Function Point Analysis: Converting various design elements into Function Points

Ms. Bhawna Sharma[1], Ms. Kavita Choudhary[2], Mr Rajendra Purohit[3]
[1]M.Tech. Scholar, [2]Associate professor, [3]Research Scholar
Department of CSE
JIET, Jodhpur, India
bvnbhati@gmail.com

Abstract— **Determining functional size estimate of a development project is an important and primary step while planning a project. Functional Size estimation further affects other estimations like cost, budget and schedule of the project. There are many popular methods to determine functional size of a software project. Function Point Analysis (FPA) is one of the most used methods among these. Literature survey shows lot of work has been done on performing FPA of different projects. Most of these works begin with identification and categorization of functions which are needed as input parameters to FPA. There are generic guidelines available for this phase of FPA. In this work we have covered initial phase of FPA whereby we have some requirements specifications of a project represented by various notations like Use Case Diagrams, Class Diagrams and Data Flow Diagrams. We have developed some simple and specific mapping rules to convert these design specification into data and transaction function of FPA.**

Keywords- Software estimation, Functional Size Measurement Methods (FSMM), Function Point Analysis (FPA)

I. INTRODUCTION

The success of a software project is a function of many different factors involving completeness of requirements specified, completion in time, schedule should not overrun, and cost incurred should be in optimal range of estimated cost. In simplest terms all these mean actual duration and cost should closely match with estimated duration and cost [8, 13]. And all these depend on first stage of estimation that involves functional size measurement of the software.
In this paper we will analyze various design specification for the purpose of finding Functional Size Measurement of a project [9, 10]. We have chosen a Library Management System for this work. We will go through most popular design models from both types of project modeling approaches - Use Case Diagrams and Class Diagrams commonly used by Object oriented modeling [3, 4] and Data Flow Diagrams commonly used by Structural Modeling approaches.

II. FUNCTION POINT ANALYSIS (FPA)

Function Point Analysis (FPA) is a Functional Size Measurement Method which was introduced by Albrecht in 1979 [1]. It measures functional size and complexity of software from user's perspective, and uses Function Point (FP) as units to measure the size. FP measures size in logical terms and so it is not dependent on technology used to implement the software [11]. This makes FPA consistent for various projects implemented in different languages. Effort required to implement each FP off course depends on technology used [11, 12].
Process of calculating size in terms of FPs [2] is as simple as identifying and counting data and transaction functions and assigning FPs to these functions based on complexity of each function. Various functions are categorized as one of the following

A. Data Functions

1 Internal Logical File (ILF) - These are data entities maintained within application boundary.
2 External Interface File (EIF) - These are data entities referred by application but not maintained within applications. These are availed by some external application.

B. Transaction Functions

3 External Input (EI) – Any data or control information entering the application boundary. These are used to maintain Internal Logic Files.
4 External Output (EO**)** - Any data or control information leaving the application boundary. These are generally either direct information from ILF or some processed information based on data stored in ILFs.
5 External Inquiry (EQ) – Any input data meant for generating some output from ILFs. Input data generally represent some criteria used to process and retrieve an output.

Table 1 Complexity for External Inputs

FTRs	Data Elements		
	1-4	5-15	> 15
0-1	Low	Low	Avg
2	Low	Avg	Hgh
3 or more	Avg	Hgh	Hgh

FPA begins with identifying all data and transaction functions for an application. Further the Function Point Count of the project is calculated in 3 steps:

1. *Determining Functional complexity of all Functions*

Complexity of *transaction* function is based on number of Data Element Types (DETs) referenced and File Types Referenced (FTR). Complexity of *data* function is based on number of Data Element Types (DETs) referenced and Record Element Types (RETs) referenced. Tables 1, 2 and 3 provide detailed criteria for selecting complexity of functions.

2. *Calculating Unadjusted Function Point (UFP) Count of the System*

An Unadjusted Function Point (UFP) Count is then obtained for each function using Table 4. Sum of these FPs, represents total UFP for the project.

