 and 5, the II metric is Average and its value is 4. If the II functional value is in-between 6 and 8, the II metric is High and its value is 6. If the II functional value is greater than 8, the II metric is very high and its value is 9.

EO Functional Values

The EO of all the functions are identified and tabulated. Then, the EO functional values are categorized and valued based on its complexity. The metrics and its values of EO functional values are shown in the Table 4.4.

Table 4.4: EO Functional Values

S. No	EO Functional Values	EO Metrics	EO Metric Values
1	1 to 4	Low	4
2	5 to 6	Average	5
3	7 to 9	High	7
4	>9	Very High	10

If the EO functional value is in-between 1 and 4, the EO metric is low and its value is 4. If the EO functional value is in-between 5 and 6, the EO metric is Average and its value is 5. If the EO functional value is in-between 7 and 9, the EO metric is High and its value is 7. If the EO functional value is greater than 9, the EO metric is very high and its value is 10.

IO Functional Values

The IO of all the functions are identified and tabulated. Then, the IO functional values are categorized and valued based on its complexity. The metrics and its values of IO functional values are shown in the Table 4.5.

Table 4.5: IO Functional Values

S. No	IO Functional Values	IO Metrics	IO Metric Values
1	1 to 3	Low	3
2	4 to 5	Average	4
3	6 to 8	High	6
4	>8	Very High	9

If the IO functional value is in-between 1 and 3, the IO metric is low and its value is 3. If the IO functional value is in-between 4 and 5, the IO metric is Average and its value is 4. If the IO functional value is in-between 6 and 8, the IO metric is High and its value is 6. If the IO functional value is greater than 8, the IO metric is very high and its value is 9.

DT Functional Values

The DT of all the functions are identified and tabulated. Then, the DT functional values are categorized and valued based on its complexity. The metrics and its values of DT functional values are shown in the Table 4.6.

Table 4.6: DT Functional Values

S. No	DT Functional Values	DT Metrics	DT Metric Values
1	1 to 4	Low	4
2	5 to 6	Average	5
3	7 to 9	High	7
4	>9	Very High	10

If the DT functional value is in-between 1 and 4, the DT metric is low and its value is 4. If the DT functional value is in-between 5 and 6, the DT metric is Average and its value is 5. If the DT functional value is in-between 7 and 9, the DT metric is High and its value is 7. If the DT functional value is greater than 9, the DT metric is very high and its value is 10.

EQ Functional Values

The EQ of all the functions are identified and tabulated. Then, the EQ functional values are categorized and valued based on its complexity. The metrics and its values of EQ functional values are shown in the Table 4.7.

Table 4.7: EQ Functional Values

S. No	EQ Functional Values	EQ Metrics	EQ Metric Values
1	1 to 3	Low	3
2	4 to 5	Average	4
3	6 to 8	High	6
4	>9	Very High	9

If the EQ functional value is in-between 1 and 3, the EQ metric is low and its value is 3. If the EQ functional value is in-between 4 and 5, the EQ metric is Average and its value is 4. If the EQ functional value is in-between 6 and 8, the EQ metric is High and its value is 6. If the EQ functional value is greater than 8, the EQ metric is very high and its value is 9.

ILF Functional Values

The ILF of all the functions are identified and tabulated. Then, the ILF functional values are categorized and valued based on its complexity. The metrics and its values of ILF functional values are shown in the Table 4.8.

Table 4.8: ILF Functional Values

S. No	ILF Functional Values	ILF Metrics	ILF Metric Values
1	1 to 7	Low	7
2	8 to 14	Average	10
3	15 to 21	High	15
4	>21	Very High	22

If the ILF functional value is in-between 1 and 7, the ILF metric is low and its value is 7. If the ILF functional value is in-between 8 and 14, the ILF metric is Average and its value is 10. If the ILF functional value is in-between 15 and 21, the ILF metric is High and its value is 15. If the ILF functional value is greater than 21, the ILF metric is very high and its value is 22.

EIF Functional Values

The EIF of all the functions are identified and tabulated. Then, the EIF functional values are categorized and valued based on its complexity. The metrics and its values of EIF functional values are shown in the Table 4.9.

Table 4.9: EIF Functional Values

S. No	EIF Functional Values	EIF Metrics	EIF Metric Values
1	1 to 5	Low	5
2	6 to 9	Average	7
3	10 to 13	High	10
4	>13	Very High	14

If the EIF functional value is in-between 1 and 5, the EIF metric is low and its value is 5. If the EIF functional value is in-between 6 and 9, the EIF metric is Average and its value is 7. If the EIF functional value is in-between 10 and 13, the EIF metric is High and its value is 10. If the EIF functional value is greater than 13, the EIF metric is very high and its value is 14.

4.1.5 Calculating Functional Units (FU) of MM

All the classes and functions are analyzed and listed with the corresponding functional units using Table 4.10 format. All the functional units are identified in each functions of software and are tabulated. The total number of functions referred and a total functional unit of each type is calculated at the end of the table.

Table 4.10: Calculating Functional Units

S. No	Name of the Function	EI	II	EO	IO	DT	EQ	ILF	EIF
1									
2									
3									
4									
Total number of functions referred									
Total Functional Units									

4.1.6 Complexity Adjustment Factors (CAF) of MM

The project complexity and management process is one of the challenging tasks 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 is the duties, the degree of autonomy and the scope of responsibilities.

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

- Technology used in the software.
- Standardisation and development models associated to the software.
- Distribution and processing of application.
- The novelty and innovation of the developing system.
- 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?
3. Whether it has any distributed processing?

4. Is representation complex?
5. Whether the system works in congested environment?
6. Does it require any online updating?
7. Whether the system has online input, output and operations?
8. Does it require any major file on online updating?
9. Does 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. Does 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. Does the system using more than one programming languages, DBMS, Web tools, Drivers, etc.?
18. Does the networking environment using more than one network topologies?
19. Does the system installed in different nations and uses different social, cultural, economic and environmental laws?

20. Does the system giving multiple forms of output?

21. Does the trial version and model version of software development affects the system?

22. Does 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 (4.1) gives the value of MM Complexity Adjustment Factor (MMCAF) of the software.

$$MMCAF = 0.25 + 0.01 * Fi \qquad (4.1)$$

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

4.1.7 Calculating Unadjusted Modern Metrics Function Points (UMMFP)

The UMMFP is the number of raw function points present in software. The Table 4.11 is used to calculate the UMMFP.

Table 4.11: Calculation of UMMFP

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	

The total number of functions is the sum of the functions calculated individually in each functional unit. It is calculated during the functional unit calculations of each function in software. If the function having any functional unit then immediately the corresponding function count is increased by one.

The distinct functional units of each function is calculated and represented as shown in Table 4.10. The total functional units are the sum of each functional unit in all functions.

The ratio of total functional units and total number of functions is known as Average Functional Units.

$$AFU = TFU/TF \qquad (4.2)$$

The value of metrics and metric value (w) are calculated by using weightage factor and weightage of the functional units as shown in Table 4.2 to Table 4.9.

The UFP is the product of total number of functions and weightage.

The UMMFP is the sum of all the Unadjusted Function Points of each functional unit.

4.1.8 Modern Metrics Size (MMSize)

MMSize is the size of the software based on MM. The unit of MM software size is MMFP (Modern Metrics Function Points). It is calculated using the Equation (4.3)

$$MMSize = UMMFP * MMCAF \qquad (4.3)$$

The MMSize is the product of UMMFP and MMCAF.

4.2 ALGORITHM FOR MM

It is a step by step instruction to find the solution for modern software size using MM.

Nomenclature

EI	-	External Inputs
II	-	Internal Inputs
EO	-	External Outputs
IO	-	Internal Operations
DT	-	Data and Text
EQ	-	External Inquiries
ILF	-	Internal Logical Files
EIF	-	External Interface Files
FEI	-	Functions in External Input
FII	-	Functions in Internal Inputs
FEO	-	Functions in External Outputs
FIO	-	Functions in Internal Operations
FDT	-	Functions in Data and Text
FEQ	-	Functions in External Inquiries
FILF	-	Functions in Internal Logical Files
FEIF	-	Functions in External Interface Files
AEI	-	Average functional units of External Inputs

AII	-	Average functional units of Internal Inputs
AEO	-	Average functional units of External Outputs
AIO	-	Average functional units of Internal Operations
ADT	-	Average functional units of Data and Text
AEQ	-	Average functional units of External Inquiries
AILF	-	Average functional units of Internal Logical Files
AEIF	-	Average functional units of External Interface Files
WEI	-	Weightage of External Inputs
WII	-	Weightage of Internal Inputs
WEO	-	Weightage of External Outputs
WIO	-	Weightage of Internal Operations
WDT	-	Weightage of Data and Text
WEQ	-	Weightage of External Inquiries
WILF	-	Weightage of Internal Logical Files
WEIF	-	Weightage of External Interface Files
UEI	-	Unadjusted External Inputs
UII	-	Unadjusted Internal Inputs
UEO	-	Unadjusted External Outputs

UIO	-	Unadjusted Internal Operations
UDT	-	Unadjusted Data and Text
UEQ	-	Unadjusted External Inquiries
UILF	-	Unadjusted Internal Logical Files
UEIF	-	Unadjusted External Interface Files
UMMFP	-	Unadjusted Modern Metrics Function Points
CAF	-	Complexity Adjustment Factors
MMCAF	-	Modern Metrics Complexity Adjustment Factors
MMSize	-	Modern Metrics Size

Algorithm Modern Metrics

1. Declare and initialize variables

   Initialize variables for functional units EI, II, EO, IO, DT, EQ, ILF and EIF as zero.

   Initialize variables for count functions FEI, FII, FEO, FIO, FDT, FEQ, FILF and FEIF as zero.

   Initialize variables for finding average functional units AEI, AII, AEO, AIO, ADT, AEQ, AILF and AEIF as zero.

Initialize variables for weight age of functional units WEI, WII, WEO, WIO, WDT, WEQ, WILF and WEIF as zero.

Initialize variables for unadjusted Function Points UEI, UII, UEO, UIO, UDT, UEQ, UILF and UEIF as zero

Declare a variable for Unadjusted Modern Metrics Function Points UMMFP

Declare other variables CAF, MMCAF, MMSize

2. Analyze the functions

a) External Input (EI):

Analyzes the entire function and finds all the External Inputs and each occurrence increases EI by one.

After completing the analysis, if at least one EI value is present in the function then FEI is increased by one.

b) Internal Input (II):

Analyzes the entire function and finds all the Internal Inputs and each occurrence of it increases II by one.

After completing the analysis, if at least one II value is present in the function then FII is increased by one.

c) External Output (EO):

Analyzes the entire function and finds all the External Outputs and each occurrence of it increases EO by one.

After completing the analysis, if at least one EO value is present in the function then FEO is increased by one.

d) Internal Operations (IO):

Analyzes the entire function and finds all the Internal Operations and each occurrence of it increases IO by one.

After completing the analysis, if at least one IO value is present in the function then FIO is increased by one.

e) Data and Text (DT):

Analyzes all the historical data, help files and other documents in the function and count the words of it, then perform the division operation. The word count is divided by 8000 then takes the quotient value. If the quotient value is greater than zero then add quotient with DT and increase the value of FDT by one.

f) External Inquiries (EQ):

Analyzes the entire function and finds all the External Inquiries and each occurrence of it increases EQ by one.

After completing the analysis, if at least one EQ value is present in the function then FEQ is increased by one.

g) Internal Logical Files (ILF):

Analyzes the entire function and finds all the Internal Logical Files and each occurrence of it increases ILF by one.

After completing the analysis, if at least one ILF value is present in the function then FILF is increased by one.

h) External Interface Files (EIF):

Analyzes the entire function and finds all the External Interface Files and each occurrence of it increases EIF by one.

After completing the analysis, if at least one EIF value is present in the function then FEIF is increased by one.

Step 2 is repeated until all the functions are analyzed.

