SOFTWARE SIZING APPROACHES

Dr. T. S. Shiny Angel

Dr. John T Mesia Dhas

The Palm Series

SOFTWARE SIZING APPROACHES

Dr. T. S. Shiny Angel

Dr. John T Mesia Dhas

Title: SOFTWARE SIZING APPROACHES

Author: Dr. T. S. Shiny Angel, 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

Mogappair West

Chennai -600037

India

ISBN: 978-93-5437-820-1

CONTENTS

CHAPTER 1

SOFTWARE SIZING PROCESS

Software sizing is a major process in software engineering that is used to measure the quantity of a software application or component. It helps to implement other software project management activities. Software project management is the method of planning, implementing, monitoring, controlling, leading, and managing software projects. For the successful completion of software development process, perfect planning is necessary. During planning, the Software Size is assessed, the effort is estimated in person hour or person months, cost and budget calculated, schedule prepared, resources and works allocation process to be completed [1]. The software size is important for perfect planning of development process because Size is the basic factor in determining the effort, duration, schedule, cost and other factors that affect the development process [2]. The software industry uses various sizing techniques to quantify the software size. They are Lines of code, Function points, Feature points, Use case points, Object points, Internet points, etc. These sizing techniques are programming language dependent or programming methodology dependent. The sizing technique that is used for assessing the size of the software system to be developed does not effectively determine the size that leads towards failure.

1.1 Project Planning

The success factors for Software are faster, cheaper, and better. We can achieve success in software through good planning and management [4,24]. The systematic software development process follows analysis, design, coding, testing and maintenance phases in sequential, concurrent or divide and conquers fashion. In the analysis stage, first capture all the requirements, then construct the initial business model, and lastly finalize the plan to develop the project. The process to plan a project starts with an assessment of the constraints that affects the project [5,37]. 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 series of steps followed for project planning [3,40].

i. Establish the project constraints.
ii. Make initial assessments of the project parameters.
iii. Define project milestones and deliverables.
iv. While project has not been completed, do the following steps otherwise the loop is cancelled
    a. Draw up project schedule.
    b. Initiate activities according to the schedule
    c. Review project progress.
    d. Revise the estimation of project parameters
    e. Update the project schedule
    f. Renegotiate project constraints and deliverables.
    g. If (problem arise) then
        i. Initiate technical review and possible revision
    Endif
v. End Loop

The above-stated project planning algorithm states 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 [5].

1.2 Activities during software project planning

The major activities in a project planning stage are assessing or estimating project parameters, Resources capturing, Project scheduling [2,6]. The following figure 1.1 shows these activities in detail.

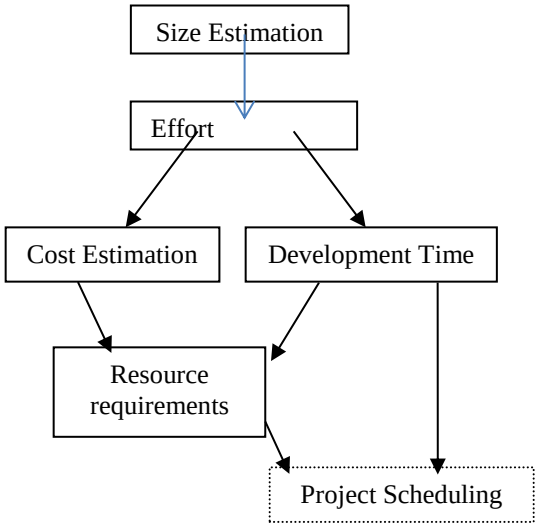

Fig. 1.1 Major activities in Project planning

Estimation is the process of expectation setting, which forms the basis of quantifying the resources to accomplish certain goals based on the clearly stated assumptions [1,64-65]. Size estimation is the predetermination of the size of final work product. Size is the basic measure to calculate other project factors.

1.2.2Specific quantities to Estimate and measure during the life cycle of project

Software Engineering Institutes capability maturity model- Integrated for systems/ software engineering (CMMI-SW/SE) recorded that the following quantities to be measured during the lifecycle of the project [56].

Effort (Activities)
Staff (Number, Skill and Experience, Turnover)
Time (Phase, Schedule, Milestones)
Costs (Labor and Non-Labor)
Computer resources used for development and test
Performance (Capacity, Accuracy, Speed, Response time)
Quality (Conformance to requirements, Dependability, Security, Data Integrity)

Price and total ownership cost.
Size or Amount (Created, Modified, Purchased)

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

1.3 About sizing Approaches

Software size is a key factor in determining the amount 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 [6]. The basic element for estimating is the size, and the e-learning systems have no exceptions. 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 [1-5,32-36]. These estimation techniques are broadly classified into three categories. They are

I. Expert Based Techniques
II. Code Based Techniques
III. Functional Size Measurement Techniques

In Expert based techniques, Expert or a group of experts uses their experience to understand the proposed project and they make estimation [2,41]. Expert judgment, estimation by analogy, Delphi techniques and Pattern matching and function points are the expert based techniques. Code based techniques measure the size or complexity of software using the count of source code. Lines of code, Halstead's program length are the code based sizing techniques. Functional Size Measurement techniques measure the size of the software based on functionalities specified in software and take account of their complexity. Functi1on points, Feature points, Use case points, Object points, Internet points, 3D function points, Backfiring function points, COSMIC function points, DeMarco "bang" function points, Full function points, Function point light, IFPUG function points, Mark II function points, Micro function points, Netherlands function points (NESMA), story points, Web object points, ELSE, LOP are the sample techniques under functional size measurement [1-5].

CHAPTER 2

IMPORTANCE OF SOFTWARE SIZE IN SOFTWARE PROJECT MANAGEMENT

This chapter focuses on the importance of Software Size and sizing in software project management and planning activities such as Cost, Effort duration estimation. The project plan is defined as the method of work to be done. This project plan gives a definition for each major task, an estimate of the time, cost, resources required, and the framework for management review and control [7]. The project plan is the powerful learning vehicle. If it is documented properly, then it will be a benchmark to compare with actual performance.

Watts S. Humphrey stated that the project plan is developed at the beginning of the job and is successfully refined as the work progress. The elements of a software project plan are Goals and objectives, work breakdown structure (WBS), product size estimates, resource estimates and project schedule. Goals and objectives describe what is to be done, for whom, and by when, as well as the criteria for determining the success of the project. The WBS subdivides the project into tasks that can be defined, estimated, and tracked. Product size estimates are the quantitative assessment of the code requires for each product element. Resource estimates yield the reasonable resources requires for each WBS element. Based on the product size, resources and available project staffing, the schedule for the key tasks and the deliverable item is produced. This plan provides management with the basis for periodically reviewing and tracking progress against the plan. A contingency in the plan that is specifically the software size inflates the resource estimates, add to the schedule and increase costs [6,63].

Capers Jones stated that project management is the weak link in the software engineering chain. More software problems such as cost and schedule overruns can be attributed to poor project management. Poor project management leads to failures [1, 23-25]. The term failure refers to the projects that are cancelled without completing due to reasons like cost or schedule overruns. He also tabulated the project management activities and software management performance on successful and unsuccessful projects [3]. Following table 2.1 shows the same.

Table 2.1 Software Management Performance on Successful and unsuccessful projects

Sl.No	**Project Management Activities**	**Successful projects**	**Unsuccessful projects**
1	Sizing	Good	Poor
2	Planning	Very Good	Fair
3	Estimating	Very Good	Very poor
4	Tracking	Good	Poor
5	Measurement	Good	Very poor
6	Quality control	Excellent	Poor
7	Change Control	Excellent	Poor
8	Problem resolutions	Good	Poor
9	Risk analysis	Good	Very poor

Sl.No	Project Management Activities	Successful projects	Unsuccessful projects
10	Personal management	Good	Poor
11	Supplier management	Good	Poor
12	Overall performance	Very Good	Poor

The United States patent holders Paul A. Below and Poulsbo of Electronic Data Systems Corporation, Plano, TX (US) invented a new sizing approach in 2007 by modifying FPA. He stated that the required investment seems overwhelming to the business managers, who are sometimes unaware of the information that the portfolio evaluations are critical to their analytical decision process. A second problem occurs to the organizations that need to develop estimates early in a project life cycle. Normally, a count or function point estimate provides the functional size measure in order to feed the estimating process. For this estimation the time frame needed is short, and so the decisions need to be made quickly regarding alternatives and scheduling. But for large projects, FP counting an entire system can take days, weeks, or in some extreme cases, it takes months. The organization often fails upon expert opinion-based estimates, as there will be no way to check the reasoning ability of the result. In some instances, the non-optimum plans are made, with disastrous results. Size is one of the most essential of all the metrics needed for fair analysis of software engineering processes, quality evaluation, and related productivity [11,20,59].

Lavanya Srinivasan and Steven S. Stefan of United States Patent holders mentioned that it is desirable to complete the software development projects on time and within the allocated budget. Also, estimating the level of effort (LOE), or man-hours, that is needed to complete a project helps in project scheduling and project budgeting. Without an accurate LOE estimate, the project may not be completed within the required time. For example, too many or too few developers in a project deliver it late and over-budget. So, an enterprise can track the budget and schedule the project by comparing an estimated LOE generated before commencing a project with an actual LOE that is measured during or after a project. There are various methods of estimating the LOE that is needed to complete a project. It has been employed with varying degrees of success. One of the methods is function point analysis. It involves in identifying and counting functionalities or function points that are to be delivered by a project. The function points can be assigned by difficulty ratings, which reflects the amount of effort needed to achieve the specified functionality. Naturally, when more function points are delivered, then the complexity of the function points will also be greater. Then it needs more LOE to complete the project [62].

