International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320-088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 12, December 2017, pg.157 – 163

Comparative Study of Different Project Size Estimation Technique for the Development of Software

Dr. Prakash Kumar

Associate Professor, JRU, Ranchi

mail.drprakash@gmail.com

DOI: 10.47760/ijcsmc.2021.v06i12.001

ABSTRACT

In SDLC (**Software development life cycle model**) there are various phase we use to develop the software in that the one is planning phase in this phase we use some estimation technique for estimate the Size, Cost, Effort etc for the software. The main objective of software engineering discipline is to develop the software in systematic and discipline manner as per user requirement. And also the software should deliver in time and in budget. To acquiring this feature is called planning of the software i.e. how much it take time and cost to complete and effort required form development is depend on nature of the software. The objective of this paper is to find out advantages and shortcoming of different Size estimation technique. In this paper we compared all traditional approach for size estimation technique.

Keywords: Software Engineering, Size Estimation Techniques, LOC, FP and Feature Point.

I. INTRODUCTION

Once a project is found to be feasible, software project managers undertake project planning. Project planning is undertaken and completed even before any development activity starts. Project planning consists of the following essential activities[1]:

- Estimating the following attributes of the project:

Project size: What will be problem complexity in terms of the effort and time required to develop the product?

Cost: How much is it going to cost to develop the project?

Duration: How long is it going to take to complete development?

Effort: How much effort would be required?

The effectiveness of the subsequent planning activities is based on the accuracy of these estimations.

- Scheduling manpower and other resources
- Staff organization and staffing plans
- Risk identification, analysis, and abatement planning
- Miscellaneous plans such as quality assurance plan, configuration management plan, etc.

Different project related estimates done by a project manager have already been discussed. Fig. shows the order in which important project planning activities may be undertaken. From fig it can be easily observed that size estimation is the first activity. It is also the most fundamental parameter based on which all other planning activities are carried out. Other estimations such as estimation of effort, cost, resource, and project duration are also very important components of project planning[2].

II. Metrics for software project size estimation

Accurate estimation of the problem size is fundamental to satisfactory estimation of effort, time duration and cost of a software project. In order to be able to accurately estimate the project size, some important metrics should be defined in terms of which the project size can be expressed. The size of a problem is obviously not the number of bytes that the source code occupies. It is neither the byte size of the executable code. The project size is a measure of the problem complexity in terms of the effort and time required to develop the product[3].

Currently two metrics are popularly being used widely to estimate size: lines of code (LOC) and function point (FP). The usage of each of these metrics in project size estimation has its own advantages and disadvantages.

2.1 Lines of Code (LOC)

Determining the LOC count at the end of a project is a very simple job. However, accurate estimation of the LOC count at the beginning of a project is very difficult. In order to estimate the LOC count at the beginning of a project, project managers usually divide the problem into modules and each module into sub modules and so on, until the sizes of the different leaf-level modules can be approximately predicted. To be able to do this, past experience in developing similar products is helpful. By using the estimation of the lowest level modules, project managers arrive at the total size estimation[4,5].

A) Line Counting Rules

- Comments do not count.
- Blank lines do not count.
- Standard operating system include files do not count.
- Code lines count.
- Job control lines count

- User defined include files count (once).
- SQL statements count.

B). Advantages of LOC

- It is simple to use
- easily can be automated (plenty of tools are available)

C). Disadvantages of LOC

- It is dependent on language and programmer
- penalizes the well designed short programs
- Need a level of detail that may not be available at the early stages of development.
- Users cannot easily understand it.