3. Find the average of functional units

AEI = EI / FEI

AII = II / FII

AEO = EO / FEO

AIO = IO / FIO

ADT = DT / FDT

AEQ = EQ / FEQ

AILF = ILF / FILF

AEIF = EIF / FEIF

4. Find the weightage of functional units

a) Weightage of External Input:

```
If AEI <= 3 then
        WEI = 3
Else if AEI > 3 and AEI <= 5 then
        WEI = 4
Else if AEI > 5 and AEI <= 8 then
        WEI = 6
Else
        WEI = 9
End If
```

b) Weightage of Internal Input:

```
If AII <= 3 then
        WII = 3
Else if AII > 3 and AII <= 5 then
        WII = 4
Else if AII > 5 and AII <= 8 then
        WII = 6
Else
        WII = 9
End If
```

c) Weightage of External Output:

```
If AEO <= 4 then
```

```
        WEO = 4
Else if AEO > 4 and AEO <= 6 then
WEO = 5
Else if AEO > 6 and AEO <= 9 then
        WEO = 7
Else
        WEO = 10
End If
```

d) Weightage of Internal Operations:

```
If AIO <= 3 then
        WIO = 3
Else if AIO > 3 and AIO <= 5 then
        WIO = 4
Else if AIO > 5 and AIO <= 8 then
        WIO = 6
Else
        WIO = 9
End If
```

e) Weightage of Data and Text:

```
If ADT <= 4 then
        WDT = 4
Else if ADT > 4 and ADT <= 6 then
        WDT = 5
Else if ADT > 6 and ADT <= 9 then
        WDT = 7
```

```
Else
        WDT = 10
End If
```

f) Weightage of External Inquiries:

```
If AEQ <= 3 then
        WEQ = 3
Else if AEQ > 3 and AEQ <= 5 then
        WEQ = 4
Else if AEQ > 5 and AEQ <= 8 then
        WEQ = 6
Else
        WEQ = 9
End If
```

g) Weightage of Internal Logical Files:

```
If AILF <= 7 then
        WILF = 7
Else if AILF > 7 and AILF <= 14 then
        WILF = 10
Else if AILF > 14 and AILF <= 21 then
        WILF = 15
Else
        WILF = 22
End If
```

h) Weightage of External Interface File:

If AEIF <= 5 then

WEIF = 5

Elseif AEIF > 5 and AEIF <= 8 then

WEIF = 7

Elseif AEIF > 8 and AEIF <= 12 then

WEIF = 10

Else

WEIF = 14

End If

5. Unadjusted Function Point (UFP) calculation:

UEI = FEI * WEI

UII = FII * WII

UEO = FEO * WEO

UIO = FIO * WIO

UDT = FDT * WDT

UEQ = FEQ * WEQ

UILF = FILF * WILF

UEIF = FEIF * WEIF

6. Unadjusted Modern Metrics Function Point (UMMFP) calculation:

UMMFP = UEI + UII + UEO + UIO + UDT + UEQ + UILF + UEIF

7. MM Complexity Adjustment Factor (MMCAF):

The Complexity Adjustment Factors (CAF) is valued using the complexity factors.

MMCAF = (0.25 + 0.01 * CAF)

8. Modern Metrics Size (MMSize) calculation:

MMSize = UMMFP * MMCAF

9. Stop

The above algorithm analyzes all the intermediate steps of Modern Metrics size estimation process. The accuracy of the estimation is increased because it does a deep analysis in the software.

4.3 OTHER ESTIMATIONS BASED ON MM

The other important metrics of SPM like productivity, effort, duration, cost and price of the software also calculated using MMSize.

4.3.1 Modern Metrics Productivity Factor (MMPF)

MMPF defines the amount of time required for completing one function point. The productivity factor may change from organization to organization. MMPF is calculated using the following Equation (4.4),

MMPF = Total Hours required to Complete a project / MMSize (4.4)

4.3.2 Modern Metrics Effort (MME)

MME denotes the amount of man-hours required for completion of the project. Software size is the primary independent variable affecting software development effort. The following Equation (4.5) is used for calculating effort using MM.

$$MME = MMSize * MMPF \tag{4.5}$$

The organization uses productivity factor as 11 because an average of 11 hours per Modern Metrics Function points were taken for software development.

4.3.3 Modern Metrics Duration (MMD)

MMD denotes the total time required for completing the project. The following Equation (4.6) is used for calculating duration using MM.

$$MMD = MME / (176 * \text{number of persons involved in the software development}) \tag{4.6}$$

The value 176 denotes monthly working hours of a person. The software industry people work on 22 days per month and per day 8 hours, totally 22*8 = 176 hours.

4.3.4 Modern Metrics Cost (MMC)

MMC of the software project is calculated based on the total expenditure for the development of the software. The following Equation (4.7) is used for calculating Cost of the project using MM.

MMC = Number of persons involved * Average remuneration of software developers * MMPF + Management cost (4.7)

The management cost will be varied from organization to organization. The Modern Metrics Unit Cost (MMUC) is calculated using the following Equation (4.8).

MMUC = MMC / MMSize (4.8)

4.4 SUMMARY

Modern Metrics (MM) is an Indian metrics, which is used to find the size of modern software in its design phase of system development life cycle. It is an opt method finding the size for all types of software. The MM has eight functional units. They are, Internal Inputs, Internal Operations, Internal Logical Files, External Inputs, External Outputs, External Inquiries, External Interface Files and Data and Text.

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. The effort modifiers estimation is explained in Appendix 2.

The size, productivity, effort, duration and cost of the software is estimated using the MM formulas.

CHAPTER 5

PRACTICAL IMPLEMENTATION OF MODERN METRICS

MM, is a novel technique for estimating the size of modern software system based on its internal, external and hybrid function points. The procedure for implementing MM is discussed in Chapter 4. Using the Aadhaar processing system, a practical implementation of MM is analyzed.

5.1 USE CASE MODEL OF MM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipse.

External Input, External Output, External Inquiries, External Interface File, Internal Input, Internal Operations, Data and Text and Internal Logical Files are the important use cases present in MM. External user, External software, database and storage are the users interacting with the use cases. All the use cases and users are combined then gives the size of modern software system.

The use case diagram of MM is present in the following Figure 5.1.

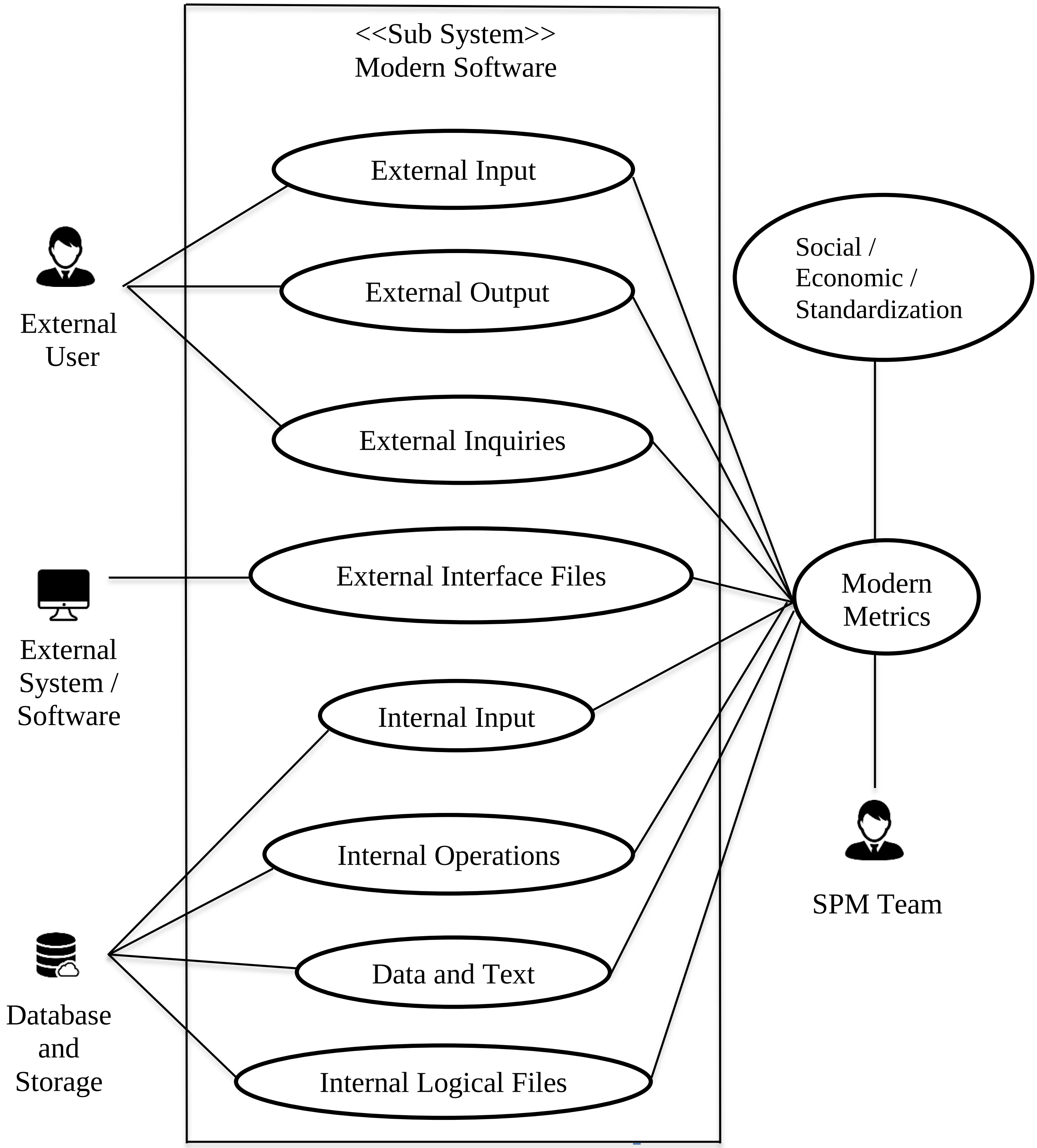


Figure 5.1: Use Case Model of MM

5.2 CALCULATING THE FUNCTIONAL UNITS

The functional units of each function in Aadhaar processing system is analyzed separately and tabulated using the Table 5.1.

Table 5.1: Functional Units Calculation

S.No	Name of the Function	EI	II	E	IO	D	E	ILF	EIF
1	allsched1	5	0	6	0	0	0	1	2
2	cprocess1	5	0	7	0	0	0	1	2
3	cprocess2	3	3	4	0	0	0	1	2
4	cpwd1	2	0	0	0	0	1	0	1
5	cpwd	4	6	0	0	0	1	0	2
6	cregister	9	0	2	1	1	0	0	2
7	ctransit1	1	0	10	3	0	0	1	2
8	ctransit	1	0	1	0	0	0	0	2
9	czpro	3	0	1	0	0	0	0	1
10	dt1	1	0	2	0	1	0	0	1
11	dt2	0	6	2	0	1	0	1	2
12	dt3	2	0	0	0	1	0	1	2
13	fcitizen	0	5	0	0	0	1	1	2
14	lic2	0	0	2	0	0	0	1	2
15	licapp1	1	0	2	1	0	0	1	2
16	licapp2	1	3	6	0	0	0	1	2
17	licapp3	0	6	2	0	0	0	1	2
18	licapp11	1	0	2	0	0	0	1	2
19	licpro2	0	6	3	0	0	0	1	2
20	licst2	1	0	3	0	0	1	1	2
21	licst3	1	8	10	0	0	0	1	2
22	pinmast1	1	7	3	3	0	0	0	2
23	pinmast	0	0	2	0	0	0	1	2
24	pp1	1	0	2	0	0	1	1	2
25	ppst1	1	4	6	0	0	0	0	2
26	ppst11	1	1	1	0	0	0	0	2
27	prolic2	1	0	1	1	0	0	1	2
28	register	0	0	0	0	1	0	1	1

S.No	Name of the Function	EI	II	E	IO	D	E	ILF	EIF
29	registerc	0	0	0	0	1	1	1	1
30	sappno	0	1	1	0	0	0	0	2
31	signin	0	3	4	3	0	1	0	2
32	sregister	0	3	2	0	1	1	1	2
33	tprolic	1	1	0	0	0	0	1	2
34	transit1	1	0	1	0	0	1	1	2
35	transit	1	1	1	0	0	0	0	2
36	tsched	1	6	0	0	0	0	1	2
37	updlic	4	0	1	0	0	1	1	2
38	vastaff	0	4	1	0	0	0	0	2
39	vcz1	1	10	7	0	0	0	0	2
40	vcz	0	0	1	0	0	0	0	1
41	vpp1	1	1	2	2	0	1	0	1
42	vpp2	1	11	12	0	0	0	1	2
43	vpp3	3	0	4	0	0	0	0	2
44	vpppro2	5	0	2	0	0	0	1	2
45	vpropp1	1	1	3	0	0	0	1	2
Total number of functions		33	23	38	6	7	11	29	45
Total functional units		69	10	12	11	20	16	29	87

5.3 UNADJUSTED MM FUNCTION POINTS CALCULATION

The Aadhaar processing software is having 45 functions. In it, 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 UMMFP of Aadhaar processing system calculation is shown in Table 5.2.