Mehwish Nasir, H. Farooq Ahmad [6] suggested that formal methods for estimating size, effort and cost for the project should be implemented apart from heuristics used for estimation. Different type of estimation methodologies is applicable in different categories of projects. None of the methodologies gives 100% accuracy, but if there is proper use, then the estimation process will be smoother. In any software application, accurate software estimation is desirable. The Software applications have to schedule the budget, resources, time, and cost. This helps to avoid overrunning but the software organizations have to estimate and plan better to get the projects in bidding. Pre-bid estimation is a dominant feature in getting business for the company. The accuracy of the pre-bid estimation governs the smooth running and success of a project. The Practice followed for software estimation is

- Software Size computation
- Effort estimation in person-hour, which is derived from software size.
- Cost & budget calculation.
- The proper scheduling and resource allocation is done as a final step.

Mahir Kaya*et al.,* [8] states that Software size is the key input for cost and effort estimation models. The effort and cost estimations, as well as software size is needed as early as possible for the project.

Daniel V. Ferens*et al.*,[10] discusses the issue of software size estimation, a key consideration in estimating software costs. Although software sizing is relatively a new area of interest, numerous techniques have been developed to predict software size. However, the usefulness of various techniques depends on the unique situation of the estimator. Furthermore, some techniques are of questionable validity. Nevertheless, the area of software size estimation will continue to receive emphasis and new techniques will emerge

Renowned cost analyst Dr. Barry Boehm [30-31] states, "The biggest difficulty in using today's algorithmic software cost models is the problem of providing sound size estimates."

Z. Zia*et al.*,[12] states that Software cost estimation is important for budgeting, project planning, risk analysis, and software improvement analysis. Numerous estimation techniques are there in Software cost estimation. For the past three decades, there had been some significant developments in effort estimation, cost estimation methodology and size of software. Current software cost estimation models have been experiencing difficulties in estimating the costs of software, as new software development methodologies and technologies are emerging very rapidly. Most of the software cost models rely on such inputs as the estimate of lines of source code, delivered sets of instructions, function points and processing complexity or experience levels to produce the cost estimates. These models generally produce inaccurate results to estimate the cost of software development. In current development environments, they use component-based software development environments like Visual languages.

Md. Forhad Rabbi *et al.*,[13] finds that the Software industry has grown with time, from small application of few lines of codes to software application of millions of lines of code. Nowadays, the developed software is grandly scalable and more robust with high usability features. This results in the immense size and complexity of the software. Unfortunately, the early estimation used for these projects are often proved incorrect in the later part of the projects lifecycle. In recent years, the major concern of many large companies has been the project estimation. It is because the software cost estimates are critical to both developers and customers in terms of cost and effort. Also, there is no panacea technique available in the industry that guarantees 100percentages. Nevertheless, there are only a few project estimation techniques available in the industry that helps in making the estimation quite optimal and worthy[14-17]. Function points have been a means to determine the software size since the early 70's [32]. The IFPUG is the most traditional and widely used method for calculating the function size of a software project. However, different function point analysis (FPA) techniques have evolved over the years. All these techniques aim to correct the lapses of the IFPUG. Only four of them have matured to ISO measurement standard level - ISO 19761: COSMIC FFP [33], ISO 20926: Function point analysis, e.g., IFPUG 4.1 [36], ISO 20968: Mk II [35] and ISO 24570: NESMA [34]. Most of these methods measure specific application type.

Edilson J. D. Candido*et al.*,[18] presents that Software size estimation is a key factor in determining the amount of time and effort that is needed to develop the software systems.

Linda M. Laird [19] informs that in software, we primarily want to estimate three aspects of a project: effort, schedule, and cost. Most of the methods are started by estimating size as thousands of lines of code (KLOC), function points or some other proxy point and uses that to estimate effort. From effort, you typically derive staffing, schedule, and cost.
Steven Fraser *et al.*,[21] judges that "The consequences of poor or good judgment in estimation are reflected in software quality, cost, time-to-market, and operational reliability."
Daniel V. Ferens [22] states that there are many sophisticated models and methods for estimating the size, cost, and schedule of software projects. However, the ability to estimate the software cost, size, or schedule is still dubious.
Iman Attar zadeh*et al.*, [25] Proposed a New-High Performance Model for Software Cost Estimation because Effort, time and cost estimate at the early stages of the software development are the most difficult to obtain, and they are often the least accurate.
The popular Effort and cost estimation models are COCOMO [37], SLIM [38], Function Point, Use Case Points [39] and SEER-SEM [40].The main cost driver of these models is the size of the software. In COCOMO and SLIM models, the size is measured in Source Lines of Code (SLOC). However, the function point and the use case point models take software size in function points (FP) and use case points (UCP) respectively. Expert Judgment, Estimation by analogy are the other human-based models. Expert judgment involves consulting a group of experts to use their experiences to propose an estimation of a given project [41]. The Delphi technique is used to provide communication and cooperation among the experts during estimation [42].These models also use the size as the base factor. The following table 2.2 describes other Cost and Effort Estimation Models and the sizing approaches used by them for cost and Effort estimation.

Table 2.2 Significance of sizing approaches in Effort and Cost Estimation Models

Sl. No	Cost/Effort Estimation Models	Used Sizing Technique
1	COCOMO(Constructive Cost Model)	LOC
2	Knowledge Plan	LOC or IFPUG's FPA
3	PRICE-S	LOC, IFPUG's FPA
4	SEER-SEM	LOC or IFPUG's FPA
5	SLIM	LOC
6	Basili -1996	LOC
7	Niessink and Van Vliet- 1998	IFPUG's FPA,
8	Abran -1995	Extended IFPUG's FPA
9	Abran -2002	COSMIC-FFP
10	Caivano -2001	LOC
11	Sneed- 2004	LOC, IFPUG's FPA, Object-points
12	Jorgensen- 1995	LOC
13	De Lucia -2005	LOC

It also shows that size is the basic input to estimate cost, effort and other parameters of software. It implies the importance of sizing.

CHAPTER 3

SIZING APPROACHES

Sizing approach denotes a method or technique, which is used to quantify the size of the software. There are numerous sizing techniques and is mentioned in chapter 2.1. These Sizing Approaches are broadly classified into three categories. They are

- Expert based Techniques
- Code based techniques
- Functional Size Measurement techniques

The sizing approaches under the broad categories are represented in figure 2.1.The following section describes the popular sizing approaches in the Software Industry and their limitations in Sizing E-Learning system.

3.1 Lines of Code

The computer era starts from the mid of 20th century. The Lines of Code 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 [22].

Advantages

i. It is widely used and universally accepted.
ii. It measures software from the developer point of view.

Limitations of LOC in the sense of E-Learning System

i. E-Learning system is a web application, which holds a huge volume of the document.
ii. The one line of the document is not equal to the one line of executable instruction.
iii. The worth of one executable line that may explore any object like picture, animation, simulation, video or audio kind file is multiple times greater than LOC.
iv. Mapping the document part of E-Learning system is difficult to the sizing concept behind LOC.

3.2 Halstead's Software Science

The Software Science developed by M.H. Halstead attempts to estimate the programming effort [2,40]. 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.

$$n = n1 + n2 \qquad (3.1)$$

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.

$$N = N1 + N2 \qquad (3.2)$$

Advantages of Halstead's Software Science

i. Requires no in-depth analysis of programming structure.
ii. Predicts the rate of error.
iii. Predicts the maintenance effort.
iv. Useful in scheduling and reporting projects.
v. Measures the overall quality of programs.
vi. Easy to calculate.
vii. Used for any programming language.
viii. Various industry studies support the use of Halstead to predict programming effort and mean a number of programming bugs.

Limitations of Halstead Software Science in the sense of E-Learning system

i. It depends on completed code.
ii. Used only for scientific languages like FORTRAN.
iii. E-Learning System has a huge volume of learning content with different media elements like video, audio, animation and simulation. Sizing these elements is not specified in Halstead Software Science.

3.3 Expert Judgment

Expert or group of experts uses their experience to understand the proposed project, and they make estimation. 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[21].
Steps:
1. Coordinator gives each expert a specification and an estimation form.
2. Coordinator calls a group meeting in which the experts discuss estimation issues 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.

Advantages

i. The new projects that are similar to the old projects are accessed and give good estimates.
ii. Experts are good in the problem domain then we can receive accurate estimates from them.

iii. Good for level 3 and level 4 organizations.

Limitations of Expert judgment in the sense of E-Learning system

i. This method cannot be quantified.
ii. Human errors are possible
iii. Consumes time and cost.
iv. Spend more money for experts
v. Estimating each part of E-Learning system requires different experts such as simulation, animation, etc. that is more expensive.
vi. Knowledge in past projects, experience in project management and judgment are required for experts otherwise, it gives disappointment.

3.4 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 areNarrow 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.

Advantages

i. Implementation is easy and inexpensive in the case of simple systems.
ii. It takes the expertise of several people.
iii. All the participants will get a better idea about the software.
iv. It requires no historical data.
v. It is used for detailed and high-level estimation.
vi. Results are more accurate and less dangerous than LOC

Limitations of Delphi technique in the sense of E-Learning system

i. Estimating each part of E-Learning system requires different experts such as simulation, animation, etc. that is more expensive.
ii. Single expert may not have the expected skill set.
iii. Experts may not provide 100% confidence regarding sizing.

