International Journal of Computer Applications (0975 – 8887)
Volume 108 – No 16, December 2014

Function Point an Effective Planning Tool

Usman Waheed
Assistant Professor
Bahria University (Karachi Campus) Pakistan

Tazeen Muzzamil
Assistant Professor
Bahria University (Karachi Campus) Pakistan

Beenish Tanveer
SZABIST,
Karachi, Pakistan

ABSTRACT

Prerequisite of every estimation technique is effective requirement discovery and planning.

FP is observed to be good source of initial planning. While using Function Point (FP), the whole application is brainstormed. Application is planned with gathering all inputs, outputs, inquiries, external interfaces and files.

Whereas use case point (UCP) are limited to identifying actor, and use cases. UCP is not effective in initial planning. Research presented in this paper takes advantage by combining UCP with FP model in identifying objects like (External Inputs, External Outputs, External Inquiries, External Interface files, Internal Logical files, and others) for each use case. That will help in effective planning of project at initial stages of application. Results show that by combining the FP model and UCP model, component identification enhances by 66 %.

Keywords

FP (Function Points), UCP (use case point)

1. INTRODUCTION

Good planning and estimation is not only dependent on the tools, methodologies and practices; but on the conjoint realization and the positive attitude of both the software developer and the management. When the managers and the developers work together and have an understanding of what has already be done and what needs to be done, this result in successful project planning so that they are more predictable.

Schedule and cost estimation of software projects rely on a prediction of the size of the upcoming system. Unluckily, the software prediction is extremely inaccurate when estimating cost and schedule. Initial estimates typically miss many basic elements of application. Reliable early estimations are difficult to obtain because of the insufficiency of detailed information about the future system at an early stage. However, early estimates are essential when bidding for a contract or defining whether a project is achievable in the terms of a cost-benefit analysis. Research suggests that if early planning and object recognizing is precise there are more chances of improved estimation results.

1.1 Function Point and its Variation

Albrecht [1] in late 1970's was the first to propose function point metric and associated function point analysis method.

It is based on 5 user identifiable logical functions" which are divided into 2 data function types and 3 transactional function types as shown in below table. For a given software application, each of these elements is quantified and weighted as mentioned I table 1, counting its characteristic elements, like file references or logical fields. Numerous function size metric and methods have been proposed since their original publications.

ISO/IEC 14143 Function size measurement (FSM) standard provides the explanation and classification of FSM [2] [3] . Following are the sizing methods that are certified by ISO:, International Function Point Users Group (IFPUG) FPA [5] , MKII FPA [6] , COSMIC FPA [4] and NESMA FPA [7] .

The IFPUG has continuously been improving the initial Albrecht method for function sizing and is the most popular technique of all the sizing methods. Function Point Analysis measures the functionality, based on the "external user view" and "logical internal view" of an application as compared to measuring the "internal technical view". The FPA measures relates directly to the Business Requirements and the Business Data of the software application [10] . In the classic FPA method proposed by Albrecht the complexity of the external user view was somewhat subjective. IFPUG have propagated rules on how to measure it [5] . Another change in the FPA method was related to the Adjusted Function Point Count which is no longer recommended by IFPUG.

Charles Symons found some limitations in the Albrecht's FPA method due to which he published his own FPA method name MK II [11] in 1999. UK Software Metrics Association is now accountable for its continuing developments [6] . The basic concept of Symon's work is based on Albrecht method; by overcoming the limitations of the Albrecht FPA. The foundation of the MK II FPA is based on the fundamental concept of Albrecht FPA; that is the size of the product can be measured by the product of Information Processing Size and Technical Complexity Adjustment (TCA). The criticism on Albrecht's approach was on the TCA that used fourteen general applications characteristic which were hard to differentiate. MK II made modifications to the TCA by adding five characteristics to the list. For Logical transactions Albrecht used the process of involving five component types, external inputs, external outputs, external interfaces, external enquiries and logical internal files. MK II views the system as a collection of logical transactions (United Kingdom Software Metrics Association. "MK II Function Point Analysis: Counting Practices Manual Version 1.3. 1." (1998).

The Full Function Point (FFP) method was proposed in 1997. Many improvements have been proposed by Common Software Measurement International Consortium (COSMIC); which were published in May 2001 as version 2.1 of the COSMIC-FFP Functional Size Measurement Method [4] . COSMIC-FFP method was designed to measure the functional size of real-time software, multi-layered software and business application software (such as telecom, process control and operating system) all on the same measurement scale [4] . The COSMIC-FFP Measurement Process is based on three basic phases. First Setting the Measurement Strategy that is establishing the scope and purpose of the measurement. Secondly mapping the 'Functional User Requirements' (or

'FUR') of the software to be measured to the COSMIC-FFP concepts and in the end measuring the resulting COSMIC-FFP model of the FUR.

1.2 Use Case Point

Since last two decades UML [9] is widely used in industry. Use Case Point (UCP) [12] is one of most popular mechanism used that starts with actor use case diagram.