3. *Determining Value Adjustment Factor (VAF) and calculating adjusted Function Point Count*

The value adjustment factor (VAF) is based on 14 general system characteristics (GSC's) (Table-5) that rate the general functionality of the application being estimated. Each characteristic has associated descriptions that help 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. Add the degrees of influence for all 14 general system characteristics to produce the total degree of influence (TDI).

$$VAF = (TDI * 0.01) + 0.65$$

The final Function Point Count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF).

$$FP = UAF * VAF$$

Table 2 Complexity for External Outputs and Inquiries

FTRs	**Data Elements**		
	1-5	**6-19**	**> 19**
0-1	Low	Low	Avg
2-3	Low	Avg	Hgh
4 or more	Avg	Hgh	Hgh

Table 3 Complexity for Data Files

RETs	**Data Elements**		
	1-19	**20-50**	**> 50**
0-1	Low	Low	Avg
2-5	Low	Avg	Hgh
6 or more	Avg	Hgh	Hgh

Table 4 Complexity weights of FPA components

Function type	**Complexity**		
	Low	**Average**	**High**
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6
Internal Logical File	7	10	15
External Interface File	5	7	10

Table 5 FPA – General System Characteristics

General System Characteristic		Brief Description
1	Data communications	Complexity of communication facilities.
2	Distributed data processing	Level of complexity of distributed data and processing functions.
3	Performance	In terms of response time and throughput.
4	Heavily used configuration	Level of usage of current hardware platform where the application will be executed.
5	Transaction rate	Frequency of transactions executed (daily, weekly, monthly)
6	On-Line data entry	Percentage of the information is entered On-Line
7	End-user efficiency	End-user efficiency required.
8	On-Line update	Percentage of On-Line transactions.
9	Complex processing	Volume of extensive logical or mathematical processing.
10	Reusability	Scope of components reuse.
11	Installation ease	Ease of installation.
12	Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

III. OBJECT ORIENTED OR UML MODEL

Most of the projects developed these days use Object Oriented Programming paradigm. UML (Unified Modeling Language) modeling is a standard set of diagrams used by project managers to represent design specifications. Various structural and behavioral diagrams comprise UML specifications of a project. In this work we have used Use-Case diagrams among behavioral models and class diagrams among structural models.

Use-case diagram has three major components – actor, use-case and system. Systems are represented by rectangle and use-cases as ovals. An actor is basically a user and its connection with use cases is shown using an arrow. Class diagram is a diagram that shows a collection of the declaration of the model elements, such as classes, class member functions and methods and relationships between classes.

IV. STRUCTURAL MODEL

Structural Modeling is the design modeling that was used for old projects when programming paradigm used was process oriented and not Object oriented. It is still relevant in current projects when any project is process intensive in nature. Process intensive applications typically have well defined set of data entities used by system and processes to maintain those systems. Data Flow Diagrams (DFD) are one of the most common modeling elements among all. DFDs represent data flow between functions and database of the system. A Data Flow Diagram (DFD) has 4 components – Function, Input/output, Flow and File/database. These 4 components are shown using circle, rectangle, open box and directed lines respectively.