3.5 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's not supported for E-Learning system sizing because of complex parameters.

Bhoem, describes this estimation as “analogy estimation.” This is based on the estimation technique described by Ray Wolverton. This estimation tries to identify the similarities and differences between projects. Quantitative analogy techniques are the simple way to adjust the historical values. Experts quantify how much the difference the value of new object is related to the value of known object. To estimate the differences, experts can use the Delphi technique, experience or some other technique. The estimated differences are applied additively or multiplicatively. These two analogy techniques are an improvement over the basic Delphi technique.

3.5.1 Additive Analogy

In additive analogy, the estimator adds or subtract a small amount from a known(historical) value to calculate the estimated value for the new object. Randall Jensen describes a similar technique called “relative comparison.”

3.4.5.2 Multiplicative Analogy

In multiple analogies, the ratio scaling is applied to a quantity for estimation. It determines the percentages of each factor using PERT or Delphi technique.

3.5.2 Algorithmic analogy

Algorithmic analogy estimation produces quantitative results in a repeatable way. It uses the data from multiple objects that have similar characteristics. A database maintains the software development effort for some completed projects and size of the software. This technique has two main activities

i. Define the method
   - Define a data set
   - Define measure for each characteristic and the quantity
   - Collect data set for multiple existing instances
   - Validate the data
   - Define an algorithm to measure the degree of similarity between instances
   - Define an algorithm to select the nearest neighbours for the new instance.
   - Define an algorithm in order to calculate the value of the new instance.

ii. Apply the method
   - Specify the characteristics of the new object.
   - Locate and select the desired number of nearest neighbours.
   - Combine the values from the neighbours to get the value to be estimated.

Adrian Cowderoy and Joh Jenkins described one of the first structured approaches to use the analogy for estimation. For software or system estimation, the analogies are applied to the objects in the same problem domain. This is more properly called as the case-based reasoning [2].

3.5.3 Pattern matching and function points

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 [1,4]. 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 the Function point approach has to be considered.

Advantages

i. Implementation is easy and inexpensive if there is any historical data of similar projects.
ii. It is used for high level and detailed estimation.
iii. Results are more accurate and less dangerous

Limitations of Analogy method in the sense of E-Learning system

i. Historical data is highly essential
ii. In E-Learning system, Historical data are limited because E-Learning system development field is currently in a growth.

3.6 Function Point Analysis (FPA)

Function Point Analysis (FPA) is the standard metrics for measuring functional size of a software system. The function point was first defined by A.J. Albrecht at IBM in late 1970's. The Function Point Analysis (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 Line 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 [10].
There are two types of functionality in FPA: The first one is data functions to count 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 (EIFs), Internal Logical Files (ILFs), External Inputs (EI's), External Outputs (EO's) 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. [12]

Transaction Functions

There are three types of transaction functions. They are External Input (EI), External Output (EO) and External Inquiry (EQ).

External Input (EI): External Inputs 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 Inquires (EI): Inquires 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 Point for the five categories of data and transaction functions.

Table3.1 Unadjusted Function Point Calculation

Function Type	**Weight by Functional Complexity**		**Total (FP)**
External Input (EI's)	Low	__x 3	
	Average	__x 4	
	High	__x 6	
External Output (EO's)	Low	__x 4	
	Average	__x 5	
	High	__x 7	
External Enquiry (EI's)	Low	__x 3	
	Average	__x 4	
	High	__x 6	
Internal Logical File (ILF)	Low	__x 7	
	Average	__x 10	
	High	__x 15	
External Interface File (EIF)	Low	__x 5	
	Average	__x 7	
	High	__x 10	
Total Number of Unadjusted Function Points:			

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

General System Characteristic		Brief Description
1.	Data communications	There are communication facilities to aid in transferring or exchanging the information with the application or system.
2.	Distributed data processing	Handling the distributed data and processing functions.
3.	Performance	The response time or output required by the user.
4.	Heavily used configuration	The heavy use of the current hardware platform where the application is executed.
5.	Transaction rate	The transactions that are executed daily, weekly, monthly, etc.
6.	On-Line data entry	The On-line percentage of the information is entered.
7.	End-user efficiency	The end-user's efficiency to design the application.
8.	On-Line update	Updating the ILF's through On-Line Transaction.
9.	Complex processing	The application provides extensive logical or mathematical processing.
10.	Reusability	The application is developed to meet one or many user's needs.
11.	Installation ease	The difficulties of the conversion and installation.
12.	Operational ease	The effective and automated are a start-up, back-up, and recovery procedures.
13.	Multiple sites	The applications are specifically designed, developed, and supported to install at multiple sites for multiple organizations.
14.	Facilitate change	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)

$$CAF = 0.65 + (0.01\, x \sum_{i=0}^{n} C\, i) \quad (3.3)$$

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

$$FP = UFP\ x\, CAF \quad (3.4)$$

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 E-Learning system

i. FPA focuses on the computation part of an application. In 1980's, an application system has a full computational part. So, it is focused on external inputs, outputs, inquires, internal logical files and External interface files. But E-Learning 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. The count of each component is high then it states the complexity is high. But it not considered the worthy of each component.
iii. It is not well suited to non-MIS applications especially E-Learning system like web applications.

3.7 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. In 1986, software productivity research developed feature point analysis [5]. FPA never consider the complexity of algorithms involves in each application. To overcome that problem feature point method was introduced. The complexity of algorithms defined in terms of the number of rules required to express that algorithm. The formula for calculating Feature point FuP is given in equation 3.5.

$$FuP = Raw\ Feature\ Point * CAF \quad (3.5)$$

Where FuP denotes Feature Point, and CAF denotes Complexity Adjustment Factor.

Determination of Raw Feature point

Count inputs, outputs, files, inquiries, algorithms and interfaces require a system and multiply with an average weighting factor. All of these values are then scored, and the total is expressed in raw feature point. The following table 3.3 assists for calculating raw feature point.

Table 3.3 Calculating raw feature point.

Feature Type	**Average**	**Total**
No. of Inputs	___ * 4 =	_____
No. of Outputs	___ * 5 =	_____
No. of Files	___ * 7 =	_____
No. of Inquiries	___ * 4 =	_____
No. of Interfaces	___ * 7 =	_____
Count the number of Algorithms	___ * 3 =	_____
Total Raw Feature Point		_____

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 following table 3.4assiststo 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 Complexity adjustment factor (CAF) is assigned. The following 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

Multiply the raw feature point with CAF provides the exact feature point of the system. Use Equation 3.5.

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. But naturally, it is good for the embedded system and the real time system sizing.

Limitations of Feature points in the sense of E-Learning System

i. It considers the simple entities and algorithms used by the application system. But, it never considers the complex entities like video, audio, simulation, animation and their worth fullness.
ii. It never considers the database and networking support that is needed for the application.
iii. It never considers other technical factors that influence the execution of e-learning system.

3.8 Use Case Points

Using the case point was introduced in the year of 1993 by Gustav Karner of Objectory. It is an extension of FPA [2]. It supports sizing in the early stage itself. The following equation 3.6 is used for calculating Use case points.

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

Where UCP = Use Case Points.
UUCP = Unadjusted Use Case Points.
TCF = Technical Complexity Factor
EF = Environmental Factor

Determination of Unadjusted Use Case points

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 theweight 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. 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 unadjusted use case points (UUCP).

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

Where,

UUCP - Unadjusted use case points
UAW –Unadjusted Actor Weight
UUCW – Unadjusted Use case Weight

Determination of technical complexity factor

The technical complexity of the product can be calculated based on the degree of influence of 13 technical factors. The Following 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.

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

Determination of environmental factor

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

Table 3.7 shows 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 factor EF.

$$EF = 1.4 - 0.03 * Esum \qquad (3.9)$$

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 E-Learning System

i. It counts the number of actors and use cases involved in an application system and identifies the complexity. But it never identifies the implementation level difficulties.
ii. Use case provides the initial view of the business model. But it is not much detail and using this we can't provide exact estimates.
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 E-Learning system is not mentioned.
v. Simulation, animation, video and audio specifications and their complexities are not assessed.

3.9 Object points

Object point was introduced by Banker in 1991. It was objected 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 [2].

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 objects point (OP). Multiply OP by a reuse factor (RF). Reuse is expressed in percentage, 10% corresponds to the value of 0.1 and find the new object point (NOP) using the following equation 3.10.

$$NOP = OP * (1 - RF) \quad (3.10)$$

Where,

NOP - New Object Point
OP - objects point
RF - reuse factor

Advantages

i. Good for GUI based applications.
ii. It highly considers for reusability.

Limitations of object point in the sense of E-Learning system

i. E-Learning 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. E-Learning 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 the only reusability out of all technical and environmental factors that influence the system.

3.10 Other Sizing Approaches

The above-stated sizing techniques are popular in the software industry. But the following sizing techniques are used by some companies based on the need though they are not much popular [1-5].

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. E-Learning system has multiple algorithms, produce multiple reports. The database and different media files are also involved. So, this sizing technique is not suited for E-Learning 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, COTS components, graphic files, multimedia files and scripts. It counts all objects and as well as FPA web objects are also sized. It is good for assessing

the size of web site, but E-Learning system is highly more than a website. It is a document rich web application. Each of the learning documents should be generated using different pedagogies. So, it is not sufficient for sizing E-Learning 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, so there is no way to support in assessing the size of E-Learning 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. 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.