Use case points can be counted from the use case analysis of the system. The first step is to classify the actors as simple, average or complex.

- Actor type: Simple, that interact through API,
- Actor type: Average, that interacts through text based GUI .
- Actor type: Complex, that interact through now days GUI.

Use case complexity is then defined

- Simple:3 or fewer transactions
- Average: 4 to 7 transactions
- Complex: More than 7 transactions

There are some problems with UCP. Such as it is clear that if use cases does not belong to any transaction then its type is simple whether it is technically complex such as any non functional requirements as mentioned in [13] [14] [16] .

Another problem if any use case contains less transaction, but these transactions are complex in nature, such as General ledger transaction with complex business rules and multiple postings.

It is also observed that in UCP only quantifying use case by transaction. And in business situations transactions are of eight different types [15] Other way of classifying transaction can be in term of complexity of front end, complexity of backend tables, intermediate interfaces, external interfaces and or combination of all mentioned types.

For that purpose research proposes NOT to define use case in term of transaction, but also in term of other development components as supported in FP model.

2. METHODOLOGY FOR USE CASE DRIVEN PLANNING USING FUNCTION POINTS

UCP is weak for planning and identifying requirements of application. Use cases are usually abstract in nature and in naming convention. They can't be good source of planning. Methodology proposes to elaborate each use case with the help of function points, that may help in discovering and planning the project.

2.1 Solution

When new application is developed, the entire objects or components of the project are needed to be known. This means number of function points of the application plus any other function points that need to be developed would be included. In the end, it would have the number of function points for the application to be installed plus any other functions need to be developed.

2.2 Steps of Methodology

Following are the steps of proposed methodology that starts with actor-use case model diagram and ends with detailed discovery of objects with the help of function points.

Step 1: First the system is analyzed and documented in terms of actor use cases diagram.

Step 2: Against each use case, following items are to be determined

a. II (Input Interface)
b. OI (Output Interface)
c. IQI (Inquiry Interface)
d. ILF (Internal Logical File)
e. EI (External Interface)
f. Others

Where

II (Input Interface): represents user data interface or control that is used to enter information into system (files), or from external input device such as barcode reader, thumb impression reader, or from any other device. Formatted or semi-formatted files can be used as input if they are stored partially or completely into system files, otherwise they are external interfaces.

Do not include inputs captured from other systems, portal, website or API as they are included in external interfaces. Do not include input that are used for just searching criteria and are not stored into system, as they are inquiry type.

OI (Output Interface): represents reports, files and messages generated from system, or can be for other applications.

Do not include such outputs that are input of other application, they are external interfaces.

Do not include outputs that are generated because of technology used.

IQI (Inquiry Interface): represents those input interfaces that causes and generates an immediate output. These input interfaces used to search and not updated into files. Outputs generated in response of input inquiry are also called inquiry interface.

Do not include pre designed output reports as they are classified in reports.

ILF (Internal Logical File): represents any persistent storage mechanism, such as flat file, tables of database, XML files, or any other format that is used to store and retrieve data or information.

Do not include files that are not accessible to the user through external input, output, or inquiry types.

EIF (External Interface Files): data or Files passed or shared between applications are known as external interfaces. Data of files enters or leaves the application is known as external interfaces.

Others: anything that is not covered in above five categories can be added in it, it can represent technical/logical code/script used in input, output, external interface and inquiry. Code that is implementation of design pattern, architectural pattern, or algorithm. It can be complex transformation or computation, special code for middleware. Code used for storage, retrieval, sorting and searching of unstructured data or information.

2.3 Templates used for Methodology

Proposed methodology uses actor use case diagram for identification of use cases of the system as in Figure 1. And table 2 shows that list of FP components against each use case.

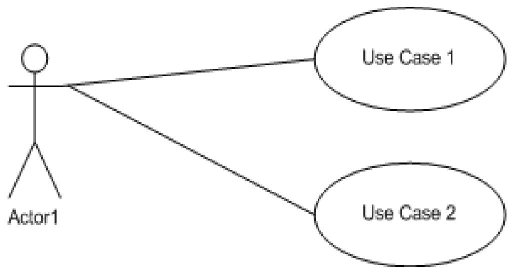

Figure 1: Generalized Use Case Diagram

Table 1: Template for Use case FP table

use case 1	II (Input Interface)
	OI (Output Interface)
	IQI (Inquiry Interface)
	ILF (Internal Logical File)
	EI (External Interface)
	Others
use case 2	II (Input Interface)
	OI (Output Interface)
	IQI (Inquiry Interface)
	ILF (Internal Logical File)
	EI (External Interface)
	Others

3. EVALUATION

To evaluate the methodology, research conducted evaluations from multiple groups. Evaluation was conducted by 5 groups of well recognized Software houses. Details are mentioned below

3.1 Two Day Activity

In this evaluation, five groups of four members each was invited form top ten software houses of Karachi city. Complete two day activity was conducted with two facilitators and one RSD (Requirement Specification Document) that documented the detailed needs of the system to be developed. The system considered was a S*tudent Information System*. Only three modules (Admission, Registration and Fee submission) was taken as a case study. The activity was conducted in two phases.