Table 5.2: Unadjusted MMFP Calculations

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	1.0000000	Low	7	203
8	EIF	45	87	1.8913043	Low	5	230
						Total UMMFP	855

The Aadhaar processing software is having 45 functions. In it, 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.

5.4 COMPLEXITY ADJUSTMENT FACTOR (CAF)

The novelty, usage, complexity, distinct technologies, standardizations and policies used in the system are calculated using CAF factors. An analyzation of CAF calculation for Aadhaar processing system is shown in Table 5.3.

Table 5.3: MMCAF

S. No	Factors	Scale of Factors						
		Nil (0)	**Secondary (1)**	**Moderate (2)**	**Average (3)**	**Important (4)**	**Essential (5)**	**Value**
1	Whether backup is required to the system?	-	-	-	-	-	5	5
2	Whether data communication is important?	-	-	-	3	-	-	3
3	Whether it has any distributed processing?	-	-	-	3	-	-	3
4	Is representation complex?	-	-	-	-	4	-	4
5	Whether the system works in congested environment?	-	-	-	-	-	5	5
6	Does it require any online updating?	-	-	-	-	-	5	5
7	Whether the system has online input, output and operations?	-	-	-	-	-	5	5
8	Does it require any major file on online updating?	-	-	-	-	-	5	5
9	Does it work in multi environment?	-	-	-	3	-	-	3

S. No	Factors	Scale of Factors						
		Nil (0)	Secondary (1)	Moderate (2)	Average (3)	Important (4)	Essential (5)	Value
10	Is the internal operation critical?	-	-	-	3	-	-	3
11	Is the code designed to be reusable?	-	-	-	-	4	-	4
12	Whether the software is extensible?	-	-	2	-	-	-	2
13	Is it good for different organizations?	-	-	-	-	4	-	4
14	Does it permit user interactions?	-	-	-	-	4	-	4
15	Whether the system uses indexed or list data (single index or multi index)?	-	-	2	-	-	-	2
16	Whether the system uses more than one SDLC models?	-	-	2	-	-	-	2
17	Does the system using more than one programming language, DBMS, Web tools, Drivers etc.?	-	-	2	-	-	-	2

S. No	Factors	Scale of Factors						
		Nil (0)	**Secondary (1)**	**Moderate (2)**	**Average (3)**	**Important (4)**	**Essential (5)**	**Value**
18	Does the networking environment using more than one network topologies?	-	-	-	3	-	-	3
19	Does the system installed in different nations and uses different social, cultural, economic and environmental laws?	0	-	-	-	-	-	0
20	Does the system giving multiple forms of output?	-	-	-	-	-	5	5
21	Does the trial version and model version of software development affects the system?	-	-	-	3	-	-	3
22	Does User Interface influence 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 Aadhaar processing system is also derived based on these factors. The complexity adjustment factor of Aadhaar processing system is 82.

Modern Metrics Complexity Adjustment Factor (MMCAF)

The value of MMCAF is calculated using the Equation 4.1,

$$MMCAF = 0.25 + 0.01 * CAF$$

$$= 0.25 + 0.01 * 82$$

$$MMCAF = 1.07$$

Modern Metrics Size (MMSize)

MMSize of the software is calculated using the Equation 4.3,

$$MMSize = UMMFP * MMCAF$$

$$= 855 * 1.07$$

$$MMSize = 914.85 \text{ MMFP}$$

5.5 OTHER ESTIMATIONS

Modern Metrics Productivity Factor (MMPF)

MMPF is calculated using the Equation (4.4).

$$MMPF = \text{Total Hours required to Complete a project} / MMSize$$

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

MMPF = 6.29

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

Modern Metrics Effort (MME)

MME is calculated using the Equation (4.5).

MME = MMSize * MMPF

= 914.85 * 6.29

MME = 5754.40

(5754 Hours and 24 Minutes) Man-Hour required for completing the project Aadhaar processing system.

Modern Metrics Duration (MMD)

MMD is calculated using the Equation (4.6).

MMD = MME / (176 * number of human beings involved in the software development)

MMD = 5754.40 / (176 * 6)

MMD = 5.18 Months

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

Modern Metrics Cost (MMC)

MMC is calculated using the Equation (4.7).

MMC = The average remuneration of software developers * number of persons involved * MMPF+ Management 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* 6.29 + 210000

MMC = 4696715.37 (Indian Rupee)

Modern Metrics Unit Cost (MMUC)

MMUC is calculated using the Equation (4.8).

MMUC = MMC / MMSize

MMUC = 4696715.37/914.85

MMUC = 5133.86 (Indian Rupee)

Modern Metrics Price (MMP)

MMP is the market price of the software product. It is the sum of MM Cost, Maintenance cost of the firm and profit of the industry. The MMP is calculated using the Equation (5.1).

MM Price = MM Cost + (MM Cost + (MM Cost * Percentage of Maintenance cost)) * Percentage of profit of the industry (5.1)

Assuming, the maintenance cost as 40% of the MM cost and percentage of profit as 30%, then price of the software is calculated by

MMP = 4696715.37 + (4696715.37 + (4696715.37 *40/100)) * 30/100

= 4696715.37 + (4696715.37 + (4696715.37 * 0.4)) *0.3

MMP = 6669335.82 (Indian Rupee)

The productivity, effort, duration, cost and price of the Software Project Management prerequisites are calculated using Indian software industrial values.

5.6 SUMMARY

In Aadhar Processing System, all SPM factors like size, productivity, effort, duration, cost and price are estimated using different formulas derived using Modern Metrics. The final report of the MM Size estimation is shown in the Table 5.4.

Table 5.4: MM Report

MODERN METRICS SIZE REPORT		
		Date: 30-7-2018
Name of the Software	Aadhaar Processing System	
Total Number of Functions	45	
Functional Units		
S.No	Functional Units	Total Functional Units (TFU)
1	External Inputs (EI)	69
2	Internal Inputs (II)	100
3	External Outputs (EO)	124
4	Internal Operations (IO)	11
5	Data and Text (DT)	20
6	External Inquiries (EQ)	16
7	Internal Logical Files (ILF)	29
8	External Interface Files (EIF)	87
MMCAF		1.07
MMSize		914.85 MMFP
Number of Persons involved for development		6
Number of Months required to complete the project		5.18
Average remuneration per developer (₹)		118884.88
MM Productivity		6.29
MM Effort (Hours)		5754.40
MM Cost (₹)		4696715.37
MM Unit Cost (₹)		5133.86
Price of the Software (₹)		6669335.82
Time required for estimation		18Hours

CHAPTER 6

RESULT ANALYSIS

The Aadhaar processing system application is used for analyzing the performance of FPA and MM. The performance of traditional FPA and MM is analyzed with various parameters and the results are tabulated. To find the size of the software, only one person was involved. It took 3 days (19 Hours) for traditional FPA and 2 days (12 Hours) for MM.

6.1 TRADITIONAL FUNCTION POINT ANALYSIS (FPA) METHOD

In the traditional FPA method, all the five functional units are categorised based on its availability with the functions as low, average and high. The weightage factors and its values were discussed in Table 3.1.

The functional units of traditional FPA and UFP of Aadhaar processing system is shown in Table 6.1.

Table 6.1: Traditional FPA

S. No	Functional Units	Weighting Factor	Number of Functions	Weightage	Total Weightage	Total
1	EI	Low	10	3	30	152
		Average	8	4	32	
		High	15	6	90	
2	EO	Low	6	4	24	224
		Average	12	5	60	
		High	20	7	140	
3	EQ	Low	2	3	6	52
		Average	4	4	16	
		High	5	6	30	
4	ILF	Low	18	7	126	246
		Average	6	10	60	
		High	4	15	60	
5	EIF	Low	32	5	160	270
		Average	10	7	70	
		High	4	10	40	
					UFP	944

The complexity of the software is measured based on complexity factors listed in the CAF of FPA and is calculated using the Equation 3.3.

CAF of traditional FPA = 1.25

The size of the software is calculated using the Equation 3.4. The size is quantified using the unit Function Points (FP).

The size of software using FPA = 1180 FP

6.2 MODERN METRICS (MM) METHOD

The Table 6.2 displays all the functional units of MM and calculates the UMMFP of Aadhaar processing system.

Table 6.2: Updated UMMFP

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	1.0000000	Low	7	203
8	EIF	46	87	1.8913043	Low	5	230
Total UMMFP							855

The various Complexity factors are valued and calculated MMCAF

MMCAF = 1.07

MMSize = 914.85 MMFP

6.3 COMPARISON OF FPA AND MM WITH INTERMEDIATE RESULTS

The performance of MM over traditional FPA based on intermediate results of the calculation is shown in Figure 6.1.

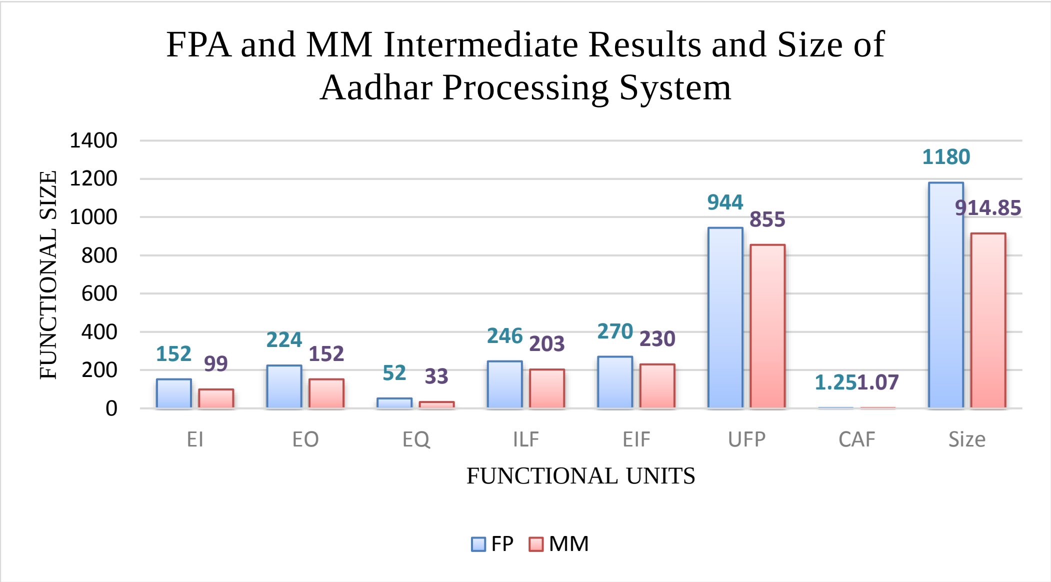


Figure 6.1: FPA and MM Intermediate Results

The unadjusted functional units in Aadhaar processing system are shown in Figure 6.1. All the intermediate result values of MM is comparatively less when compared to the values FPA. The inflated functional values of FPA are reduced in MM. It is the main reason for reduction in the size of intermediate results of functional units.

6.4 COMPARISON OF FPA AND MM WITH OTHER RESULTS

The Figure 6.2 shows the performance of MM over traditional FPA based on the results of application software Aadhaar processing system.

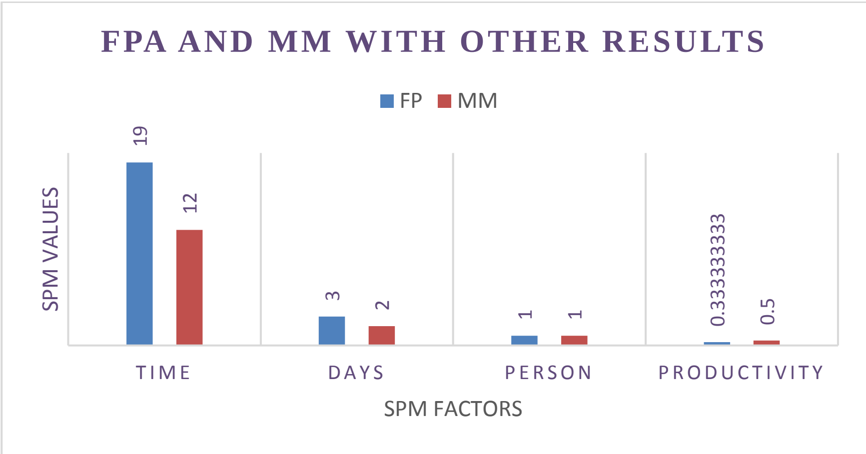


Figure 6.2: FPA and MM with other Results

From the Figure 6.2, it is shown that the size of the software is less in MM compared to FPA. Therefore, the factors of SPM based on software size are also less in MM over FPA. The results of MM are mostly same as that of industrial results.