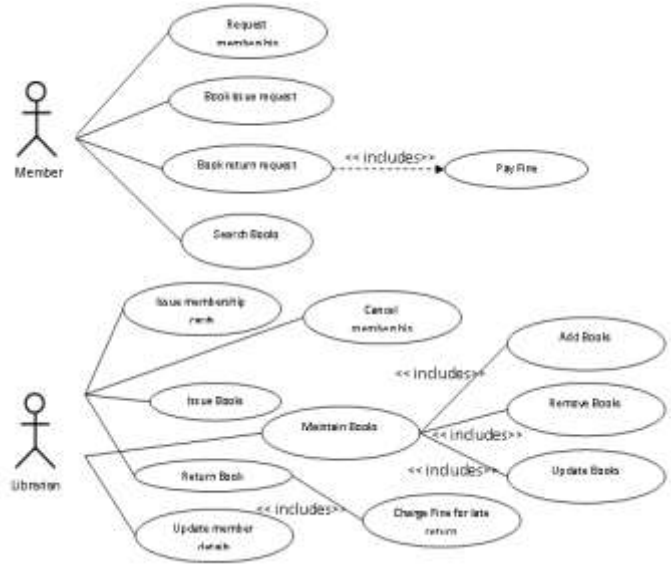

Figure 1 - Use-Case Diagram of Library Management System

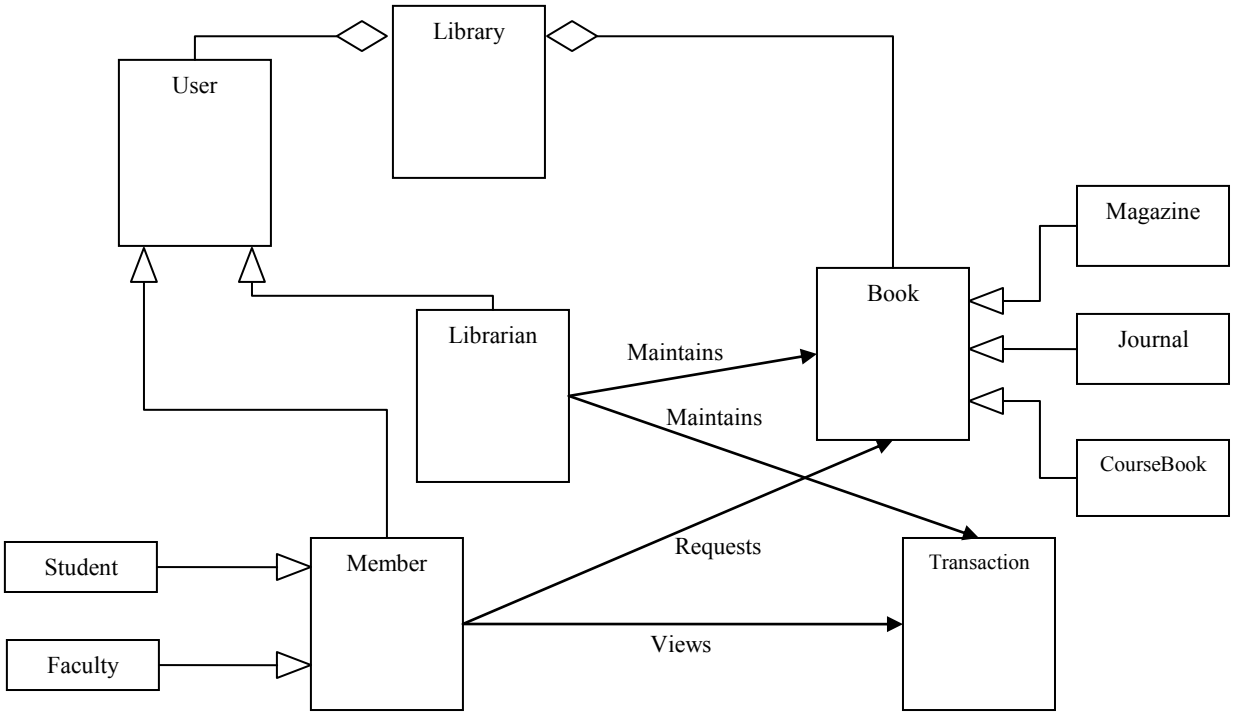

Figure 2 Class Diagram of a Library Management System

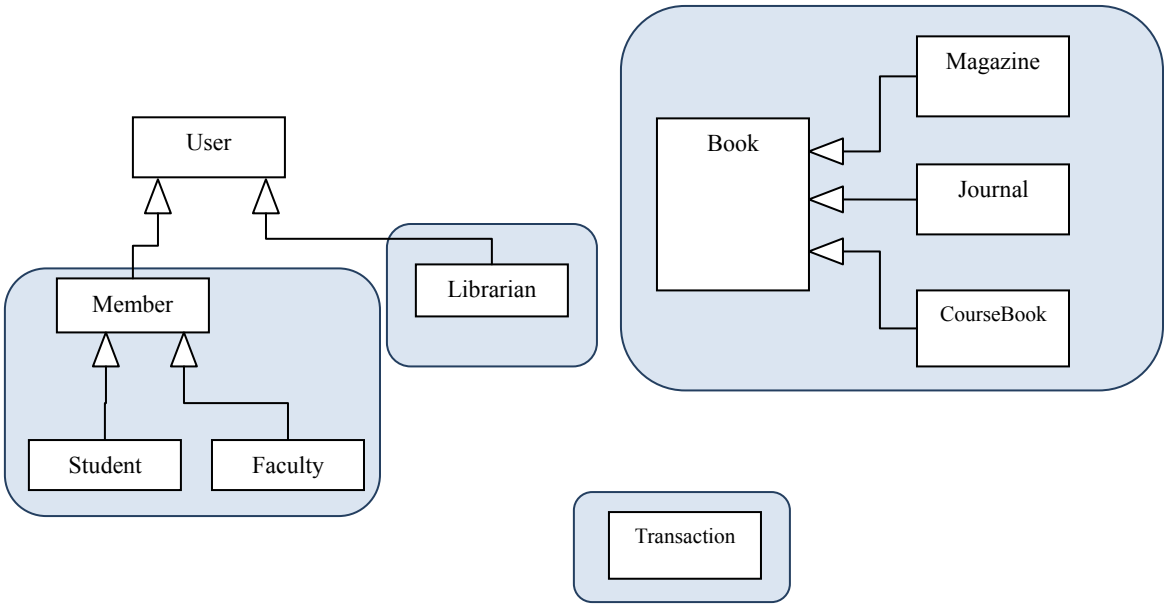

Figure 3 Data Function identified from Class Diagram

V. ANALYSIS OF LIBRARY MANAGEMENT SYSTEM

A. Identifying FPA components based on Use Case Diagram and Class Diagram

Mapping Use Case Diagram and Class Diagrams into Data and transaction function of FPA is done as follows [5, 6]. If we look at Class Diagrams most classes represent data files being maintained by the projects. Inheritance hierarchy of few classes is as simple as considering a whole class hierarchy as a single data function. Dependencies among classes or various member methods of class represent transaction functions. Similarly if we look at use case diagrams, subjects being maintained by use-case represent data functions and phrases representing an action on that subject are a transaction function.

Figure 1 shows Use case Diagram of a Library Management System, Figure 2 shows Class Diagram of same project. Figure 3 shows a breakdown of the class diagram into various data functions namely – Member, Librarian, Book and Transaction. As all these entities are maintained within application they are all Internal Logic Files (ILF). All transaction functions related to these ILFs can be obtained from Use-Case Diagram. Table 6 thus obtained shows complexity of all these functions. Complexity of these functions can be derived by using tables 1, 2 and 3. Requesting a book issue and return by member are considered as part of actual action of issue and return from librarian, and so are not mentioned as separate transactions.

B. Identifying FPA components based on Data Flow Diagram

Figure 4 shows DFD of a Library Management System. If we look closely at it we can see how easily it can be mapped into data and transaction functions for the purpose of Function Point analysis [7]. All data files directly represent data functions. If there is no input to these data files, that means they are external reference files. If data files are being sent some input they represent Internal Logic Files. All data flows connected to functions represent transaction function. Arrows of these flows tell us about category of transaction functions. Flow coming in to a function is External Input, Flow out of function is External Output and Bidirectional arrows represent External Inquiry.