Model Blitz

This method suggests that based on the requirements; construct a model and this model supports for sizing. It is a simple and quick sizing technique, but sizing happens after the designing phase. For E-Learning system, the model itself takes more cost and time.

CHAPTER 4

ELSE: E-LEARNING SYSTEM ESTIMATOR

ELSE stands for E-Learning system estimator. It was an attempt to overcome difficulties associated with Function point analysis for estimating the size of E-Learning system. Function point analysis is one of the traditional methods used for estimating the size of software system [1] - [5]. It was introduced in mid-1970 by Allan J. Albrecht to overcome the drawbacks of calculating size using Line of Code (LOC). FPA is one of the functional size measurement mechanisms and one of the popular approaches in software industry also it is good for sizing applications. E-Learning system is application software; it can have Document for teaching part (DTP), Assessment part, database management, security setups, etc. The document for teaching part consists of learning document which includes text, images, links, animations, simulations, Video and audio data. The existing Function point analysis is good for estimating the size of the normal computational part of the E -Learning application. But they are not good for estimating the size of DTP. So ELSE is projected for E-Learning system estimation. The major contents of this chapter are published in the research articles named as "ELSE: E-Learning System Estimator" of International Review on Computers and Software in November 2012[53]. The following sections discuss about ELSE in detail.

4.1 About ELSE

ELSE follows FPA because FPA is the widely accepted estimation system for application software. FPA is used to calculate Function points but ELSE is used to calculate E-Learning System Size(ELSSIZE). An ELSSIZE is a unit of measurement to express the amount of business functionality an e-learning system provides to a user. The major components of ELSE are Unadjusted E-Learning points (UEP) and Value adjustment factors (VAF). The components of UEP are as follow.

- ELSI – E-Learning System Inputs
- ELSO – E-Learning System Outputs
- ELSQ – E-Learning System Queries
- ELSIM – E-Learning System Images
- ELSTD – E-Learning System Textual Documents
- ELSAS – E-Learning System Animations
- ELSV – E-Learning System Video
- ELSAU – E-Learning System Audio
- ELSS – E-Learning System Simulation
- ELSDB – E-Learning System Database Size
- ELSF – E-Learning System Functions Referred.

ELSI is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. ELSO is an elementary process in which derived data passes across the boundary from inside to outside. ELSQ is an elementary process with both input and output components that result in data

retrieval from one or more internal logical files and external interface files. ELSIM is Number of images present in our system. ELSTD is Number of pages of document present in the system. ELSAS is an Amount of time taken for animation. ELSV is an Amount of time taken for video present in the system. ELSAU is an Amount of time taken for audio present in the system. ELSS is an Amount of time taken for simulation present in the system. ELSDB is Number of records accepted in the system and ELSF is Number of internal or external files associated with the system.

4.2 Rating of Components

At first the needful components to be developed for E-Learning system are identified and should be classified under the above 11 set and count each set separately. For ELSAS, ELSV, ELSAU, ELSS fix the time it required to play. The following tables 4.1and 4.2 assist the rating process of each component. For example, E-Learning System Inputs (ELSI) count may between 0-20 then the complexity should be considered as low.

Table 4.1 Rating Processes of Components

Components (count)	**Low**	**Average**	**High**	**Very High**
ELSI	0 – 20	21 – 50	51 – 100	>100
ELSO	0 – 20	21 – 50	51 – 100	>100
ELSQ	0 -50	51 – 100	101 - 200	>200
ELSI	0 – 25	25 – 50	51 – 100	>100
ELSTD	0 – 100	101 – 200	201 - 400	>400
ELSF	0 – 25	26 – 50	51 – 100	>100
ELSDB	0 - 20000	20001 – 40000	40001 – 60000	>60000

Table 4.2rating Processes of Time-Based Components

Components (time)	**Low**	**Average**	**High**	**Very High**
ELSAS	0 – 10 hour	10 – 30 hour	30 – 75 hour	>75 hour
ELSV	0 – 50 hour	51 – 100 hour	101 – 200 hour	>200 hour
ELSAU	0 – 100 hour	51 – 100 hour	101 – 200 hour	>200 hour
ELSS	0 – 10 hour	10 – 30 hour	30 – 75 hour	>75 hour

After rating process fix the transaction value based on the following table 4.3. According to the complexity rating the transaction value will be selected. For example complexity rating of E-Learning System Inputs (ELSI) may low then the transaction value will be 3 .

Table 4.3 Values for Transaction

Components	Low	Average	High	Very High
ELSI	3	4	6	10
ELSO	4	5	7	10
ELSQ	3	4	6	9
ELSI	4	6	8	11
ELSTD	4	6	8	9
ELSF	4	6	8	10
ELSDB	3	4	6	9
ELSAS	7	10	15	20
ELSV	5	8	10	13
ELSAU	5	8	10	13
ELSS	7	10	15	20

The counts for each component can be entered into Table 4.4, ie shown below. It is supported for calculating unadjusted function points. Each count is multiplied by appropriate transaction value and the sum of each field in the row gives total E-Learning points of the specific component. These totals are then summed down to arrive at the Total Number of Unadjusted E-Learning Points.

Table 4.4 Calculating Unadjusted E-Learning Points

Type of Components	Ranking Levels				Total
	Low	Average	High	Very High	
ELSI	--*3=--	--*4=--	--*6=--	--*10=--	
ELSO	--*4=--	--*5=--	--*7=--	--*10=--	
ELSQ	--*3=--	--*4=--	--*6=--	--*9=--	
ELSI	--*4=--	--*6=--	--*8=--	--*11=--	
ELSTD	--*4=--	--*6=--	--*8=--	--*9=--	
ELSF	--*4=--	--*6=--	--*8=--	--*10=--	
ELSDB	--*3=--	--*4=--	--*6=--	--*9=--	
ELSAS	--*7=--	--*10=--	--*15=--	--*20=--	
ELSV	--*5=--	--*8=--	--*10=--	--*13=--	
ELSAU	--*5=--	--*8=--	--*10=--	--*13=--	

Type of Components	Ranking Levels				Total
	Low	Average	High	Very High	
ELSS	--*7=--	--*10=--	--*15=--	--*20=--	
Unadjusted E-Learning Points					

4.3 Value Adjustment Factor (VAF)

The value adjustment factor (VAF) is based on 15 general system characteristics (GSC's) that rate the general functionality of the application being counted. Each characteristic has associated descriptions that helps determine the degrees of influence of the characteristics. The degrees of influence range on a scale of zero to five, from no influence to strong influence. The 14 general system characteristics of function points are used in ELSE along with the coupling.

Table 4.5 Descriptions of Value Adjustment Factors

General System Characteristic	
1	Data communications
2	Distributed data processing
3	Performance
4	Heavily used configuration
5	Transaction rate
6	On-Line data entry
7	End-user efficiency
8	On-Line update
9	Complex processing
10	Reusability
11	Installation ease
12	Operational ease
13	Multiple sites
14	Facilitate change
15	Coupling

Once all the 15 GSC's have been answered, they should be tabulated and calculate the Value Adjustment Factor (VAF) using the Equation 4.1. The rate of factors varies from 0 to 5. The given factor not used in the system then rate is 0. Otherwise the rate of a given factor is represented based on its influence.

0 - No influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 – Essential

$$VAF = 0.65 + [\sum_1^{15} ci/100] \quad (4.1)$$

Where i is vary from 1 to 15 representing each GSC and Ciis the degree of influence of each General System Characteristic. It is summation value of all 15 GSC's.

4.4 ELSSIZE Calculation

The final E-Learning system size (ELSSIZE) is obtained by multiplying the VAF with Unadjusted E-Learning Points. The equation 4.2 is used for ELSSIZE calculation.

$$ELSSIZE = UEP * VAF \quad (4.2)$$

4.5 ELSE versus FPA

In the software project management view ELSE is too good for estimating the size of E-learning system. The following table 4.6 Shows the comparison between FPA and ELSE based on the features of Estimators and E-Learning system.

Table 4.6The comparison between FPA and ELSE

Features	FPA	ELSE
Technology	Parametric / Proxy based / Algorithmic method	Parametric / Proxy based / Algorithmic method
Past project experience	Necessary for producing effective parameters for estimation but not mandatory	Necessary for producing effective parameters for estimation but not mandatory
Time	Estimation time reduced	Estimation time reduced
Accuracy	Accurate according to the specifications of FPA. but for Web/GUI based applications it is not good	Accurate for E-Learning system and Web applications
Dependency	Language Dependent	Not Dependent
cost	Cost is less for estimation	Cost is less for estimation
Quality	Good for procedure oriented programming	High
Reusability	considered	considered
GUI support	Little bit Supported	Highly supported

Features	FPA	ELSE
Database	Little bit considered	Highly considered
Networking	Little bit considered	Highly considered
Storage	Little bit considered	Highly considered
Distribution	Little bit considered	Highly considered
Multimedia specialization	Not Considered	Highly considered
Effort of special effects	Not Considered	Considered
Consideration of animation	Not Considered	Considered
Simulation	Not Considered	Not Considered

4.6 Benefits of ELSE

i. ELSE can be used to size E-Learning applications accurately. Sizing is important component in determining productivity.
ii. It is easily understood by the non-technical user. This helps communicate sizing information to a user or customer.
iii. Conversion to LOC is similar to FP to LOC conversion.

It also supports to estimate any kind software application other than E-Learning system also.