6.5 ANALYSIS WITH OTHER SOFTWARE: CASE STUDY

The size and time required to find the size of the software in MM and FPA of different types of software are shown in the Table 6.3.

Table 6.3: FPA Size and MM Size

S. No	Software Name	MM		FPA	
		Size	Time in Hour	Size	Time in Hour
1	Aadhaar	914.85	12	1180	19
2	Online Shopping	517.88	8	481.25	8
3	Battle Ship	68.48	1	20	0.5
4	Calculator	31	0.66	22	0.5
5	Stack	28	0.58	16	0.41

The Aadhaar processing system is application software with more EI and less IO and DT. Therefore, the size in MM is less than FPA. The software like online shopping, battle ship, scientific calculator and stack implementation are having more number of internal operations and online shopping software having many databases and drivers. Therefore, the not required functional units in FPA like Internal Inputs, Internal Operations and Data and Text are considered in MM. It is increased size of MM over FPA. The detailed study of size analysis is present in Appendix 3 and Appendix 4.

The Figure 6.3 gives the size of the projects in a detailed manner. The size calculated using MM is same as that of industry based results of SPM factors like effort, time and cost.

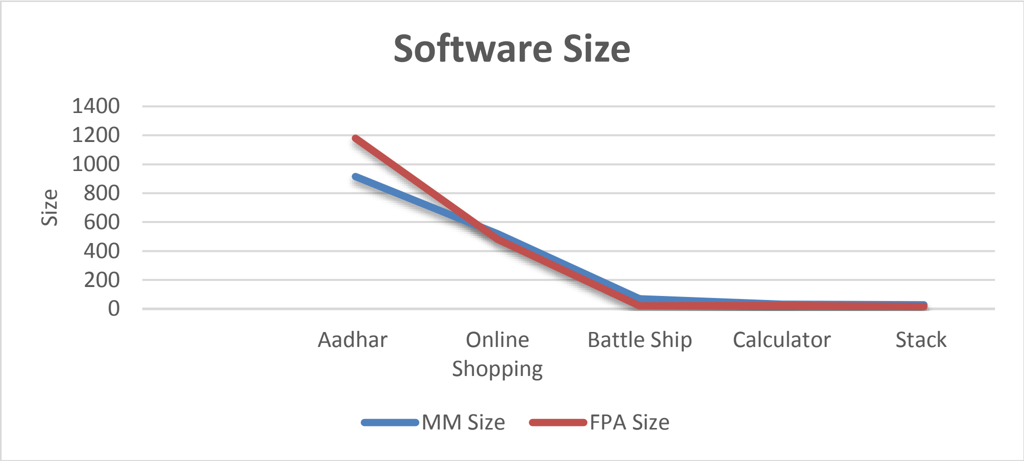


Figure 6.3: MM Size and FPA Size of Software

6.6 ANALYSIS WITH DIFFERENT FUNCTIONAL UNITS

In this proposed work, the size and SPM prerequisites like cost, time and effort of57 distinct functions have been analysed. These values were compared with actual market values (the detailed study is present in Appendix 3 and Appendix 4). The results of the study are displayed in the Figure 6.4 and Figure 6.5.

Analysis based on Size

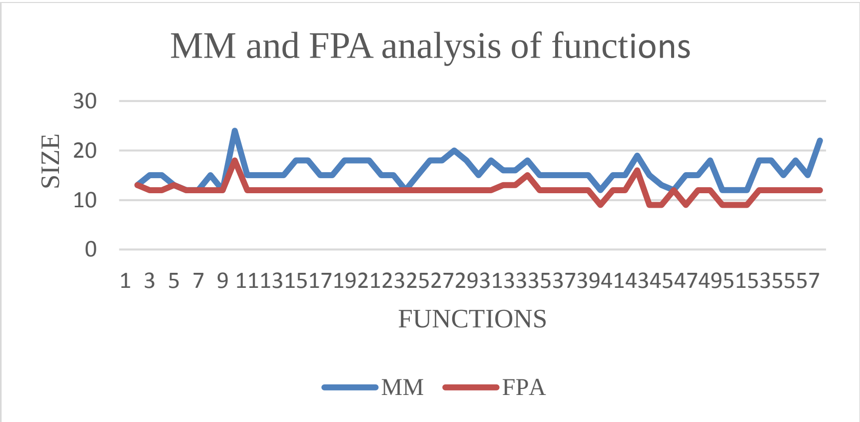


Figure 6.4: MM and FPA Size

Analysis based on Cost

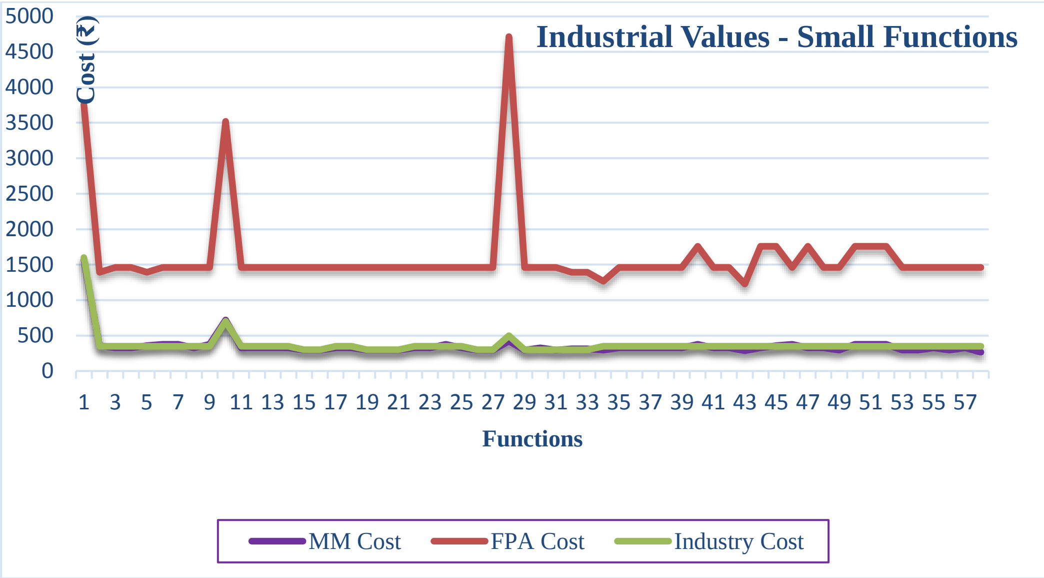


Figure 6.5: MM, FPA and Industry Values

The results received from above studies conclude that the defects per function point of FPA are 4.5 and it is near to zero in MM. The cost per function point is ₹. 10552.58 in FPA and it is ₹. 6389.37 in MM. The productivity of FPA is 2.123 and it is 1.055 in MM. The effort, cost and duration of MM is nearly same as that of Indian industrial values.

6.7 DIFFERENCE BETWEEN FPA AND MM

The various differences and merits of MM over FPA are listed in the Table 6.4.

Table 6.4: Differences between FPA and MM

S. No	Function Point Analysis	Modern Metrics
1	The traditional FPA methods only focused on the external input and output values of a function.	The MM not only considering external input and output, but also considers internal inputs, internal operations and data bases.
2	The traditional FPA is good for basic application software. But it is not giving the actual size for modern software like RDBMS, scientific, web based and design software.	The MM is a good estimation method for application, scientific, RDBMS and web based software.
3	The traditional FPA method does not consider indexed and listed data	The MM considers indexed and listed data.
4	The traditional FPA methods are not giving importance to databases. The modern software like cloud, data mining, Big data and Data analytics applications will not give actual size using traditional function points.	The MM is giving importance to the databases. It is good for modern software like cloud, data mining, Big data and Data analytics applications.
5	The defects per function point of traditional FPA are 4.5.	The defects per function point of MM are near to zero.
6	The traditional FPA is using only five functional elements. They are External Inputs, External Outputs,	The MM uses eight functional elements. They are External Inputs, Internal Inputs, External

S. No	Function Point Analysis	Modern Metrics
	External Inquiries, External Interface files and Internal Logical files.	Outputs, External Inquiries, Internal Operations, Data and Text, External Interface Files and Internal Logical files.
7	The traditional FPA methods are not considering SDLC of software.	The MM considers SDLC of software.
8	The traditional size estimation starts at the beginning phases of the SDLC. But it is not efficient for scientific applications.	The MM also does the size estimation at the beginning phases of the SDLC and is efficient for scientific applications.
9	The traditional FPA methods are calculating functional units from every function and treat it in a separate manner. So, inflated functional values are possible in the estimation.	The MM is calculating functional units from every function and treats it in collective manner. So, the inflated functional values are highly negligible in the estimation.
10	The traditional FPA methods are using only 14 complexity adjustment factors.	The MM uses 22 complexity adjustment factors based on traditional and modern software requirements.
11	The traditional FPA methods are not efficient for distributed and parallel processing systems.	The MM is giving importance for distributed and parallel processing systems with the help of complexity adjustment factors.

S. No	Function Point Analysis	Modern Metrics
12	The multiple forms of output are not getting importance in traditional FPA methods. So the effort of the developer is under estimated.	The multiple forms of output is getting importance through complexity adjustment factors in MM.
13	The GUI is not getting importance in traditional FPA.	The GUI is getting importance through complexity adjustment factors in MM.
14	The trial versions and beta version of the software is not influenced in traditional FPA.	The trial versions and beta version of the software is considered using complexity adjustment factors of MM.

6.8 SUMMARY

MM is an efficient method for finding the size of new modern software systems. The defects are negligible and all the user, developer, internal and external factors are analysed in MM. The results shown that MM is accurate when compared to all the existing sizing techniques.

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 are overwhelmed. The traditional FPA based sizing techniques are considering only the user perspectives but, the proposed MM technique considers user and developer perspectives. Thus, the defects in functional units of MM technique are 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. These 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 in its application. Thus, MM technique gives actual size of the scientific, AI, web pages and game playing software. The undefined functional units of design and modeling software like Computer Aided Designing, Computer Aided Modeling etc. shall be considered in the future studies.

FUTURE ENHANCEMENTS

There are some directions for future work that are worth exploring. These directions involve further calibrating the approach, extending it to other types of system sizing. MM is a Functional size measurement mechanism only considering modern software system. It is suitable for any kind of Management Information System (MIS), Scientific and Web applications too. The following are the Future enhancements that are applicable.

Inabilities of MM should be identified by examining different kind of applications like animations, design and modeling applications. It extends by modifying the components and makes it suitable for sizing a different kind of software system.

The extension of MM gives the functional and dysfunctional behaviors of social systems like governance, political parties, social and political unions, economic policies, organizations and so on.

The extended forms of MM not only give solution for software industry, it gives functional and dysfunctional values for all social, technical, political, scientific and economic systems of the world.

APPENDIX 1

FINDING THE FUNCTIONAL UNITS OF MM

The functional units of MM are explained in Chapter 4.1.2. The way of finding the functional units of software is explained using small software.

```
#include<iostream.h>
void get(void);
void add (int,int);
void put();
inta,b;
void main()
 {
   get();
   add(a,b);
   put();
 }
void get()
 {
cout<<"Enter a";
cin>>a;
  b=20;
}
void add(int a, int b)
{
int c= a+b;
}
void put()
{
cout<<"Sum ="<<c;
}
```

The functional units of above program are present in the Table A1.1.

Table A1.1: Sample Functional Units

S. No	**Functional Unit**	**Function**	**Variable**
1	EI	get()	a
2	II	get()	b
3	EO	put()	c
4	IO	add()	c
5	DT		
6	EQ		
7	ILF		iostream.h
8	EIF		

APPENDIX 2

EFFORT MODIFIERS

The effort modifiers are metric values of functional units in modern software system. The metrics of modern software system are categorized into low, average, high and very high. The effort modifiers are time variants in SDLC process. Each value of effort modifiers specifies that, this is the time in hour required to complete a functional unit of particular type in its all SDLC processes. The effort modifiers of functional types are calculated by using 57 distinct functions and 5 small and medium level software projects. All the values are calculated using real time applications. The metrics of functional units of MM are in the Table A2.1.