2.2 Function Points (FP)

- Function point metric was proposed by Albrecht [1983]. This metric overcomes many of the shortcomings of the LOC metric. Since its inception in late 1970s, function point metric has been slowly gaining popularity. One of the important advantages of using the function point metric is that it can be used to easily estimate the size of a software product directly from the problem specification. This is in contrast to the LOC metric, where the size can be accurately determined only after the product has fully been developed[4,5].
- $$UFP = (\text{Number of inputs}) * 4 + (\text{Number of outputs}) * 5 + (\text{Number of inquiries}) * 4 + (\text{Number of files}) * 10 + (\text{Number of interfaces}) * 10$$
- **Number of inputs:** Each data item input by the user is counted. Data inputs should be distinguished from user inquiries. Inquiries are user commands such as print-account-balance. Inquiries are counted separately. It must be noted that individual data items input by the user are not considered in the calculation of the number of inputs, but a group of related inputs are considered as a single input.
- **Number of outputs:** The outputs considered refer to reports printed, screen outputs, error messages produced, etc. While outputting the number of outputs the individual data items within a report are not considered, but a set of related data items is counted as one input[6,7,8].
- **Number of inquiries:** Number of inquiries is the number of distinct interactive queries which can be made by the users. These inquiries are the user commands which require specific action by the system.
- **Number of files:** Each logical file is counted. A logical file means groups of logically related data. Thus, logical files can be data structures or physical files.
- **Number of interfaces:** Here the interfaces considered are the interfaces used to exchange information with other systems. Examples of such interfaces are data files on tapes, disks, communication links with other systems etc.
- Once the unadjusted function point (UFP) is computed, Each of these 14 factors is assigned from 0 (not present or no influence) to 6 (strong influence). The resulting numbers are summed, yielding the total degree of influence (DI). Now, TCF is computed as $(0.65 + 0.01 * DI)$. As DI can vary from 0 to 70, TCF can vary from 0.65 to 1.35. Finally, $FP = UFP * TCF$ [9].

Function Type	Low	Average	High
External Input	x3	x4	x6
External Output	x4	x5	x7
Logical Internal File	x7	x10	x15
External Interface File	x5	x7	x10
External Inquiry	x3	x4	x6

Low, average and high decision can be determined with this table :

	1-5 Data element types	6-19 Data element types	20+ Data element types
0-1 File types referenced	Low	Low	Average
2-3 File types referenced	Low	Average	High
4+ File types referenced	Average	High	High

In order to find adjusted FP, UFP is multiplied by technical complexity factor (TCF) which can be calculated by the formula :

$$TCF = 0.65 + (\text{sum of factors}) / 100$$

There are 14 technical complexity factor. Each complexity factor is rated on the basis of its degree of influence, from no influence to very influential:

1. Data communications
2. Performance
3. Heavily used configuration
4. Transaction rate
5. Online data entry
6. End user efficiency
7. Online update
8. Complex processing
9. Reusability
10. Installation ease
11. Operations ease
12. Multiple sites
13. Facilitate change
14. Distributed functions

$$\text{Then FP} = \text{UFP} \times \text{TCF}[10]$$

Function points are recently used also for real time systems.

A). Advantages of FP

- More accurate than estimated LOC
- Language independent
- The necessary data is available early in a project. We need only a detailed specification.
- It is not restricted to code
- Independent of programming language
- The necessary data is available early in a project
- Deals quantitatively with complexity
- Uses readily countable characteristics of the “information domain: of the problem
- Does not “penalize” inventive implementations that require fewer LOC than others
- Makes it easier to accommodate reuse and the trend toward object-Oriented approaches
- Ignores quality of output

B). Drawbacks of FP

- One-sided counting
- Hard to automate
- Ignores quality of output
- Oriented to traditional data processing applications
- Effort prediction using the unadjusted function count is often no worse than when the TCF is added
- Difficult to compute
- Built based on traditional data processing applications

Shortcomings of Function point (FP) metric

Summary of the comparison of LOC and Function Points

	Lines Of Code	Function Points
Language & technology	Dependent	Independent
Communication with clients	Not familiar with LOC	Somewhat easy to communicate with the client
Tools	There are various software tools for counting LOC	Not any automated tool for FP
Usage time	Better result in the design stage	FP is applicable in early stage of requirements gathering

Feature point metric[12,13]

A major shortcoming of the function point measure is that it does not take into account the algorithmic complexity of software. That is, the function point metric implicitly assumes that the effort required to design and develop any two functionalities of the system is the same. But, we know that this is normally not true, the effort required to develop any two functionalities may vary widely. It only takes the number of functions that the system supports into consideration without distinguishing the difficulty level of developing the various functionalities. Function Points were originally invented to solve the measurement problems of classical Management Information Systems. There are cases that the Function Points' results are not optimal such as: real time software, embedded software, communication software and process control software.

In response to the need for more accurate estimation "Software Productivity Research Inc" in 1986 developed the Feature Points approach that is suitable for the aforementioned systems.