CHAPTER 5

LEARNING OBJECT POINTS METHOD

Learning Object points method is a proposed sizing approach for sizing E-Learning system. LOP is a unit of measurement to express the amount of learning objects and operational functionalities an e-learning system provides to a user. This method was introduced to overcome the drawbacks of existing size estimation techniques [55-57]. In E-Learning system huge volume of input and output transactions are happen in the form of registration, Fund transfer, Submission of learning and assignments contents, etc. Numerous logical files involved for eligibility checking, grading calculation, grouping, ordering, assessing, etc. Interface files support for connecting external components like database to our application. E-Learning system may also a web application so number of Web pages associated. Screens and reports are associated with E-Learning system. Screens are act as a user interface and reports are the great output expected by the stakeholders of the system. It has huge volume of multimedia files, graphic files, databases and internet-based knowledge transfer happened. All these aspects of E-Learning system are considered in LOP method. It is the Modified version of ELSE proposed by T.S.Shiny Angel et al in 2012[53].The following sections discusses the architecture of LOP method and the steps to calculate the size of E-Learning system. The major contents of this chapter are published in the research articles named as "Estimating the Size of E-Learning System using Learning Object Points Method" of Indian Journal of Science and Technology in September 2016[58].

5.1 ARCHITECTURE OF LOP

LOP has three major components to cover all the features of the E-Learning system. The first component is Unadjusted learning point (ULP), which includes Number of inputs (NI), number of outputs (NO), Number of files (NF), number of interfaces (NI), number of web pages (NWP), number of screens and reports (NSR), Duration of multimedia files (DMF), number of graphic files (NGF) and Number of document pages (NDP). The second component is Technical complexity factor (TCF), It is Calculated based on the 14 system Characteristics which includes Data communications, Distributed data processing, Performance, Heavily used configuration, Transaction rate, On-Line data entry, End-user efficiency, On-Line update, Complex processing, Reusability, Installation ease, Operational ease, Multiple sites, Facilitate change and the third component is Learning complexity factor (LCF) which includes Familiar with E-Learning system development(FELSD), Analyst capability(AC), Motivation(M), Requirement stability(RS), Number of courses(NC) and Expected Students strength(ESS). The following Figure 5.1 Shows the Architecture of LOP Method.

The LOP of an E-Learning system is calculated by using equation 5.1

$$LOP = ULP * TCF * LCF \qquad \text{(Eqn -5.1)}$$

Where

LOP - Learning Object Points
ULP - Unadjusted Learning Points
TCF - Technical Complexity Factor
LCF - Learning Complexity Factor.

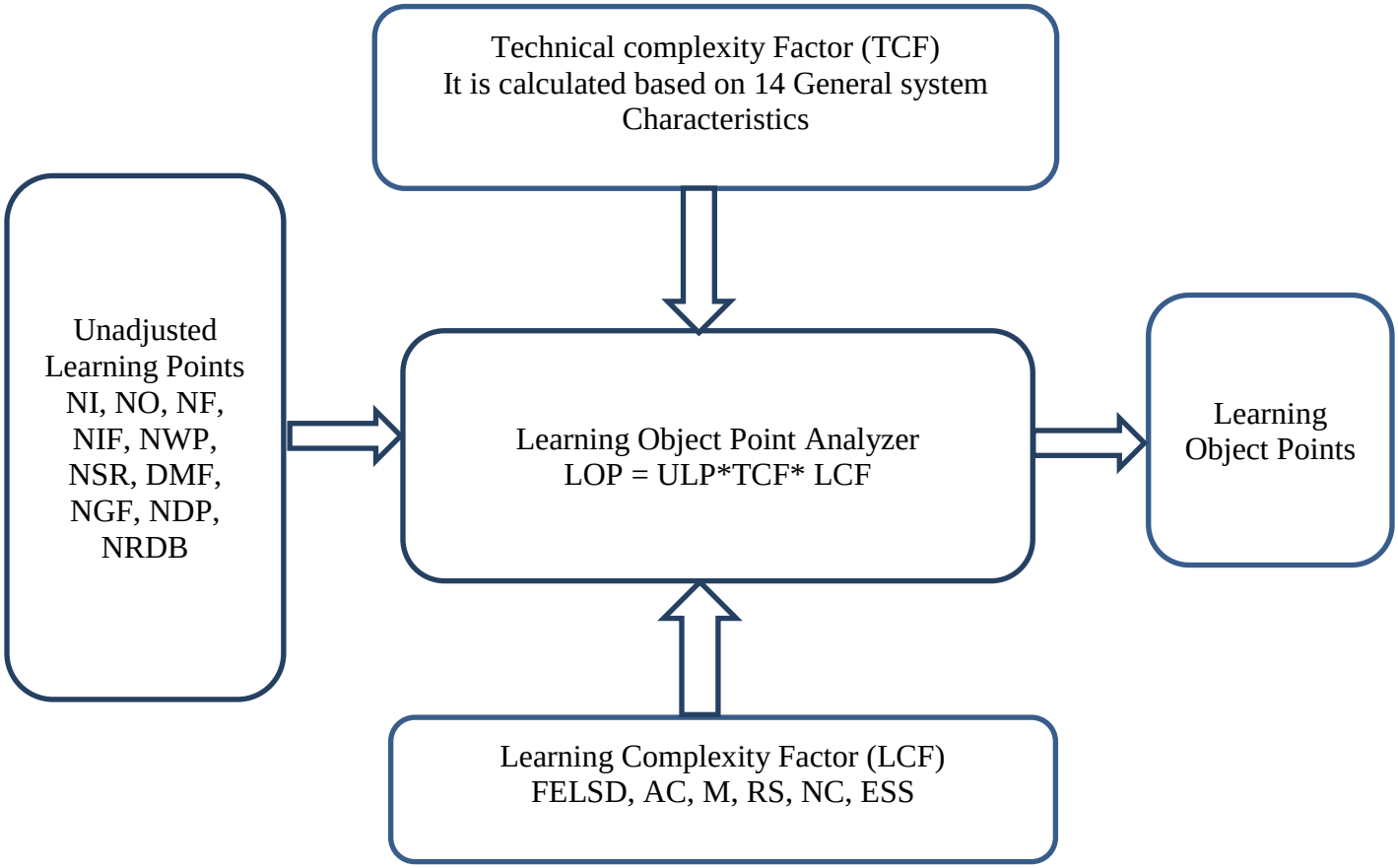

Figure 5.1 Architecture of LOP Method

5.2 ALGORITHM TO CALCULATE LOP

The following algorithm used to calculate LOP of a given software system.

i. Start the process
ii. Identify the counting Scope and application boundary. For small projects take the scope as a whole. For large projects, split them in to multiple parts, calculate LOP for each part and sum up together to receive a final count.
iii. Determine Unadjusted Learning Points.
iv. Determine Technical Complexity Factor
v. Determine Learning Complexity Factor
vi. Calculate LOP of a given system
vii. Stop the process

5.3 DETERMINATION OF UNADJUSTED LEARNING POINTS (ULP)

ULP can be calculated with the help of ten major components of E-Learning system. The components of ULP are as follow.

NI – *Number of Inputs*, Number of inputs accepted by E-Learning system. It is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data

input screen or another application. The data may be used to maintain one or more logical files. The data can be either control information or business information.
NO – *Number of Outputs*, Number of outputs produced by an E-Learning system. It is an elementary process in which derived data passes across the boundary from inside to outside.
NF –*Number of files*, Number of files used to process data in E-Learning system.
NIF – *Number of Interface files*, Number of files referenced by the application, but maintained with in the boundary of another application
NWP – *Number of web pages*, E-Learning system is a web application. So it may have huge number of web pages.
NSR – *Number of screens and reports*, E-Learning system used multiple screens and reports for receiving inputs and providing outputs.
DMF – *Duration of multimedia files*, E-Learning system has huge volume of multimedia files used to deliver learning content. They may be in the form of video, audio, animation or simulation.
NGF - *number of graphic files*, E-Learning system used graphical files for demonstrating the learning content. They may be in the form of Images, images with special effect, diagrams, text with special effects, structured tables, etc.
NDP – *Number of document pages*, It represents the number of pages used to express learning content in e-learning system.
NRDB – *Number of records in Databases*. It represents total number of records accepted by E-Learning system.

Algorithms to determine ULP

i. Determination of Unadjusted Learning Object points (ULOP). It is for calculating the count of learning objects.
ii. Determination of Unadjusted Other object Points (UOOP). It is for calculating the count of operational functionalities in a system.
iii. Calculate ULP. ULP is calculated by using equation 5. 2.

$$ULP = ULOP + UOOP \qquad \text{(Eqn – 5.2)}$$

Where ULOP- Unadjusted Learning Objects Points
UOOP- Unadjusted Other object Points

5.3.1 Determination of Unadjusted Learning Object Points (ULOP)

Components *DMF*, *NGF* and *NDP* are used to determine the size of learning content. This course content may be delivered in the form of full video, audio, textual document, document with simulation or animation otherwise combination of all. For small projects Calculate ULOP as a whole. For Large systems, If all the course contents having similar type of objects then calculate the ULOP of one course and multiply with number of courses(n). For all the courses in a system having numerous form of Learning objects then group them based on similarity, calculate the ULOP of Each group and sum up together to provide ULOP of a system. The DMF component assessed based on the duration of multimedia files deliver the learning objects. Count the number of Graphic Files involved in the course content. Count the document pages or number of slides delivers the course content. Find the complexity of each component. Table 5.1 assist to assign complexity of learning content. The complexity level to be assessed as Low, Average, High and Very high.