Table A2.1: 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 Table A2.2.

Table A2.2: 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

The real analysis of effort modifiers of External Input (EI) is present in the Table s from Table A2.3 to Table A2.6. The average time required to complete low EI is 3; average EI is 4; high EI is 6 and very high EI is 9. These are the actual low, average, high and very high metric values of EI.

Similarly, all the values of metrics of other functional units are calculated by using the tables from Table A2.7 to Table A2.24.

Table A2.3: Low EI

Low External Inputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Aadhaar	69	203
2	Calculator	1	2
3	Arithmetic Operations	2	2
4	Relational Operations	2	3
5	Logical Operators	3	4
6	Bitwise Operator	2	2
7	Increment and Decrement	1	1
8	sizeof function	3	3
9	getchar function	1	2
10	getche function	2	2

Low External Inputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
11	Roots of a quadratic equation	3	4
12	Even numbers	1	3
13	Number triangle	1	7
14	Number Pyramid	1	6
15	Factorial	1	3
16	Sum of digits	1	2
17	Sum of n numbers	1	4
18	Prime or not	1	3
19	Exponential Series	2	6
20	Sine series	2	6
21	Cos series	2	6
22	Reverse a number	1	3
23	Sum of series	1	3
24	Octal to decimal	1	3
25	Palindrome	1	6
26	Line of string	1	8
27	Substring detection	2	7
28	Substring removal	2	8
29	NCR	2	4
30	GCD	2	4
31	Fibonacci series	1	4
32	Matrix transpose	3	6
33	Matrix determinant	3	4
34	Insertion sort	2	6
35	Bubble sort	2	6

Low External Inputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
36	Linear Search	2	7
37	Function multiplication	2	3
38	strlen function	1	2
39	strcpy function	2	2
40	Union marks	3	6
41	Area of a circle	1	3
42	Biggest digit	1	3
43	Check armstrong	1	4
44	Sum of digits	1	4
45	Prime number	2	4
46	Arrange the digits	1	6
47	Leap year	1	4
48	Binary search tree	1	8
Total Low EI		145	
Total time to complete the EI		402	
Average time required to complete each EI		3	

Table A2.4: Average EI

Average External Inputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Online Shopping	41	168
2	Matrix addition	4	14
3	Matrix Subtraction	4	14
Total Average EI		49	
Total time to complete the EI		196	
Average time required to complete each EI		4	

Table A2.5: High EI

High External Inputs			
S. No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Matrix multiplication	6	36
2	Report card	8	40
Total High EI		14	
Total time to complete the EI		76	
Average time required to complete each EI		6	

Table A2.6: Very High EI

Very High External Inputs			
S. No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Battle ship	17	154
Total Very High EI		17	
Total time to complete the EI		154	
Average time required to complete each EI		9	

Table A2.7: Low II

Low Internal Inputs			
S.No	**Name of the Program**	**II**	**Time (Hour) Required to complete**
1	Online Shopping	9	32
2	Factorial	1	3
3	Sum of digits	1	3
4	Sum of n numbers	1	5
5	Exponential Series	2	5
6	Sine series	2	5
7	Cos series	2	5
8	Sum of series	2	5

Low Internal Inputs			
S.No	**Name of the Program**	**II**	**Time (Hour) Required to complete**
9	Line of string	2	6
10	Substring detection	3	6
11	NCR	1	4
12	Fibonacci series	2	2
13	Sum of n numbers	2	3
14	Preprocessor	3	1
15	Area of a circle	1	1
16	Biggest digit	1	1
17	Check armstrong	1	1
18	Sum of digits	3	2
19	Arrange the digits	1	1
Total Low II		40	
Total time to complete the II		91	
Average time required to complete each II		3	

Table A2.8: Average II

Average Internal Inputs			
S.No	**Name of the Program**	**II**	**Time (Hour) Required to complete**
1	Aadhaar	100	404
2	Substring removal	4	8
Total Average II		104	
Total time to complete the II		412	
Average time required to complete each II		4	

Table A2.9: Very High II

Very High Internal Inputs			
S.No	**Name of the Program**	**II**	**Time (Hour) Required to complete**
1	Calculator	19	169
Total Very High II		19	
Total time to complete the II		169	
Average time required to complete each II		9	

Table A2.10: Low EO

Low External Outputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Aadhaar	124	499
2	Online Shopping	41	172
3	Calculator	2	9
4	Relational Operations	1	3
5	Logical Operators	1	4
6	Increment and Decrement	4	3
7	sizeof function	3	4
8	getchar function	1	4
9	getche function	2	8
10	Roots of a quadratic equation	2	8
11	Even numbers	1	3
12	Number triangle	1	8
13	Number Pyramid	2	8
14	Factorial	2	8
15	Sum of digits	1	4

Low External Outputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
16	Sum of n numbers	1	4
17	Prime or not	2	7
18	Exponential Series	1	7
19	Sine series	1	8
20	Cos series	1	8
21	Reverse a number	1	4
22	Sum of series	1	7
23	Octal to decimal	1	4
24	Palindrome	1	4
25	Line of string	4	14
26	Substring detection	1	8
27	Substring removal	2	6
28	NCR	1	4
29	GCD	1	6
30	Fibonacci series	1	4
31	Matrix addition	1	8
32	Matrix Subtraction	1	8
33	Matrix multiplication	1	10
34	Matrix transpose	1	8
35	Matrix determinant	1	12
36	Insertion sort	1	4
37	Bubble sort	1	6
38	Linear Search	1	4
39	Function without arguments	1	4
40	Function multiplication	1	4

Low External Outputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
41	strlen function	1	3
42	strcpy function	2	3
43	Sum of n numbers	1	4
44	Structure student	4	4
45	Union marks	1	4
46	Preprocessor	3	6
47	Area of a circle	2	5
48	Biggest digit	1	8
49	Circle	1	4
50	Ellipse	1	4
51	Line	1	4
52	Check armstrong	1	6
53	Sum of digits	1	6
54	Prime number	1	4
55	Arrange the digits	1	5
56	Leap year	1	4
57	Binary search tree	1	7
Total Low EO		63	
Total time to complete the EO		283	
Average time required to complete each EO		4	

Table A2.11: Average EO

Average External Outputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Arithmetic Operations	5	24
2	Bitwise Operator	6	30
Total Average EO		11	
Total time to complete the EO		54	
Average time required to complete each EO		5	

Table A2.12: High EO

High External Outputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Report card	9	55
Total High EO		9	
Total time to complete the EO		55	
Average time required to complete each EO		7	

Table A2.13: Very High EO

Very High External Outputs			
S.No	**Name of the Program**	**EI**	**Time (Hour) Required to complete**
1	Battle ship	8	82
Total Very High EO		8	
Total time to complete the EO		82	
Average time required to complete each EO		10	

Table A2.14: Low IO

Low Internal Operations			
S.No	**Name of the Program**	**IO**	**Time (Hour) Required to complete**
1	Aadhaar	11	36
2	Online Shopping	33	102
3	Relational Operations	3	10
4	Logical Operators	3	11
5	getchar function	1	2
6	Roots of a quadratic equation	1	4
7	Even numbers	1	2
8	Number triangle	1	5
9	Number Pyramid	1	5
10	Factorial	1	2
11	Sum of digits	1	2
12	Prime or not	1	2
13	Exponential Series	1	4
14	Sine series	1	4
15	Cos series	1	4
16	Reverse a number	1	2
17	Palindrome	2	6
18	Line of string	2	7
19	Substring detection	2	8
20	NCR	1	2
21	GCD	1	2
22	Fibonacci series	1	4
23	Matrix addition	3	8

Low Internal Operations			
S.No	Name of the Program	IO	Time (Hour) Required to complete
24	Matrix Subtraction	3	8
25	Matrix multiplication	3	8
26	Matrix transpose	2	6
27	Matrix determinant	3	9
28	Insertion sort	2	8
29	Bubble sort	2	7
30	Linear Search	2	6
31	Function without arguments	1	2
32	Function multiplication	1	2
33	strlen function	1	2
34	strcpy function	1	2
35	Sum of n numbers	1	2
36	Preprocessor	1	2
37	Biggest digit	1	2
38	Circle	1	2
39	Ellipse	1	2
40	Line	1	2
41	Check armstrong	1	2
42	Sum of digits	1	2
43	Prime number	2	5
44	Arrange the digits	2	6
45	Leap year	2	6
Total Low IO		109	
Total time to complete the IO		327	
Average time required to complete each IO		3	

Table A2.15: Average IO

Average Internal Operations			
S. No	**Name of the Program**	**IO**	**Time (Hour) Required to complete**
1	Substring removal	4	16
Total Average IO		4	
Total time to complete the IO		16	
Average time required to complete each IO		4	

Table A2.16: High IO

High Internal Operations			
S.No	**Name of the Program**	**IO**	**Time (Hour) Required to complete**
1	Report card	6	32
2	Binary search tree	5	24
Total High IO		11	
Total time to complete the IO		56	
Average time required to complete each IO		6	

Table A2.17: Very High IO

Very High Internal Operations			
S.No	**Name of the Program**	**IO**	**Time (Hour) Required to complete**
1	Battle ship	88	791
2	Calculator	18	160
Total High IO		106	
Total time to complete the IO		951	
Average time required to complete each IO		9	

Table A2.18: Low DT

Low Data and Text			
S.No	**Name of the Program**	**DT**	**Time (Hour) Required to complete**
1	Aadhaar	20	68
Total Low DT		20	
Total time to complete the DT		68	
Average time required to complete each DT		4	

Table A2.19: Very High DT

Very High Data and Text			
S.No	**Name of the Program**	**DT**	**Time (Hour) Required to complete**
1	Online Shopping	9	85
Total Very High DT		9	
Total time to complete the DT		85	
Average time required to complete each DT		10	

Table A2.20: Low EQ

Low External Inquiries			
S.No	**Name of the Program**	**EQ**	**Time (Hour) Required to complete**
1	Aadhaar	16	43
Total Low EQ		16	
Total time to complete the EQ		43	
Average time required to complete each EQ		3	

Table A2.21: Very High EQ

Very High External Inquiries			
S.No	**Name of the Program**	**EQ**	**Time (Hour) Required to complete**
1	Online Shopping	7	57
Total Very High EQ		7	
Total time to complete the EQ		57	
Average time required to complete each EQ		9	

Table A2.22: Low ILF

Low Internal Logical Files			
S.No	**Name of the Program**	**ILF**	**Time (Hour) Required to complete**
1	Aadhaar	29	201
2	Online Shopping	33	232
3	Calculator	1	6
Total Low ILF		63	
Total time to complete the ILF		439	
Average time required to complete each ILF		7	

Table A2.23: Low EIF

Low External Interface Files			
S.No	**Name of the Program**	**EIF**	**Time (Hour) Required to complete**
1	Aadhaar	87	437
2	Battle ship	3	18
3	Online Shopping	37	185
4	Arithmetic Operations	1	4
5	Relational Operations	1	4

Low External Interface Files			
S.No	**Name of the Program**	**EIF**	**Time (Hour) Required to complete**
6	Logical Operators	1	4
7	Bitwise Operator	1	4
8	Increment and Decrement	1	4
9	sizeof function	2	7
10	getchar function	2	9
11	getche function	2	9
12	Report card	2	14
13	Roots of a quadratic equation	3	12
14	Even numbers	2	9
15	Number triangle	2	12
16	Number Pyramid	2	12
17	Factorial	2	12
18	Sum of digits	2	12
19	Sum of n numbers	2	9
20	Prime or not	3	12
21	Exponential Series	3	16
22	Sine series	3	16
23	Cos series	3	16
24	Reverse a number	2	10
25	Sum of series	2	10
26	Octal to decimal	2	10
27	Palindrome	3	15
28	Line of string	3	14
29	Substring detection	3	15
30	Substring removal	3	15

Low External Interface Files			
S.No	**Name of the Program**	**EIF**	**Time (Hour) Required to complete**
31	NCR	3	15
32	GCD	2	10
33	Fibonacci series	2	10
34	Matrix addition	2	10
35	Matrix Subtraction	2	10
36	Matrix multiplication	2	10
37	Matrix transpose	2	10
38	Matrix determinant	2	10
39	Insertion sort	2	10
40	Bubble sort	2	10
41	Linear Search	2	10
42	Function without arguments	2	10
43	Function multiplication	2	10
44	strlen function	3	12
45	strcpy function	3	12
46	Sum of n numbers	2	10
47	structure student	2	10
48	Union marks	2	10
49	Preprocessor	2	10
50	Area of a circle	2	10
51	Biggest digit	2	10
52	Circle	2	10
53	Ellipse	2	10
54	Line	2	10
55	Check armstrong	3	10

Low External Interface Files			
S.No	**Name of the Program**	**EIF**	**Time (Hour) Required to complete**
56	Sum of digits	2	10
57	Prime number	3	10
58	Arrange the digits	2	10
59	Leap year	2	10
60	Binary search tree	2	10
Total Low EIF		250	
Total time to complete the EIF		1235	
Average time required to complete each EIF		5	

Table A2.24: Average EIF

Average External Interface Files			
S.No	**Name of the Program**	**EIF**	**Time (Hour) Required to complete**
1	Calculator	6	43
Total Average EIF		6	
Total time to complete the EIF		41	
Average time required to complete each EIF		7	

APPENDIX 3

FUNCTIONAL UNITS, UNADJUSTED FUNCTION POINTS AND SIZE OF MM AND FPA WITH SAMPLE FUNCTIONS

In this analysis 57 functions, 5 medium and small software are used. The way of finding functional units, unadjusted function points, complexity adjustment factor and size of the software of MM are present in Chapter 4. The way of finding functional units, unadjusted function points, complexity adjustment factor and size of the software of FPA are present in Chapter 3.4.3.