Feature point metric incorporates an extra parameter algorithm complexity. This parameter ensures that the computed size using the feature point metric reflects the fact that the more is the complexity of a function, the greater is the effort required to develop it and therefore its size should be larger compared to simpler functions. The calculation of the Feature Points is similar to the calculation of the Function Points.

Conclusions and Future Work:

Accurate and early estimating require and historical data will identify the capacity of delivery; this factor should be considered in estimation and considering the contingency for the estimation. No estimation method is perfect by itself, if it were; it would be called prediction, not estimation. Using other estimation methods along the chosen estimation method enables the estimator to cross check and will results in an accurate estimate. After the size estimation we start cost estimation and Software size estimation is a crucial input for the cost estimate process. Software engineer or the person who is in charge of the size estimation should select a suitable and accurate size estimation model or method (in order to eliminate the propagation of the error in estimation in the later stages)[14]. There are a variety of size estimation models and methods that each of them has its strengths and weaknesses. It seems to finding right model or method is a difficult activity, however, by examining the available models and methods it is possible to find the accurate and suitable model for estimation. In my future work can improve function point metric or can propose a methodology to estimate the size of the software from design phase of the project mainly Use case model and class diagram. In SDLC model a project manager can gather all refine requirement from customer after several process that means there is minor chance to mistake to gather requirement form customer. After gathering requirement design Use case model and class diagram it will be helpful to estimate the actual size of the project by some experience or expert team member[15].

References

- [1] Kathleen Peters, Berkun, Scott, "*Art of Project Management. Cambridge*", MA:O'Reilly Media. ISBN 0-596-00786-8. Software Project Estimation (White paper), Software Productivity Centre Inc. (SPC) in Vancouver, British Columbia, Canada, 2005.
- [2] Nilesh Chandra Shukla, "*A Tool for Software Project Management for Estimation, Planning & Tracking and Calibration*". IIIT-Allahabad.
- [3] Rajib mall, "*Fundamentals of Software Engineering*", PHI learning pvt.Ltd.2009 edition.
- [4] Nasib Singh Gill, "*Software Engineering, Software Reliability, Testing and Quality Assurance*", Khanna Book Publishing Edition.
- [5] D. R. Jeffery, G. C. Low, and M. Barnes, "*A comparison of function point counting techniques*", IEEE Trans on Soft. Eng., vol. 19, no. 5, 1993, pp. 529-532.
- [6] D. St-Pierre, M Maya, A. Abran, J. Desharnais and P. Bourque, "*Full Function Points: Counting Practice Manual*", Technical Report 1997-04, University of Quebec at Montreal, 1997.
- [7] S.K.Mohanty & A.K.Bishnoi, "*Software Effort Estimation Approaches- A Review*", International Journal of Internet Computing ISSN No: 2231 – 6965, VOL-1, ISS-3 2012.
- [8] Hareton Leung, Zhang Fan, "*Software Cost Estimation*", Article 2001.

- [9] Jyoti G Borade & Vikas R Khalkar, “*Software Project effort and cost Estimation Techniques*”, *International Journal of Advanced Research in Computer Science and Software Engineering* ISSN-2277-128X, Vol. 3, Issue 8, aug 2013.
- [10] Matthias Kerstner, “*Software Test Effort estimation Methods*”, 2 February 2011.
- [11] Ali BouNassif, Luiz Fernando Capretz and Danny Ho, “*Enhancing Use Case Point Estimation Method Using Soft Computing Techniques*”, ISSN-2229-371X.
- [12] Mike Cohn, “*Estimating With Use Case Points*”, Founder-Mountain Goat Software.
- [13] Erik Stensrud, “*Estimating with Enhanced Object Points vs. Function Points*”. Andersen Consulting Drammensveien 165, 0212 Oslo, Norway& University of Oslo, Dept. of Informatics.
- [14] Kurmanadham V.V.G.B. Gollapudi, “*Function Points or Lines of Code?—An Insight*”, Wipro Technologies.
- [15] Hihn, J., Gemoetz, L., Mahmood, M. & Longstreet, D. (1997), “*The Impact of Learning on Function Point Counting Accuracy*”, Presentation on 12* COCOMO Forum, Los Angeles, October 6-10.