In first phase (first day) all the groups were given three modules and were told to develop an actor-use case diagram. And then to list and identify, possible database tables, input prototype interfaces, inquiry interfaces and possible reports for the whole system. At the end of first phase, each group listed their development component as mentioned in table 2.

Table 2: First day activity

Groups	Inputs	Outputs	Inquiries	Tables
G1	10	5	2	9
G2	12	5	2	10
G3	9	3	1	7
G4	15	3	2	12
G5	12	4	3	10

In second phase (second day), they were guided to proposed methodology, and suggested to develop step1 and step2 accordingly mentioned in (section 2.2). In this activity all groups brainstormed once again and then redeveloped the previous day list once again. While developing components against each use case, many components repeated. At the end of the day, after eliminating all duplicates, the finalist that was developed is mentioned in table 3.

Table 3: Second day activity

Groups	Inputs	Outputs	Inquiries	Tables	Others
G1	14	6	4	11	4
G2	16	7	4	12	5
G3	12	4	3	11	3
G4	18	6	3	15	3
G5	15	7	5	13	5

After analyzing these two tables, total no of components identified by each is mentioned, and it is concluded that on average 66% new components are discovered, when proposed methodology is used. Results are mentioned in table 4.

The two day activity concluded that, when FP components are used along with use cases, it facilitates in planning. And add further value when FP components are computed against each use case that resulted further 66% improvement.

Table 4: improvements in result

Groups	First day results	Second day results	%age improvement
G1	26	39	67
G2	29	44	66
G3	20	33	61
G4	32	45	71
G5	29	45	64

4. CONCLUSIONS

Initial planning is prerequisite of any estimation, and FP is found to be efficient in identifying development components. Research used FP to strengthen the use case model. And evaluation showed 66% improvement in component identification. Research concludes that for the success of any estimation model is in its initial planning. More your time and effort spent on planning and identifying component is better for the rest for estimation and development of the project.

5. REFERENCES

[1] Albrecht A. (1979). Measuring Application Development Productivity. In *IBM Applications Development Symposium*, Monterey, CA.

[2] ISO/IEC. "" Information Technology - SoftwareMeasurement - Functional Size Measurement" - Part 1:Definition of Concepts." 1998.

[3] ISO/IEC. ""Information Technology - SoftwareMeasurement - Functional Size Measurement - Part 2":Conformity Evaluation of Software Size MeasurementMethods to ISO/IEC 14143-1." 1998, 2002.

[4] ISO/IEC. ""Software Engineeir-- COSMIC: A Functional Size Measurement Method" 2nd Edition." Geneva: ISO, 2011.

[5] ISO/IEC. ""Software and Systems Engineering--Software measurement--IFPUG Functional Size Measurement Method" ISO." Geneva, 2009.

[6] ISO/IEC. ""Software Engineeir-- MK II: A Functional Point Analysis-Counting Practtices Manual" ISO." Geneva, 2002.

[7] ISO/IEC. ""Software Engineeirn--NESMA Functional Size Measurement Method Version 2.1--Definition andCounting Guidelines for theApplication of FunctionPoint Analysis" ISO." Geveva, 2005.

[8] A.J. Albrecht. "“Measuring Application Development Productivity”." Application Development Symposium,. October, 1979,.

[9] d’Souza, Desmond, and Alan Cameron Wills. *Catalysis: Objects, Components, and Frameworks with UML*. Vol. 223. Object Technology Series. Addison-Wesley, 1998.

[10] C. Gencel, O. Demirors, E. Yuceer. "“A Case Study on Using Functional Size Measurement Methods for Real Time Systems”." 15th. InternationalWorkshop on Software Measurement (IWSM). Montreal, Canada, Shaker-Verlag, 12-14 Sept. 2005,. 159-178.

[11] "United Kingdom Software Metrics Association. "MK II Function Point Analysis: Counting Practices Manual Version 1.3. 1." (1998)." n.d.

[12] G. Karner, "Resource Estimation for Objectory Projects,"

[13] Y. Ossia. IBM haifa research lab. *IBM Haifa Research Lab* [Online].2011. Available: https://www.research.ibm.com/haifa/projects/software/nfr/index.html

[14] Z. Jiang, P. Naudé and B. Jiang, "The effects of software size on development effort and software quality," International Journal of Computer and Information Science and Engineering, vol. 1, pp. 230-234, 2007.

[15] S*yed* Irfan Hyder, Usman Waheed, “Transaction Sets from Transaction Pattern”, TECHNOLOGY FORCES (Technol. forces): PAF-KIET Journal of Engineering and Sciences, Volume 01, Number 02, July-December 2007, pp39-42

[16] W. Xia, L. F. Capretz, D. Ho and F. Ahmed, "A new calibration for Function Point complexity weights," *Information and Software Technology,* vol. 50, pp. 670-683, 2008.