Table 5. 1 Learning complexity assessment

Components	**Low**	**Average**	**High**	**Very High**
Duration of multimedia files	1 to 15 hour	16 to 30 hour	31 to 45 hour	>45 hour
Number of graphic files	1 to 15 clips	16 to 30 clips	31 to 45clips	>45clips
Number of document pages	1 to 375 pages or slides then Transaction value should be 50	376 to 750 then Transaction value should be 100	751 to 1125 then Transaction value should be 150	>1125then Transaction value should be 200

After learning complexity assessment, assign proper weights. Table 5. 2 show the weights for learning complexity.

Table 5.2 Weights for Learning Complexity

Components	**Low**	**Average**	**High**	**Very High**
Duration of multimedia files	7	10	15	24
Number of graphic files	4	6	8	13
Number of document pages	50(Fixed Value)	100(Fixed Value)	150(Fixed Value)	200(Fixed Value)

The counts for NGF and NDP components and total time required for multimedia file can be entered into following Table 5.3. It is used to calculate Unadjusted Learning Objects. Each count is multiplied by the weight value shown in table 4.2 determine the rate of each component except NDP because developing NDP in E-Learning system is not complex as like coding so a fixed value given. The rated values on each row are summed across the table, giving a total value for each type of component. These totals are then summed down to arrive at the Unadjusted Learning Objects.

Table 5.3 Unadjusted Learning Object Points calculation

Components	**Low**	**Average**	**High**	**Very High**	**Total**
Duration of multimedia files	__*7= __	__*10=__	__*15=__	__*24=__	____
Number of graphic files	__*4=__	__*6=__	__*8=__	__*13=__	____
Number of document pages	50	100	150	200	____
Unadjusted Learning Object Points					____

5.3.2 Determination of Unadjusted Other Object Points (UOOP)

Count the number of all other components considered for determining UOOP. Rate them based on their complexity. Complexity should be specified based on the count of each component. Count is high then complexity is high, otherwise low or average. Table 5.4 assists to find the complexity of a component.

Table 5.4 Complexity assessment for other objects

Components	**Low**	**Average**	**High**	**Very High**
Number of Inputs	1-20	21-40	41-60	>60

Number of Outputs	1– 20	21-40	41-60	>60
Number of files	1– 10	11-25	26 – 50	>50
Number of Interface files	1-5	6-10	11-15	>15
Number of web pages	1 -5	6-10	11-20	>20
Number of screens and reports	1 -15	16 – 25	26 – 35	>35
Number of records in Databases	1– 10000 then Transaction value should be 5	10001 - 20000 then Transaction value should be 10	20001- 30000 then Transaction value should be 15	>30000 then Transaction value should be 25

Find the count and complexity of each component using above table 4.4 and fix the weight value of each component using table 5.5. The weight values are specified based on the good features of existing sizing techniques.

Table 5.5 Weight values for other objects

Components	**Low**	**Average**	**High**	**Very High**
Number of Inputs	3	4	6	10
Number of Outputs	4	5	7	12
Number of files	4	10	15	22
Number of Interface files	5	7	10	16
Number of web pages	5	10	15	20
Number of screens and reports	3	7	11	16
Number of records in Databases	3	4	6	10

The counts for all components can be entered into Table 5.6 other then Number of records in Databases because instead of count, use transaction value should be assigned for specifying UOOP of a component and Each count is multiplied by the weight value shown in the table determine the rated value. The rated values on each row are summed across the table, giving a total value for each type of component. These totals are then summed down to arrive at the UOOP of E-Learning System.

Table 5.6 UOOP Calculations

Components	**Low**	**Average**	**High**	**Very High**	**Total**
Number of Inputs	__*3=__	__*4=__	__*6=__	__*10=__	____
Number of Outputs	__*4=__	__*5=__	__*7=__	__*12=__	____
Number of files	__*4=__	__*10=__	__*15=__	__*22=__	____
Number of Interface files	__*5=__	__*7=__	__*10=__	__*16=__	____
Number of web pages	__*5=__	__*10=__	__*15=__	__*20=__	____
Number of screens and reports	__*3=__	__*7=__	__*11=__	__*16=__	____
Number of records in Databases	3	4	6	10	____
Unadjusted Other object Points (UOOP).					____

The resultant of unadjusted learning object points (ULOP) and Unadjusted Other object Points (UOOP), enough to calculate Unadjusted Learning Points(ULP). The Equation 5.2 is used for calculating ULP.

5.4 DETERMINATION OF TECHNICAL COMPLEXITY FACTOR

The technical complexity factor (TCF) is based on 14 general system characteristics (GSC's) that rate the general functionality of the application being counted[1-5]. Each characteristic has associated descriptions that help to determine the degrees of influence of the characteristics[2]. The degrees of influence range on a scale of zero to five, from no influence to strong influence. The General System Characteristics are represented in Table 5.7.

Table 5.7 General System Characteristics

General System Characteristics	
1	Data communications
2	Distributed data processing
3	Performance
4	Heavily used configuration
5	Transaction rate
6	On-Line data entry
7	End-user efficiency
8	On-Line update
9	Complex processing
10	Reusability
11	Installation ease
12	Operational ease
13	Multiple sites
14	Facilitate change

Once all the 14 GSC's have been answered, they should be tabulated, using TCF equation and It also similar to FPA's Value Adjustment Factor calculation. The rate of factors varies from 0 to 5. The given factor not used in system then rate is 0. Otherwise the rate of a given factor is represented based on its influence.0 - No influence, 1 – Incidental,2- Moderate, 3 – Average, 4 – Significant and 5 – Essential. Sum up all the influence level of each component and called as Tsum. The TCF is calculated using equation 5.3.

$$TCF = 0.65 + (0.01 * Tsum) \qquad \text{(Eqn - 5.3)}$$

5.5 DETERMINATION OF LEARNING COMPLEXITY FACTOR (LCF)

LCF addresses the skills, performance and abilities of development environment. This factor influences to productivity and stability. There are six factors [2] associated with LCF calculation is represented in the table 5.8.

Table 5.8 Factors to assess learning complexity and their weights

Factor	Description	Weight
F1	Familiar with E-Learning system development(FESD)	1.5
F2	Analyst capability(AC)	0.5
F3	Motivation(M)	1
F4	Requirement stability(RS)	2
F5	Number of courses(NC)	2
F6	Expected Students strength(ESS)	2

Rate each factor's influence from 0 to 5. Zero denotes no experience in e-learning system development; poor in analyst capability, motivation and requirement stability and courses and strength. Five denotes high experience in e-learning system development; very high in analyst capability, motivation, requirement stability, courses and strength are Expected level. For each factor, multiply the degree of influence by the weight, and sum all products to obtain Learning complexity sum, Lsum. Compute LCF using Equation 5.4.

$$LCF = 1.4 - 0.03 * Lsum \quad \text{(Eqn – 5.4)}$$

5.6 DETERMINATIONOF LOP

To determine LOP, three important factors are necessary, they are Unadjusted Learning Points(ULP), Technical complexity factor (TCF), Learning complexity Factor(LCF). After determine these three factors, compute Learning object point (LOP) using the equation 5.1.

Apply the above steps to calculate LOP of any kind of E-Learning system. The following chapter 6 analyses the performance of LOP method in terms of size, effort, duration and cost with FPA method. The performance analysis prove LOP is more accurate and optimal for sizing E-Learning system.

Comparison about LOP over FPA based on the features of E-learning system

The software project management view LOP is trustful for quantifying the size of E-learning system. The following Table 4.9 Shows the comparison between LOP and FPA based on the features of E-Learning system. LOP considers all the features for sizing than FPA.

Table 6.11 Comparison between LOP and FPA based E-Learning system features

Techniques/Features of E-Learning System	**Function Point Analysis**	**Learning Object Point**
Input and output consideration	Inputs and outputs are the two separate components for calculating Unadjusted function points	Inputs and outputs are the two separate components for calculating Unadjusted Learning points
Logical files involvement	Count all files for calculating Unadjusted function points	Count all files for calculating Unadjusted Learning points
Interface files	Count all interface files for calculating Unadjusted function points	Count all interface files for calculating Unadjusted Learning points
Web pages	Nil	Good for sizing web pages
Screens and reports (GUI Support)	Calculate inputs and outputs but it never identify the worth of screens and reports	Major components to calculate Unadjusted Learning points
Multimedia files – video, audio, simulation, animation	Nil	All kind of multimedia files are considered and their complexity are assessed for sizing

Techniques/Features of E-Learning System	Function Point Analysis	Learning Object Point
Graphic files	Nil	All kind of Graphic files are considered and their complexity are assessed for sizing
Textual document	Nil	Number of textual pages are encounter for sizing
Accuracy in sizing of E-Learning system	Inaccurate	Accurate
Quality in sizing of E-Learning system	Nil	Provides Expected quality
Reusability	Considered	Considered
Database Support	Nil	Highly considered
Data communication	Considered as one of the complexity adjustment factor	Highly considered

5.7 BENEFITS OF LEARNING OBJECT POINTS METHOD

i. LOP can be used to size E-Learning applications accurately. Sizing is an important component in determining productivity, Planning and Management.
ii. It is easily understood by the non-technical user. This helps in communicating sizing information to a user or customer.
iii. Conversion to LOC is similar to FP to LOC conversion.
iv. It also supports to estimate any kind software application other than E-Learning system also.
v. Estimate development effort and Cost benefit analysis using LOP
vi. To Derive Business Decisions.