Interestingly, if we now follow above rules for mapping and create a tables for function point calculation it will same as Table 6 – except that in class diagram we have chosen Librarian also as an ILF. This was a deliberate effort to create a minor difference just to point out the fact that level of refinement in these diagramming techniques can cause difference in resulting FP count. Provided both modeling techniques use same level of refinement we will get same FP count after mapping of design elements to FP components process.

We are not showing table for various data and transaction in this section for this reason only- otherwise it will look repetition of same items and calculation.

Table 6 Functions identified for Library Management System

Name	Type	Complexity	FP Count
TRANSACTIONS			
Search Book	EQ	Low	5
Create Member	EI	Low	3
Delete Member	EI	Low	3
Update Member Details	EI	Low	3
Add Book	EI	Low	3
Update Book Details	EI	Low	3
Remove Book	EI	Low	3
Issue Book	EI	Low	3
Return Book	EI	Low	3
Calculate Fine	EQ	Low	5
FILES			
Member	ILF	Low	7
Librarian	ILF	Low	7
Book	ILF	Low	7
Transaction	ILF	Low	7
		Total Unadjusted FP	**62**

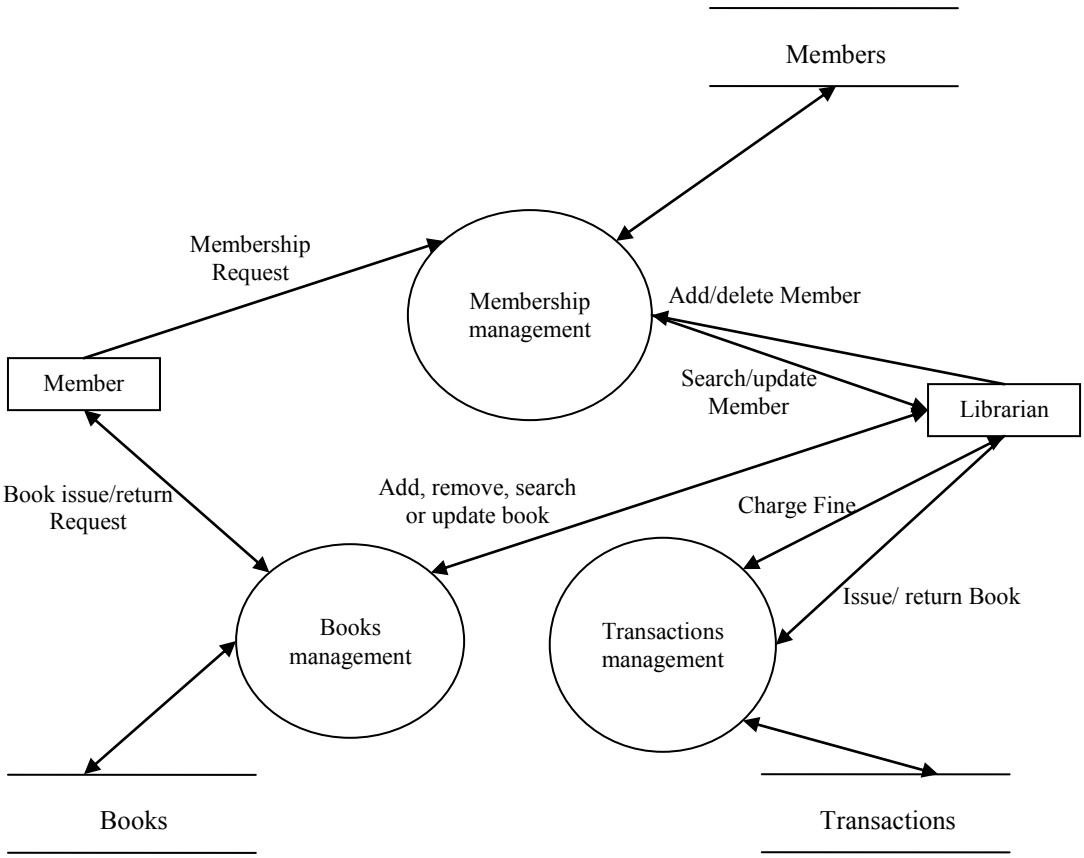

Figure 4 Data Flow Diagram of Library Management System

VI. CONCLUSION

By the help of above work we reach on following conclusion:
Function Point Analysis method can be effectively applied for Functional Size Measurement of a project irrespective of type of design models. If we observe carefully most of the design models which are used for requirements specification including the ones used in this work somehow have to specify functions to be performed by the project in way that we can identify data maintained by these functions and input or output requirements for these functions. We have successfully demonstrated it in this work. We showed -while performing Function Point Analysis of a project, simple mapping rules can be applied to identify components of FPA from various design models. Once we define these rules for any design element it becomes easier to identify components of FPA and also it ensures that none of the elements remains unidentified.
Also, we can conclude that FP count of any project derived from its design specifications depends on level of refinement of design specification instead of type of modeling or diagramming techniques. We will reach to almost same size irrespective of the fact that we are using structural modeling or object oriented modeling provided that level f refinement of these models is same.