The functional units, unadjusted function points, complexity values and size of 57 functions of MM is present in the Table A3.1. The functional units, unadjusted function points, complexity values and size of 57 functions of FPA is present in the Table A3.2. The MM and FPA size of 5 small and medium level software present in Table A3.3.

The MM considers all the IO of software. The size of some functions and small software like battle ship and calculator are high in MM is comparatively the size in FPA. This has reduced the defects in function point calculation of MM.

The MM considers the database of the software. So, the size of online shopping software is high compared to FPA. MM considers the work behind the database and all text files.

The software Aadhaar gives less size in MM because it has more number of EI and EO. The inflated function points of FPA are removed in MM.

Table A3.1: Size of MM

S. No	Program Name	Functional Units of MM								Unadjusted MM								MM Size		
		EI	**EO**	**EQ**	**ILF**	**EIF**	**II**	**IO**	**DT**	**EI**	**EO**	**EQ**	**ILF**	**EIF**	**II**	**IO**	**DT**	**TFU**	**CAF**	**Size**
1	Arithmetic Operations	2	5	0	0	1	0		0	3	5	0	0	5	0	0	0	13	1	13
2	Relational Operations	2	1	0	0	1	0	3	0	3	4	0	0	5	0	3	0	15	1	15
3	Logical Operators	3	1	0	0	1	0	3	0	3	4	0	0	5	0	3	0	15	1	15
4	Bitwise Operator	2	6	0	0	1	0	0	0	3	5	0	0	5	0	0	0	13	1	13
5	Increment and Decrement	1	4	0	0	1	0	0	0	3	4	0	0	5	0	0	0	12	1	12
6	sizeof function	3	3	0	0	2	0	0	0	3	4	0	0	5	0	0	0	12	1	12
7	getchar function	1	1	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
8	getche function	2	2	0	0	2	0	0	0	3	4	0	0	5	0	0	0	12	1	12
9	Report card	8	7	0	0	2	0	6	0	6	7	0	0	5	0	6	0	24	1	24
10	Roots of a quadratic equation	3	2	0	0	3	0	1	0	3	4	0	0	5	0	3	0	15	1	15
11	Even numbers	1	1	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
12	Number triangle	1	1	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
13	Number Pyramid	1	2	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
14	Factorial	1	2	0	0	2	1	1	0	3	4	0	0	5	3	3	0	18	1	18
15	Sum of digits	1	1	0	0	2	1	1	0	3	4	0	0	5	3	3	0	18	1	18
16	Sum of n numbers	1	1	0	0	2	1		0	3	4	0	0	5	3	0	0	15	1	15
17	Prime or not	1	2	0	0	3		1	0	3	4	0	0	5	0	3	0	15	1	15
18	Exponential Series	2	1	0	0	3	2	1	0	3	4	0	0	5	3	3	0	18	1	18

S. No	Program Name	Functional Units of MM								Unadjusted MM								MM Size		
		EI	EO	EQ	ILF	EIF	II	IO	DT	EI	EO	EQ	ILF	EIF	II	IO	DT	TFU	CAF	Size
19	Sine series	2	1	0	0	3	2	1	0	3	4	0	0	5	3	3	0	18	1	18
20	Cos series	2	1	0	0	3	2	1	0	3	4	0	0	5	3	3	0	18	1	18
21	Reverse a number	1	1	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
22	Sum of series	1	1	0	0	2	2	0	0	3	4	0	0	5	3	0	0	15	1	15
23	Octal to decimal	1	1	0	0	2	0	0	0	3	4	0	0	5	0	0	0	12	1	12
24	Palindrome	1	1	0	0	3	0	2	0	3	4	0	0	5	0	3	0	15	1	15
25	Line of string	1	4	0	0	3	2	2	0	3	4	0	0	5	3	3	0	18	1	18
26	Substring detection	2	1	0	0	3	3	2	0	3	4	0	0	5	3	3	0	18	1	18
27	Substring removal	2	2	0	0	3	4	4	0	3	4	0	0	5	4	4	0	20	1	20
28	NCR	2	1	0	0	3	1	1	0	3	4	0	0	5	3	3	0	18	1	18
29	GCD	2	1	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
30	Fibonacci series	1	1	0	0	2	2	1	0	3	4	0	0	5	3	3	0	18	1	18
31	Matrix addition	4	1	0	0	2	0	3	0	4	4	0	0	5	0	3	0	16	1	16
32	Matrix Subtraction	4	1	0	0	2	0	3	0	4	4	0	0	5	0	3	0	16	1	16
33	Matrix multiplication	6	1	0	0	2	0	3	0	6	4	0	0	5	0	3	0	18	1	18
34	Matrix transpose	3	1	0	0	2	0	2	0	3	4	0	0	5	0	3	0	15	1	15
35	Matrix determinant	3	1	0	0	2	0	3	0	3	4	0	0	5	0	3	0	15	1	15
36	Insertion sort	2	1	0	0	2	0	2	0	3	4	0	0	5	0	3	0	15	1	15
37	Bubble sort	2	1	0	0	2	0	2	0	3	4	0	0	5	0	3	0	15	1	15
38	Linear Search	2	1	0	0	2	0	2	0	3	4	0	0	5	0	3	0	15	1	15

S. No	Program Name	Functional Units of MM								Unadjusted MM								MM Size		
		EI	EO	EQ	ILF	EIF	II	IO	DT	EI	EO	EQ	ILF	EIF	II	IO	DT	TFU	CAF	Size
39	Function without arguments	0	1	0	0	2	0	1	0	0	4	0	0	5	0	3	0	12	1	12
40	Function multiplication	2	1	0	0	2	0	1	0	3	4	0	0	5	0	3	0	15	1	15
41	strlen function	1	1	0	0	3	0	1	0	3	4	0	0	5	0	3	0	15	1	15
42	strcpy function	2	2	0	0	3	0	1	0	3	8	0	0	5	0	3	0	19	1	19
43	Sum of n numbers	0	1	0	0	2	2	1	0	0	4	0	0	5	3	3	0	15	1	15
44	structure student	0	4	0	0	2	4	0	0	0	4	0	0	5	4	0	0	13	1	13
45	Union marks	3	1	0	0	2	0	0	0	3	4	0	0	5	0	0	0	12	1	12
46	Preprocessor	0	3	0	0	2	3	1	0	0	4	0	0	5	3	3	0	15	1	15
47	Area of a circle	1	2	0	0	2	1	0	0	3	4	0	0	5	3	0	0	15	1	15
48	Biggest digit	1	1	0	0	2	1	1	0	3	4	0	0	5	3	3	0	18	1	18
49	Circle	0	1	0	0	2	0	1	0	0	4	0	0	5	0	3	0	12	1	12
50	Ellipse	0	1	0	0	2	0	1	0	0	4	0	0	5	0	3	0	12	1	12
51	Line	0	1	0	0	2	0	1	0	0	4	0	0	5	0	3	0	12	1	12
52	Check armstrong	1	1	0	0	3	1	1	0	3	4	0	0	5	3	3	0	18	1	18
53	Sum of digits	1	1	0	0	2	3	1	0	3	4	0	0	5	3	3	0	18	1	18
54	Prime number	2	1	0	0	3	0	2	0	3	4	0	0	5	0	3	0	15	1	15
55	Arrange the digits	1	1	0	0	2	1	2	0	3	4	0	0	5	3	3	0	18	1	18
56	Leap year	1	1	0	0	2	0	2	0	3	4	0	0	5	0	3	0	15	1	15
57	Binary search tree	1	1	0	0	2	4	5	0	3	4	0	0	5	4	6	0	22	1	22
Total		**97**	**93**	**0**	**0**	**123**	**43**	**80**	**0**	**158**	**237**	**0**	**0**	**285**	**66**	**145**	**0**	**891**	**57**	**891**

Table A3.2: Size of FPA

S. No	**Program Name**	**Functional Units of FPA**					**Unadjusted FPA**					**FPA Size**		
		EI	EO	EQ	ILF	EIF	EI	EO	EQ	ILF	EIF	TFU	CAF	Size
1	Arithmetic Operations	2	5	0	0	1	3	5	0	0	5	13	1	13
2	Relational Operations	2	1	0	0	1	3	4	0	0	5	12	1	12
3	Logical Operators	3	1	0	0	1	3	4	0	0	5	12	1	12
4	Bitwise Operator	2	6	0	0	1	3	5	0	0	5	13	1	13
5	Increment and Decrement	1	4	0	0	1	3	4	0	0	5	12	1	12
6	sizeof function	3	3	0	0	2	3	4	0	0	5	12	1	12
7	getchar function	1	1	0	0	2	3	4	0	0	5	12	1	12
8	getche function	2	2	0	0	2	3	4	0	0	5	12	1	12
9	Report card	8	7	0	0	2	6	7	0	0	5	18	1	18
10	Roots of a quadratic equation	3	2	0	0	3	3	4	0	0	5	12	1	12
11	Even numbers	1	1	0	0	2	3	4	0	0	5	12	1	12
12	Number triangle	1	1	0	0	2	3	4	0	0	5	12	1	12
13	Number Pyramid	1	2	0	0	2	3	4	0	0	5	12	1	12
14	Factorial	1	2	0	0	2	3	4	0	0	5	12	1	12

S. No	Program Name	Functional Units of FPA					Unadjusted FPA					FPA Size		
		EI	EO	EQ	ILF	EIF	EI	EO	EQ	ILF	EIF	TFU	CAF	Size
15	Sum of digits	1	1	0	0	2	3	4	0	0	5	12	1	12
16	Sum of n numbers	1	1	0	0	2	3	4	0	0	5	12	1	12
17	Prime or not	1	2	0	0	3	3	4	0	0	5	12	1	12
18	Exponential Series	2	1	0	0	3	3	4	0	0	5	12	1	12
19	Sine series	2	1	0	0	3	3	4	0	0	5	12	1	12
20	Cos series	2	1	0	0	3	3	4	0	0	5	12	1	12
21	Reverse a number	1	1	0	0	2	3	4	0	0	5	12	1	12
22	Sum of series	1	1	0	0	2	3	4	0	0	5	12	1	12
23	Octal to decimal	1	1	0	0	2	3	4	0	0	5	12	1	12
24	Palindrome	1	1	0	0	3	3	4	0	0	5	12	1	12
25	Line of string	1	4	0	0	3	3	4	0	0	5	12	1	12
26	Substring detection	2	1	0	0	3	3	4	0	0	5	12	1	12
27	Substring removal	2	2	0	0	3	3	4	0	0	5	12	1	12
28	NCR	2	1	0	0	3	3	4	0	0	5	12	1	12
29	GCD	2	1	0	0	2	3	4	0	0	5	12	1	12
30	Fibonacci series	1	1	0	0	2	3	4	0	0	5	12	1	12