CHAPTER 6

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.

6.1 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.

6.1.1 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

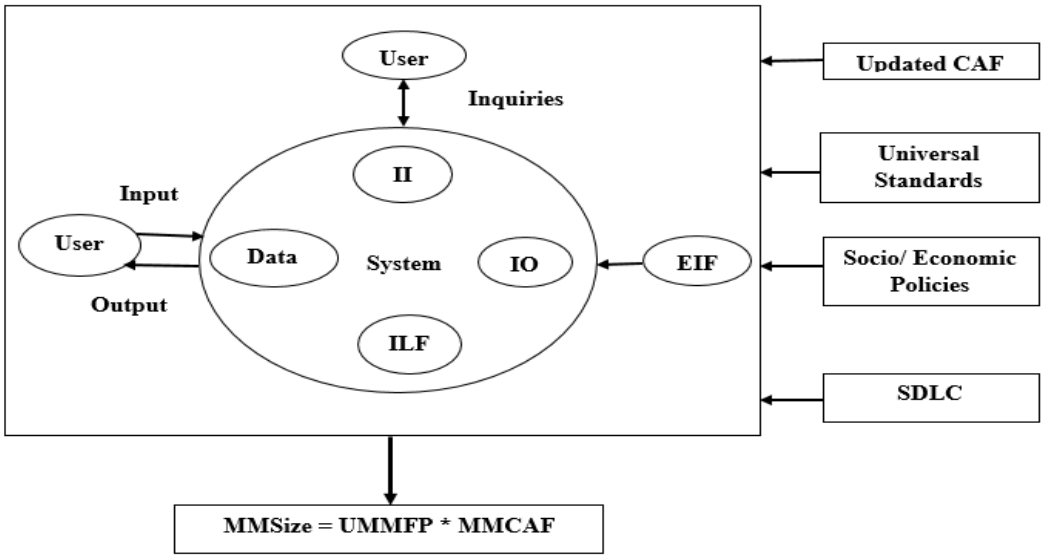

Figure 4.1: Architecture of Modern Metrics

6.1.2 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.

6.1.3 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.

6.1.4 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

S. No	EQ Functional Values	EQ Metrics	EQ Metric Values
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.

6.1.5 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									

6.1.6 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 use more than one network topologies?
19. Does the system install in different nations and uses different social, cultural, economic and environmental laws?
20. Does the system give multiple forms of output?
21. Does the trial version and model version of software development affect 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.

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

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

6.1.7 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.

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

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.

6.1.8 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)

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

The MMSize is the product of UMMFP and MMCAF

6.2 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.

6.3 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.

6.3.1 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)

6.3.2 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 (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.

6.3.3 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 * number of persons involved in the software development) (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.

6.3.4 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)

6.4 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.

Appendix A

Software Engineering
software engineering can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.
Project management
Project management is the discipline of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria.
Software project management
Software project management is theprocess of planning and leading software projects. It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled.
Project planning
Project planning is part of project management, which relates to the use of to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined.
Software development effort estimation
Software development effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours) required to develop or maintain software based on incomplete, uncertain and noisy input.
E-learning
The delivery ofan education program by electronic means. E-learning involves the use of a computer or electronic device (e.g. a mobile phone) in some way to provide training, educational or learning material.
E-Learning system.
E-Learning system is a software system supported for E-Learning
Internet
Internet is a global computer network providing a variety of information and communication facilities using standardized communication protocols.
Input
Any information or data that is sent to a computer for processing is considered *input.*
Output
A result produced by a computer that is internal to the system (from one program or process to another) or external to it (from a program or process to an *output* device) but internal to an*output* device (modem, monitor, printer, etc.).
Animation
Animation is the technique of photographing successive drawings or positions of puppets or models to create an illusion of movement when the film is shown as a sequence.
Simulation
Simulation is the production of a computer model of something, especially for the purpose of study.
Video
A sequence of images processed electronically into an analog or digital format and displayed on a screenwith sufficient rapidity as to create the illusion of motion and continuity.

Document

Document is a piece of written, printed, or electronic matter that provides information or evidence or that serves as an official record.

Software sizing

Software sizing is an activity in software engineering that is used to quantify thesize of a software application or component in order to be able to implement othersoftware project management activities (such as estimating or tracking)

Function points

A function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user.

Audio

Audio is defined as anything related to sound in terms of receiving, transmitting or reproducing or its specific frequency.

Learning Object Points

Learning Object Points is a unit of measurement to express the amount of learning objects and operational functionalities an e-learning system provides to a user.

Productivity factor

Productivity factor defines the amount of time required for complete one function point. It will be change from organization to organization and country to country.

REFERENCES

1. Capers Jones, (2008), "Applied Software Measurement-Global Analysis of Productivity and Quality",Tata McGraw Hill, Third Edition, pp 71-182.
2. Richard D. Stutzke, (2005), "Estimating Software-Intensive Systems: Projects, Products and processes", SEI Series in Software Engineering, Addison Wesley Edition, pp 1-786.
3. Capers Jones, (2007), "Estimating Software Costs: Bringing Realism to Estimating", Tata McGraw Hill, Second Edition, pp 3-629.
4. Capers Jones, (2010), "Software Engineering Best Practices: Lessons from Successful projects in the top companies", Tata McGraw Hill Edition, pp 1- 643.
5. Robert t. Futrell, Donald F. Shafer, Linda I. Shafer, (2008), "Quality software project management", Pearson Education, pp 1-600.
6. Mehwish Nasir, H. Farooq Ahmad, "An Empirical Study to Investigate Software Estimation Trend in Organizations Targeting CMMI[SM], 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, 2006.
7. Kenneth Lind, Rogardt Heldal, "Estimation of Real-Time Software Code Size using COSMIC FSM", IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 2009, pp.244-248.
8. Mahir Kaya, Onur Demirörs, "E-Cosmic: A Business Process Model Based Functional Size Estimation Approach", 37th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE, 2011, pp.404-408.
9. Ursula Passing, Martin Shepperd, "An experiment on software project size and effort estimation", International Symposium on Empirical Software Engineering (ISESE'03), IEEE, 2003.
10. Daniel V. Ferens, "Software size estimation techniques", Air Force Institute of Technology (AFIT/ISY), Wriqht-Patterson AEB, Ohio 45433, pp701-706.
11. Kjetil Molokken and Magne Jorgensen, "A Review of Surveys on Software Effort Estimation", International Symposium on Empirical Software Engineering (ISESE'03), IEEE, 2003.
12. Z. Zia, A. Rashid, K.uzZaman, "Software cost estimation for component based fourth-generation-language", IET Software, 2010,pp 103-110.
13. Md.Forhad Rabbi, Shailendra Natraj, Olorisade Babatunde Kazeem, "Evaluation of convertibility issues between IFPUG and COSMIC function points", Fourth International Conference on Software Engineering Advances, IEEE, 2009, pp.277-281.
14. Noureldin A.Z Adem , Zarinah M. Kasirun ,"Automating Function Points Analysis Based on Functional and non-functional Requirements Text", IEEE, 2010, pp 664-669.
15. A. Ferchichi , J.P. Bourey, M. Bigand, M. Barron , "Design systems engineering of software products: implementation of asoftware estimation model", IMACS Multi conference on "Computational Engineering in Systems Applications"(CESA), October 4-6, 2006, Beijing, China, pp1181-1188.
16. Juan J. Cuadrado-Gallego, Pablo Rodríguez-Soria, Saahil Hakimuddin, "Early functional size estimation with IFPUG unit modified", 9th IEEE/ACIS International Conference on Computer and Information Science, IEEE, 2010.
17. June Verner and Graham Tate, "A Software Size Model", IEEE Transactions on Software Engineering, VOL. 18, NO. 4, APRIL 1, 1992, pp 265-278.