REFERENCES

[1] A.J Albrecht. Function point analysis. Encyclopedia of software engineering, 1: John Wiley & Sons, 1994.
[2] Function Point Counting Practices Manual Release 4.3.1 (January 2010) https://ainfo.cnptia.embrapa.br/digital/bitstream/item/34989/1/0004-3-1-Part-0-2010-01-17.pdf
[3] Ribu, Kirsten. 2001. Estimating Object-Oriented Software Projects with Use Cases. Master of Science Thesis, University of Oslo, Department of Informatics. Schwaber, Ken and Mike Beedle. 2001. Agile Software Development with Scrum. Prentice Hall.
[4] A. Zivkovic, R. Ivan, and H. Marjan, "Automated Software Size Estimation based on Function Points using UML Models", Information and Software Technology, Elseiver, 2005
[5] T. Fetcke, A. Abran, and T. Nguyen, "Mapping The OO-Jacobsen Approach to Function Points", Proceedings of Tools 23'97 – Technology of Object Oriented Language and Systems, IEEE Computer Society Press, California, 1998
[6] K.v.d. Berg, D. Ton, and O. Rogier, "Functional Size Measurement Applied to UML-base User Requirements", Retrievable from doc.utwente.nl, 2005

[7] E. Lamma, P. Mello and F. Riguzzi, "A system for measuring function points from an ER-DFD specification," The Computer Journal , vol. 47, no.3, pp.358-372, 2004
[8]. Matson, J. E., Barrett, B. E., & Mellichamp, J. M. (1994). Software development cost estimation using function points. Software Engineering, IEEE Transactions on, 20(4), 275-287.
[9] F. Gramantieri, E. Lamma, P. Mello, and F. Riguzzi, "A System for Measuring Function Points from Specifications," DEIS – Universita di Bologna, Bologna. and Dipartimen to di Ingegneria, Ferrara, Tech. Rep DEIS-LIA-97-006, 1997.
[10] H. Mehler and A. Minkiewwicz, "Estimating size for object-oriented software," in Proceeding of ASM'97 Application in Software Measurement, Berlin, 1997.
[11]. Felfernig, A., & Salbrechter, A. (2004). Applying function point analysis to effort estimation in configurator development. In International Conference on Economic, Technical and organizational aspects of Product Configuration Systems, Kopenhagen, Denmark (pp. 109-119).
[12]. Jeng, B., Yeh, D., Wang, D., Chu, S. L., & Chen, C. M. (2011). A Specific Effort Estimation Method Using Function Point. Journal of Information Science and Engineering, 27(4), 1363-1376.
[13]. Kumari, S. & Pushank, S. (2013). Performance analysis of the software cost Estimation Methods: A Review. International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3(7), pp. 229 - 238.