S. No	Program Name	Functional Units of FPA					Unadjusted FPA					FPA Size		
		EI	EO	EQ	ILF	EIF	EI	EO	EQ	ILF	EIF	TFU	CAF	Size
31	Matrix addition	4	1	0	0	2	4	4	0	0	5	13	1	13
32	Matrix Subtraction	4	1	0	0	2	4	4	0	0	5	13	1	13
33	Matrix multiplication	6	1	0	0	2	6	4	0	0	5	15	1	15
34	Matrix transpose	3	1	0	0	2	3	4	0	0	5	12	1	12
35	Matrix determinant	3	1	0	0	2	3	4	0	0	5	12	1	12
36	Insertion sort	2	1	0	0	2	3	4	0	0	5	12	1	12
37	Bubble sort	2	1	0	0	2	3	4	0	0	5	12	1	12
38	Linear Search	2	1	0	0	2	3	4	0	0	5	12	1	12
39	Function without arguments		1	0	0	2	0	4	0	0	5	9	1	9
40	Function multiplication	2	1	0	0	2	3	4	0	0	5	12	1	12
41	strlen function	1	1	0	0	3	3	4	0	0	5	12	1	12
42	strcpy function	2	2	0	0	3	3	8	0	0	5	16	1	16
43	Sum of n numbers		1	0	0	2	0	4	0	0	5	9	1	9
44	structure student		4	0	0	2	0	4	0	0	5	9	1	9
45	Union marks	3	1	0	0	2	3	4	0	0	5	12	1	12

S. No	Program Name	Functional Units of FPA					Unadjusted FPA					FPA Size		
		EI	EO	EQ	ILF	EIF	EI	EO	EQ	ILF	EIF	TFU	CAF	Size
46	Preprocessor		3	0	0	2	0	4	0	0	5	9	1	9
47	Area of a circle	1	2	0	0	2	3	4	0	0	5	12	1	12
48	Biggest digit	1	1	0	0	2	3	4	0	0	5	12	1	12
49	Circle		1	0	0	2	0	4	0	0	5	9	1	9
50	Ellipse		1	0	0	2	0	4	0	0	5	9	1	9
51	Line		1	0	0	2	0	4	0	0	5	9	1	9
52	Check armstrong	1	1	0	0	3	3	4	0	0	5	12	1	12
53	Sum of digits	1	1	0	0	2	3	4	0	0	5	12	1	12
54	Prime number	2	1	0	0	3	3	4	0	0	5	12	1	12
55	Arrange the digits	1	1	0	0	2	3	4	0	0	5	12	1	12
56	Leap year	1	1	0	0	2	3	4	0	0	5	12	1	12
57	Binary search tree	1	1	0	0	2	3	4	0	0	5	12	1	12
Total		**97**	**93**	**0**	**0**	**123**	**158**	**237**	**0**	**0**	**285**	**680**	**57**	**680**

Table A3.3: Size of MM and Size of FPA

S. No	Program Name	MM Size	FPA Size
1	Aadhaar	914.85	1180
2	Battle ship	84.53	20
3	Online Shopping	517.88	481
4	Calculator	58	46
5	Functions (57 Functions)	596	680

APPENDIX 4

SOFTWARE PREREQUISITES OF MM AND FPA

The prerequisites of SPM are effort, time and cost. The way of finding prerequisites are present in Chapter 4.3. The comparison of MM software prerequisites to industrial values is present in Table A4.1. The comparison of FPA software prerequisites to industrial values present in Table A4.2.

These studies thereby reflect that the size of MM is same as that of industrial values. Hence, MM is the best method for estimating the size of software compared to FPA.

Table A4.1: MM vs Industrial Values

S. No	Program Name	Modern Metrics					Industrial Values	
		MM Size	**Productivity**	**Effort (Man-Hours)**	**Duration (Month)**	**Cost**	**Duration (Month)**	**Cost**
1	Aadhaar	914.85	6.3	5763.56	4.68	6584760	4.68	6600000
2	Battle ship	84.53	17.89	1512.24	1.23	7086460.5	1.23	7100000
3	Online Shopping	517.88	8.65	4479.66	3.64	5764850	3.64	5800000
4	Functions	596	1.61	959.56	0.78	109746	0.78	110000
5	Calculator	58	0.41	23.78	0.02	567	0.02	600
6	Arithmetic Operations	13	0.62	8.06	0.01	357	0.01	350
7	Relational Operations	15	0.53	7.95	0.01	325.5	0.01	350
8	Logical Operators	15	0.53	7.95	0.01	325.5	0.01	350
9	Bitwise Operator	13	0.62	8.06	0.01	357	0.01	350
10	Increment and Decrement	12	0.67	8.04	0.01	374.5	0.01	350
11	sizeof function	12	0.67	8.04	0.01	374.5	0.01	350
12	getchar function	15	0.53	7.95	0.01	325.5	0.01	350
13	getche function	12	0.67	8.04	0.01	374.5	0.01	350

S. No	Program Name	Modern Metrics					Industrial Values	
		MM Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
14	Report card	24	0.67	16.08	0.01	718.5	0.01	700
15	Roots of a quadratic equation	15	0.53	7.95	0.01	325.5	0.01	350
16	Even numbers	15	0.53	7.95	0.01	325.5	0.01	350
17	Number triangle	15	0.53	7.95	0.01	325.5	0.01	350
18	Number Pyramid	15	0.53	7.95	0.01	325.5	0.01	350
19	Factorial	18	0.44	7.92	0.01	294	0.01	300
20	Sum of digits	18	0.44	7.92	0.01	294	0.01	300
21	Sum of n numbers	15	0.53	7.95	0.01	325.5	0.01	350
22	Prime or not	15	0.53	7.95	0.01	325.5	0.01	350
23	Exponential Series	18	0.44	7.92	0.01	294	0.01	300
24	Sine series	18	0.44	7.92	0.01	294	0.01	300
25	Cos series	18	0.44	7.92	0.01	294	0.01	300
26	Reverse a number	15	0.53	7.95	0.01	325.5	0.01	350
27	Sum of series	15	0.53	7.95	0.01	325.5	0.01	350

S. No	Program Name	Modern Metrics					Industrial Values	
		MM Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
28	Octal to decimal	12	0.67	8.04	0.01	374.5	0.01	350
29	Palindrome	15	0.53	7.95	0.01	325.5	0.01	350
30	Line of string	18	0.44	7.92	0.01	294	0.01	300
31	Substring detection	18	0.44	7.92	0.01	294	0.01	300
32	Substring removal	20	0.8	16	0.01	420	0.01	500
33	NCR	18	0.44	7.92	0.01	294	0.01	300
34	GCD	15	0.53	7.95	0.01	325.5	0.01	300
35	Fibonacci series	18	0.44	7.92	0.01	294	0.01	300
36	Matrix addition	16	0.5	8	0.01	315	0.01	300
37	Matrix Subtraction	16	0.5	8	0.01	315	0.01	300
38	Matrix multiplication	18	0.44	7.92	0.01	294	0.01	350
39	Matrix transpose	15	0.53	7.95	0.01	325.5	0.01	350
40	Matrix determinant	15	0.53	7.95	0.01	325.5	0.01	350
41	Insertion sort	15	0.53	7.95	0.01	325.5	0.01	350

S. No	Program Name	Modern Metrics					Industrial Values	
		MM Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
42	Bubble sort	15	0.53	7.95	0.01	325.5	0.01	350
43	Linear Search	15	0.53	7.95	0.01	325.5	0.01	350
44	Function without arguments	12	0.67	8.04	0.01	374.5	0.01	350
45	Function multiplication	15	0.53	7.95	0.01	325.5	0.01	350
46	strlen function	15	0.53	7.95	0.01	325.5	0.01	350
47	strcpy function	19	0.42	7.98	0.01	287	0.01	350
48	Sum of n numbers	15	0.53	7.95	0.01	325.5	0.01	350
49	structure student	13	0.62	8.06	0.01	357	0.01	350
50	Union marks	12	0.67	8.04	0.01	374.5	0.01	350
51	Preprocessor	15	0.53	7.95	0.01	325.5	0.01	350
52	Area of a circle	15	0.53	7.95	0.01	325.5	0.01	350
53	Biggest digit	18	0.44	7.92	0.01	294	0.01	350
54	Circle	12	0.67	8.04	0.01	374.5	0.01	350
55	Ellipse	12	0.67	8.04	0.01	374.5	0.01	350

S. No	Program Name	Modern Metrics					Industrial Values	
		MM Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
56	Line	12	0.67	8.04	0.01	374.5	0.01	350
57	Check armstrong	18	0.44	7.92	0.01	294	0.01	350
58	Sum of digits	18	0.44	7.92	0.01	294	0.01	350
59	Prime number	15	0.53	7.95	0.01	325.5	0.01	350
60	Arrange the digits	18	0.44	7.92	0.01	294	0.01	350
61	Leap year	15	0.53	7.95	0.01	325.5	0.01	350
62	Binary search tree	22	0.36	7.92	0.01	266	0.01	350
Total		3062.26	65.41	13209	10.92	19565400	10.92	19630450
Average		49.39	1.055	213.048	0.17613	315570.9677	0.176129032	316620.1613

Salary per developer per month	35000
Cost per MM	6389.37
Development Time per MM (Days)	4.31
Productivity per day per person	0.23

Table A4.2: FPA vs Industrial Values

S. No	Program Name	FPA Metrics					Industrial Values	
		FPA Size	**Productivity**	**Effort (Man-Hours)**	**Duration (Month)**	**Cost**	**Duration (Month)**	**Cost**
1	Aadhaar	1180	4.881355932	5758.4	4.674026	5183890.91	4.68	6600000
2	Battle ship	20	75.6	1512	1.227273	19587272.73	1.23	7100000
3	Online Shopping	481	9.313929314	4478.11	3.63483	7414780.23	3.64	5800000
4	Functions	680	1.411764706	958.8	0.778247	296100	0.78	110000
5	Calculator	46	0.52173913	23.92	0.019416	3758.18	0.02	600
6	Arithmetic Operations	13	0.615384615	8.06	0.006542	1395	0.01	350
7	Relational Operations	12	0.666666667	8.04	0.006526	1461.82	0.01	350
8	Logical Operators	12	0.666666667	8.04	0.006526	1461.82	0.01	350
9	Bitwise Operator	13	0.615384615	8.06	0.006542	1395	0.01	350
10	Increment and Decrement	12	0.666666667	8.04	0.006526	1461.82	0.01	350
11	sizeof function	12	0.666666667	8.04	0.006526	1461.82	0.01	350
12	getchar function	12	0.666666667	8.04	0.006526	1461.82	0.01	350
13	getche function	12	0.666666667	8.04	0.006526	1461.82	0.01	350

S. No	Program Name	FPA Metrics					Industrial Values	
		FPA Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
14	Report card	18	0.888888889	16.02	0.013003	3519.55	0.01	700
15	Roots of a quadratic equation	12	0.666666667	8.04	0.006526	1461.82	0.01	350
16	Even numbers	12	0.666666667	8.04	0.006526	1461.82	0.01	350
17	Number triangle	12	0.666666667	8.04	0.006526	1461.82	0.01	350
18	Number Pyramid	12	0.666666667	8.04	0.006526	1461.82	0.01	350
19	Factorial	12	0.666666667	8.04	0.006526	1461.82	0.01	300
20	Sum of digits	12	0.666666667	8.04	0.006526	1461.82	0.01	300
21	Sum of n numbers	12	0.666666667	8.04	0.006526	1461.82	0.01	350
22	Prime or not	12	0.666666667	8.04	0.006526	1461.82	0.01	350
23	Exponential Series	12	0.666666667	8.04	0.006526	1461.82	0.01	300
24	Sine series	12	0.666666667	8.04	0.006526	1461.82	0.01	300
25	Cos series	12	0.666666667	8.04	0.006526	1461.82	0.01	300
26	Reverse a number	12	0.666666667	8.04	0.006526	1461.82	0.01	350
27	Sum of series	12	0.666666667	8.04	0.006526	1461.82	0.01	350