18. Edilson J. D. Cândido, RoselySanches, "Estimating the size of web applications by using a simplified function point method", IEEE, 2003.
19. Linda M. Laird, "The Limitations of Estimation", IT Pro November| December2006 Published by the IEEE Computer Society IEEE, 2006, pp 40 – 45.
20. Alen Jakupovic , Mile Pavlic , Patrizia Poscic ,"Estimation of the Size of Business Sectors Covered by ERP Solutions", Fifth International Multi-conference on Computing in the Global Information Technology, IEEE, 2010, pp 60-64.
21. Steven Fraser, Barry Boehm, HakanErdogmus, MagneJørgensen, Stan Rifkin, Mike Ross,"The Role of Judgment in Software Estimation", ICSE'09, Vancouver, Canada, 2009 IEEE, pp 13-17.
22. Daniel V. Ferens, " The Conundrum Software Estimation Models", Air Force Research Laboratory (AFRLIIFSD), IEEE, 1999 , pp 23-29.
23. Rodrigo C. Barros, Duncan D. Ruiz,Nelson N. Tenório Jr., Márcio P. Basgalupp,"Issues on Estimating Software Metrics in a Large Software Operation", 32nd Annual IEEE Software Engineering Workshop, IEEE, 2009, pp 152-159.
24. Cigdem Gencel, Rogardt Heldal, Kenneth Lind, "On the Relationship between Different Size Measures in the Software Life Cycle", 2009 16th Asia-Pacific Software Engineering Conference, IEEE, 2009, pp 19-26.
25. Iman Attarzadeh, Siew Hock Ow, "Proposing a New High Performance Model for Software Cost Estimation", Second International Conference on Computer and Electrical Engineering, IEEE, 2009 pp 112-116.
26. Luigi Buglione and Christof Ebert, "Estimation Tools and techniques", IEEE , 2011, pp 92-96.
27. Gustavo Bestetti Ibarra, PatríciaVilain, "Software Estimation Based on Use Case Size", Brazilian Symposium on Software Engineering, IEEE,2011, pp178 -187.
28. Erika Corona, Michele Marchesi, GiulioBarabino, Daniele Grechi, Laura Piccinno, "Size Estimation of Web Applications through Web CMF Object", IEEE 2012,14-20.
29. Najberg, Andrew C.,(1984), "Software Data Base Development, Volume I, Reading, MA, The Analytic Sciences Corporation ", (Technical Report 4612-S-1), p.2-4.
30. Boehm, Barry W., "Software Engineering Economics", Tutorial, Software Management: Third Edition, Washington, DC, IEEE Computer Society Press: 1986, p.148.
31. Boehm, Barry W., "Software Engineering Economics", Englewood Cliffs, NJ,Prentice Hall, 1981.
32. Abran A., Cuadrado J., and Desharnais J. M., Convertibility of Function Points to COSMICFFP: Identification and Analysis of Functional Outliers, MENSURA 2006, Cadiz, Spain, pp. 6-8.
33. ISO/IEC19761:2003, "Software Engineering – COSMIC-FFP A Functional Size Measurement Method", International Organization for Standardization - ISO, Geneva, 2003.
34. 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, 2005.
35. ISO/IEC 20968: 2002, "Software engineering – Mk II Function Point Analysis – Counting Practices Manual", International Organization for Standardization – ISO, Geneva, 2002.
36. ISO/IEC 20926: 2003, "Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual", International Organization for Standardization – ISO, Geneva, 2003.

37. B. W. Boehm,(1981), “Software Engineering Economics” Prentice-Hall, 1981.
38. L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering, 1978, pp. 345-361.
39. G. Karner, "Resource Estimation for Objectory Projects," Objective Systems, 1993.
40. D. D. Galorath and M. W. Evans, “Software Sizing, Estimation, and Risk Management”, Boston, MA, USA: Auerbach Publications, 2006.
41. R. T. Hughes, "Expert judgement as an estimating method," Information and Software Technology, 1996, pp. 67-75.
42. N. Dalkey and O. Helmer, "An Experimental Application of the Delphi Method to the Use of Experts," Management Science, 1963 pp. 458-467.
43. J. Lynch, “Chaos manifesto”, The Standish Group, Boston, 2009[Online]. Available:http://www.standishgroup.com/newsroom/chaos_2009.php.
44. Demirors and C. Gencel, "A Comparison of Size Estimation Techniques AppliedEarly in the Life Cycle",Software Process Improvement, vol. 3281, 2004, pp.184-194.
45. G. Kotonya and I. Sommerville, ”Requirements Engineering: Processes and Technique”,Chichester; New York: John Wiley, 1998.
46. D. Eck, B. Brundick, T. Fettig, J. Dechoretz and J. Ugljesa,(2009) "Parametric estimatinghandbook", The International Society of Parametric Analysis, Fourth Edition.
47. Allen, I. E. and Seaman, J. Staying the Course: Online Education in the United States, 2008 NeedhamMA: Sloan Consortium.
48. Allen, I.E. and Seaman, J., (2003),“Sizing the Opportunity: The Quality and Extent of Online Education in the United States”, 2002 and 2003 Wellesley, MA: The Sloan Consortium.
49. Shukor Sanim M. Fauzi, M. HairulNizam M. Nasir, Nuraminah R., KamaruzamanJusoff,N. Azylia A. Azam, M. Hafiz Ismail, “The Implementation of Software Process Improvement Models”, International Review on Computers and Software,Vol. 4. n. 3, pp. 402 – 407, May 2009.
50. Muneer Alsurori, JuhanaSalim, “The Status of Information and Communication Technology in the Higher Education in Yemen”, International Review on Computers and Software, Vol. 5 N. 6,pp. 712-723,November 2010.
51. Charalambos Vrasidas,(2004), “ Issues of Pedagogy and Design in e-learning Systems”, ACM Symposium on Applied Computing, pp 911-915.
52. Catherine S. Fichten et el,(2009), "Disabilities and e-Learning Problems and Solutions: An Exploratory Study", International Forum of Educational Technology & Society, pp 241–256.
53. T. S. Shiny Angel, Paul Rodrigues,(2012) “ELSE:E-Learning System Estimator”, International Review on Computers and Software, Vol. 7 n. 6, pp. 3033-3036, Nov- 2012 (Scopus Indexed).
54. T. S. Shiny Angel, Paul Rodrigues, (2012) “Size and Flexibility Metrics for E-Learning System”, International Journal of Software Engineering and Technology, Vol. 4, No 7 , July 2012.
55. T. S. Shiny Angel, Paul Rodrigues,(2012)“Limitations of Function Point Analysis in E-Learning System Estimation”, International Journal of Computational Engineering Research, pp. 156 -161, July 2012.
56. T. S. Shiny Angel, Paul Rodrigues,(2013),”A Sizing Approach for E-Learning System”, International Journal of Emerging Trends and Technology, Special Issue, July 2013.
57. T. S. Shiny Angel, Paul Rodrigues,(2015), “Comparative Analysis of Sizing Techniques in the sense of E-Learning system”, International Journal of Applied Engineering Research, Volume 10, Number 16,pp 36303-36312, August 2015.(Scopus Indexed).

58. T.S. Shiny Angel, Paul Rodrigues,(2015), "Estimating the Size of E-Learning System using Learning Object Points Method", Indian Journal of Science and Technology, September 2016. (Scopus Indexed).
59. Paul A. Below, Poulsbo,(2007), "System and method for function point sampling for software size estimates", US 7,213,234 B1, United States Patent.
60. Renjeev V. Kolanchery, Harish Ranganath,(2007) "Project Size Estimation Tool", US 2007/0276712 A1, United States Patent.
61. Filip Misovski,(2007), "Estimating Development Of New User Interface", US 2007/0265779 A1, United States Patent.
62. Lavanya Srinivasan, Steven S. Stefan,(2010), "Enhanced Function Point Analysis", US 7,743,369 B1, United States Patent.
63. Humphrey Watts.S,(2004), "Managing the Software Process", SEI Series in Software Engineering, Pearson Education, Singapore.
64. Henry Joel, (2008), "Software Project Management A Real-World Guide to Success", Pearson, India.
65. Gopalaswamy Ramesh, (2013), "Managing Global Software Projects", Mc Graw Hill Education, India.
66. John T Mesia Dhas and Bharathi C.R. (2018), "Modern Metrics (MM): Size Estimation of Modern Software and its Metrics Analysis" Journal of Web Engineering, Vol. 17, No. 6, pp. 1851-1861
67. John T Mesia Dhas and Bharathi C.R. (2017), "E-Commerce System Size using User Based Function Points", International Journal of Applied Engineering Research, Vol. 12, No. 16, pp. 6115-6122.
68. John T Mesia Dhas and Bharathi C.R. (2016), "Risks Associated to Size Estimation of E-Commerce System using Function Point based Estimation Techniques", Indian Journal of Science and Technology, Vol. 9, No. 7, pp. 1-6.
69. John T Mesia Dhas and Bharathi C.R. (2015), "Relative Analysis of Sizing Methods in the sense of E-Commerce system", International Journal of Applied Engineering Research, Vol. 10, No.18, pp. 39808-39816.
70. John T Mesia Dhas and Bharathi C.R. (2018), "A Study on Functional Requirements in Function Point (FP) based Size Estimation on Modern Software", IEEE Conference, International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing, Priyadarshini Engineering College, pp. 77-83.

ABOUT AUTHORS

T. S. Shiny Angel received her Ph.D. in Computer Science and Engineering from SRM University, Chennai, India. She has 19 years of Experience in the field of Education and Industry, currently she is working as an Assistant Professor Sr. Grade with Software Engineering Department of SRM Institute of Science and Technology (formerly known as SRM University), Chennai, Tamil Nadu, India.

She is also doing researches in Software Engineering, Machine Learning and Data Analytics fields. She has published more than 45 research papers in conferences and Journals.

John T Mesia Dhas received his Ph.D. in Computer Science and Engineering from Vel Tech University, Chennai, India. He has 16 years of Experience in the field of Education and Industry, currently he is working as an Associate Professor with Computer Science and Engineering Department of Audisankara College of Engineering and Technology, Gudur, Andhra Pradesh, India under Jawaharlal Nehru Technological University Anantapuramu.

He is also doing researches in Software Engineering and Data Analytics fields. He has published more than 22 research papers in conferences and Journals.

ISBN: 978-93-5437-820-1
Price: ₹. 250/-

OTHER BOOKS

S. No	Title	ISBN
1	C LOGIC PROGRAMMING	978-93-5416-366-1
2	Modern Metrics (MM): The Functional Size Estimator for Modern Software	978-93-5408-510-9
3	PYTHON 3.7.1 Vol - I	978-93-5416-045-5
4	SOFTWARE SIZING APPROACHES	978-93-5437-820-1
5	DBMS PRACTICAL PROGRAMS	978-93-5437-572-9
6	SERVICE ORIENTED ARCHITECTURE	978-93-5416-496-5

For free E-Books: jtmdhasres@gmail.com

ISBN: 978-93-5437-820-1
Price: ₹. 250/-