S. No	Program Name	FPA Metrics					Industrial Values	
		FPA Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
28	Octal to decimal	12	0.666666667	8.04	0.006526	1461.82	0.01	350
29	Palindrome	12	0.666666667	8.04	0.006526	1461.82	0.01	350
30	Line of string	12	0.666666667	8.04	0.006526	1461.82	0.01	300
31	Substring detection	12	0.666666667	8.04	0.006526	1461.82	0.01	300
32	Substring removal	12	1.333333333	15.96	0.012955	4715.45	0.01	500
33	NCR	12	0.666666667	8.04	0.006526	1461.82	0.01	300
34	GCD	12	0.666666667	8.04	0.006526	1461.82	0.01	300
35	Fibonacci series	12	0.666666667	8.04	0.006526	1461.82	0.01	300
36	Matrix addition	13	0.615384615	8.06	0.006542	1395	0.01	300
37	Matrix Subtraction	13	0.615384615	8.06	0.006542	1395	0.01	300
38	Matrix multiplication	15	0.533333333	7.95	0.006453	1264.77	0.01	350
39	Matrix transpose	12	0.666666667	8.04	0.006526	1461.82	0.01	350
40	Matrix determinant	12	0.666666667	8.04	0.006526	1461.82	0.01	350
41	Insertion sort	12	0.666666667	8.04	0.006526	1461.82	0.01	350

S. No	Program Name	FPA Metrics					Industrial Values	
		FPA Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
42	Bubble sort	12	0.666666667	8.04	0.006526	1461.82	0.01	350
43	Linear Search	12	0.666666667	8.04	0.006526	1461.82	0.01	350
44	Function without arguments	9	0.888888889	8.01	0.006502	1759.77	0.01	350
45	Function multiplication	12	0.666666667	8.04	0.006526	1461.82	0.01	350
46	strlen function	12	0.666666667	8.04	0.006526	1461.82	0.01	350
47	strcpy function	16	0.5	8	0.006494	1227.27	0.01	350
48	Sum of n numbers	9	0.888888889	8.01	0.006502	1759.77	0.01	350
49	structure student	9	0.888888889	8.01	0.006502	1759.77	0.01	350
50	Union marks	12	0.666666667	8.04	0.006526	1461.82	0.01	350
51	Preprocessor	9	0.888888889	8.01	0.006502	1759.77	0.01	350
52	Area of a circle	12	0.666666667	8.04	0.006526	1461.82	0.01	350
53	Biggest digit	12	0.666666667	8.04	0.006526	1461.82	0.01	350
54	Circle	9	0.888888889	8.01	0.006502	1759.77	0.01	350
55	Ellipse	9	0.888888889	8.01	0.006502	1759.77	0.01	350

S. No	Program Name	FPA Metrics					Industrial Values	
		FPA Size	Productivity	Effort (Man-Hours)	Duration (Month)	Cost	Duration (Month)	Cost
56	Line	9	0.888888889	8.01	0.006502	1759.77	0.01	350
57	Check armstrong	12	0.666666667	8.04	0.006526	1461.82	0.01	350
58	Sum of digits	12	0.666666667	8.04	0.006526	1461.82	0.01	350
59	Prime number	12	0.666666667	8.04	0.006526	1461.82	0.01	350
60	Arrange the digits	12	0.666666667	8.04	0.006526	1461.82	0.01	350
61	Leap year	12	0.666666667	8.04	0.006526	1461.82	0.01	350
62	Binary search tree	12	0.666666667	8.04	0.006526	1461.82	0.01	350
Total		3087	131.6681053	13205.15	10.71847	32575823.92	10.92	19630450
Average		49.79032	2.123679118	212.9863	0.172878	525416.5148	0.176129	316620.2

Salary per developer per month	35000
Cost per MM	10552.58
Development Time per MM (Days)	4.28
Productivity per day per person	0.23

REFERENCES

1. Angelica Toffano Calazans, Kathia Marcal de Oliveira and Rildo Ribeiro Dos Santos (2004), ”Adapting Function Point Analysis to Estimate Data Mart Size”, METRICS’04, IEEE, pp.1530 – 1435.

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

3. Barry W. Boehm (1981), “Software Engineering Economics”. Prentice-Hall.

4. Barry W. Boehm (1986), "Software Engineering Economics", Tutorial, Software Management, Third Edition, Washington, DC, IEEE Computer Society Press, pp. 148.

5. Capers Jones (2007), “Estimating Software Costs: Bringing Realism to Estimating”, Second Edition, Tata McGraw Hill, New York, pp. 3-629.

6. Capers Jones (2008), “Applied Software Measurement-Global Analysis of Productivity and Quality”, Third Edition, Tata McGraw Hill, New York, pp. 71-182.

7. Cigdem Gencel, Rogardt Heldal and Kenneth Lind (2009), “On the Relationship between Different Size Measures in the Software Life Cycle”, 2009 16th Asia-Pacific Software Engineering Conference, IEEE, pp. 19-26.

8. Capers Jones (2010), “Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies”, Tata McGraw Hill Edition,
pp. 1- 643.

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

10. Daniel V. Ferens (1999), “Software Size Estimation Techniques”, Air Force Institute of Technology (AFIT/ISY), Wriqht-Patterson AEB, Ohio 45433, pp. 701-706.

11. Daniel V. Ferens (1999), “The Conundrum Software Estimation Models”, Air Force Research Laboratory (AFRLIIFSD), IEEE, pp. 23-29.

12. Demirors and Gencel C. (2004), "A Comparison of Size Estimation Techniques Applied Early in the Life Cycle", Software Process Improvement, vol. 3281, pp.184-194.

13. Edilson J. D. Cândido and Rosely Sanches (2003), “Estimating the Size of Web Applications by using a Simplified Function Point Method”, IEEE.

14. Eck D., Brundick B., Fettig T., Dechoretz J. and Ugljesa J. (2009), "Parametric Estimating Handbook", The International Society of Parametric Analysis, Fourth Edition.

15. Erika Corona, Michele Marchesi, Giulio Barabino, Daniele Grechi and Laura Piccinno (2012), "Size Estimation of Web Applications through Web CMF Object", IEEE, pp. 14-20.

16. Ferchichi A., Bourey J.P., Bigand M. and Barron M. (2006),"Design Systems Engineering of Software Products: Implementation of a Software Estimation Model" IMACS Multi conference on Computational Engineering in Systems Applications(CESA), Beijing, China, pp. 1181-1188.

17. Filip Misovski (2007), "Estimating Development Of New User Interface", US 2007/0265779 A1, United States Patent.

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

19. Gustavo Bestetti Ibarra and Patrícia Vilain (2011), "Software Estimation Based on Use Case Size*"*, Brazilian Symposium on Software Engineering, IEEE, pp. 178 -187.

20. Gopalaswamy Ramesh (2013), "Managing Global Software Projects", McGraw Hill Education, India.

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

22. Humphrey Watts. S (2004), "Managing the Software Process", SEI Series in Software Engineering, Pearson Education, Singapore.

23. Henry Joel (2008), “Software Project Management A Real-World Guide to Success”, Pearson, India.

24. ISO/IEC 20968(2002), “Software Engineering – Mk II Function Point Analysis – Counting Practices Manual”, International Organization for Standardization – ISO, Geneva.

25. ISO/IEC19761 (2003), “Software Engineering – COSMIC-FFP A Functional Size Measurement Method”, International Organization for Standardization - ISO, Geneva.

26. ISO/IEC 20926(2003), “Software Engineering – IFPUG 4.1 Unadjusted Functional Size Measurement Method – Counting Practices Manual”, International Organization for Standardization – ISO, Geneva.

27. 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.

28. Iman Attarzadeh 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.

29. June Verner and Graham Tate (1992), “A Software Size Model”, IEEE Transactions on Software Engineering, VOL. 18, NO. 4, APRIL 1, pp. 265-278.

30. Juan J. Cuadrado Gallego, Pablo Rodríguez Soriaand Saahil Hakimuddin (2010), “Early Functional Size Estimation with IFPUG Unit Modified”, 9th IEEE/ACIS International Conference on Computer and Information Science, IEEE.

31. Karner G. (1993), "Resource Estimation for Objectory Projects," Objective Systems.

32. Kotonya G. and Sommerville I. (1998),”Requirements Engineering: Processes and Technique”, Chichester; New York: John Wiley.

33. Kjetil Molokken and Magne Jorgensen (2003), “A Review of Surveys on Software Effort Estimation”, International Symposium on Empirical Software Engineering (ISESE’03), IEEE.

34. Kenneth Lind and Rogardt Heldal (2009), “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.

35. Linda M. Laird (2006), “The Limitations of Estimation”, IT Pro November| December2006 Published by the IEEE Computer Society IEEE, pp. 40-45.

36. Lynch J. (2009), “Chaos Manifesto”, The Standish Group, Boston, [Online]. Available:http://www.standishgroup.com/newsroom/chaos_2009.php.

37. Lavanya Srinivasan and Steven S. Stefan (2010), "Enhanced Function Point Analysis", US 7,743,369 B1, United States Patent.

38. Luigi Buglione and Christ of Ebert (2011), "Estimation Tools and techniques", IEEE, pp. 92-96.

39. Mehwish Nasir and Farooq Ahmad H. (2006), "An Empirical Study to Investigate Software Estimation Trend in Organizations Targeting CMMI", 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.

40. Md.Forhad Rabbi, Shailendra Natraj and Olorisade Babatunde Kazeem (2009), "Evaluation of Convertibility Issues between IFPUG and COSMIC Function Points", Fourth International Conference on Software Engineering Advances, IEEE, pp. 277-281.

41. Mahir Kaya and Onur Demirörs (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.

42. Najberg and Andrew C. (1984), "Software Data Base Development", Volume I, Reading, MA, The Analytic Sciences Corporation, (Technical Report 4612-S-1), pp. 2-4.

43. Noureldin A.Z Adem and Zarinah M. Kasirun (2010),"Automating Function Points Analysis Based on Functional and non-functional Requirements Text", IEEE, pp. 664-669.

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

45. Paul A. Below and Poulsbo (2007), "System and Method for Function Point Sampling for Software Size Estimates", US 7,213,234 B1, United States Patent.

46. Richard D. Stutzke (2005), "Estimating Software-Intensive Systems: Projects, Products and Processes", SEI Series in Software Engineering, Addison Wesley, pp. 1-786.

47. Renjeev V. Kolanchery and Harish Ranganath (2007), "Project Size Estimation Tool", US 2007/0276712 A1, United States Patent.

48. Robert T. Futrell, Donald F. Shafer and Linda I. Shafer (2008), "Quality Software Project Management", Pearson Education, pp. 1-600.

49. Rodrigo C. Barros, Duncan D. Ruiz, Nelson N. Tenório Jr. and Márcio P. Basgalupp (2009), "Issues on Estimating Software Metrics in a Large Software Operation", 32nd Annual IEEE Software Engineering Workshop, IEEE, pp. 152-159.

50. ShukorSanim M. Fauzi, Hairul Nizam M. Nasir, Nuraminah R., Kamaruzaman Jusoff N.,Azylia A. Azam, Hafiz Ismail M. (2009), “The Implementation of Software Process Improvement Models”, International Review on Computers and Software, Vol. 4. No. 3, pp. 402 – 407.

51. Steven Fraser, Barry Boehm, Hakan Erdogmus, Magne Jorgensen, Stan Rifkin and Mike Ross (2009),”The Role of Judgment in Software Estimation”, ICSE’09, Vancouver, Canada, IEEE, pp. 13-17.

52. Ursula Passing and Martin Shepperd (2003), “An Experiment on Software Project Size and Effort Estimation”, International Symposium on Empirical Software Engineering (ISESE’03), IEEE.

53. Zia Z., Rashid A. and uz Zaman K. (2010), “Software Cost Estimation for Component Based Fourth-Generation-Language”, IET Software, pp. 103-110.

INDEX

ABOUT AUTHOR

John T Mesia Dhas received his Ph.D. in Computer Science and Engineering from Vel Tech University, Chennai. He has 15 years of Experience in the field of Education and Industry, currently he is working as an Associate Professor in Computer Science and Engineering Department of Audisankara College of Engineering and Technology, Gudur, Andhra Pradesh under Jawaharlal Nehru Technological University Anantapuramu.

He is also doing researches in Software Engineering and Data Analytics fields. He has published more than 14 research papers in conferences and Journals. He is one of the Rashtrapathi Scout Award winners of Bharath Scouts and Guides.

Title: Modern Metrics (MM): The Functional Size Estimator for Modern Software

Author: Dr. John T Mesia Dhas M.E., M.B.A., Ph.D.

Publisher: Self

ISBN: 978-93-5408-510-9

Price: ₹. 450/-

Address: No-1, MGR Street, Charles Nagar, Pattabiram
Chennai – 600072
India
Email: jtmdhasres@gmail.com