Empirical Study of Function Point Analysis during Software Development Phase

Ashish Sharma[1], Yogesh Sharma[2], Radhika Bansal[3], and Sushant Verma[4]

[1]Computer Science Department, Maharaja Agrasen Institute of Technology, India, ashish@mait.ac.in

[2]Computer Science Department, Maharaja Agrasen Institute of Technology, India, yogeshsharma@mait.ac.in

[3]Computer Science Department, Maharaja Agrasen Institute of Technology, India, radhikabansal5898@gmail.com

[4]Computer Science Department, Maharaja Agrasen Institute of Technology, India, sushantv7597@gmail.com

Abstract: During the development of the software, Software Requirement Changes (SRC) often occurs. Software estimation helps in maintaining a balance between the increase and decrease of the price, time, etc. Function Point Analysis (FPA) is a software effort estimation technique used for measuring the size and complexity of software by calculating the functionality from a user's point of view. Function count focuses on what functionality is being delivered. It enables programmers to perform function point counting themselves instead of contacting an expert. An empirical study has been conducted to analyze the capability of the FPA technique. The results of the FPA-SDP (Function Point Analysis for Software Development Phase) model can help software project managers in (i) knowing the inconsistent states of software artifacts (ii) estimating the actual size of a change request with its complexity level for the software development phase.

Keywords: Software Development Phase, Function Point Analysis, Software Requirement Changes, Size Estimation, Line of Code, Effort Estimation

1. INTRODUCTION

The software goes through changes at each stage of the Software Development Life Cycle (SDLC). An effective decision has to be made on whether to accept or reject the changes. Software Estimation is an important tool for the success of the project with proper planning. It helps in deciding the decisions to be taken and achieve accuracy and quality to a great extent. One of the techniques of software estimation is Function Point Analysis (FPA). FPA is used to determine the size at an early stage of project development. FPA indicates the function point count that can be expected after the completion of the design phase. For each step of FPA, there is a mixing of scales and assignments of weights. Function Point Counts at the end of requirements, analysis, design, code, testing and implementation can be compared. The function point count at the end of requirements and/or designs can be compared to function points delivered.

Software effort estimation is an important approach that helps the software project manager in making an effective change approval decision. Software efforts estimation is the process of predicting the most accurate amount of effort required to develop or maintain software based on a large number of changing variables [1]. The estimation of required efforts for requirement changes can be done in many ways, some of them are:

DOI: 10.5281/zenodo.3715382

Expert Judgment, Estimation by Analogy, Impact Analysis, Source Lines of Code (SLOC) and Function Point Analysis (FPA). It has been continuous research because there are still many views and debates in getting an accurate effort estimation result. All techniques showed that they can be useful to estimate the effort and cost of software, but not any of them every time provides an accurate estimation result. [2]

Many software projects fail due to inaccurate software estimation and misunderstanding or incompleteness of the requirements. This encouraged researchers to conduct a study on software estimation to come up with better software matrices. As software estimation becomes critical to reduce possibilities of project failures, evaluation in the early phases of the software life cycle became vital. The significance of the early estimation reveals when it is required to bid on the project or commit to a contract between the consumer and the developer. The early software estimation is conducted at a point when the information about the problem is not yet revealed. This is known as the size estimation paradox [3].

The current challenge of relying solely on the FPA method is that mostly this technique [4] is used for the software maintenance phase, where software artifacts are consistent. On the other hand, in the software development phase, software artifacts are in inconsistent states. So it becomes a challenging task for software project managers to accept or reject change requests during the software development phase. [5]

The rest of this paper is represented as follows: Section 2 represents the literature review, Section 3 represents the terminology used, Section 4 represents a case study, and Section 5 represents a conclusion.

2. LITERATURE REVIEW

The principle of Albrecht's FPA is that a system is decomposed into five functional units.

- Data Function Types – Internal Logical Files, External Interface Files
- Transactional Function Types – External Input, External Output, External Inquiry [6]

Each function point is ranked according to the complexity (low, average, high). There exist pre-defined weights for each function point in each category. The assigning of weights or rank function points depends on the organization (which evaluate on basis of past projects). Unadjusted Function Point (UFP) is calculated by multiplying each function point by its corresponding weight factor. Final function point is calculated as follows:

$$\text{Final FP} = \text{UFP} * \text{CAF}$$
$$\text{CAF} = 0.65 + 0.01 * \Sigma F$$

Where CAF stands for Complexity Adjustment Factor which is calculated using 14 aspects of processing complexity. The 14 questions are answered on a scale of 0 to 5.

3. TERMINOLOGY USED

3.1 Change Impact Analysis

Change Impact Analysis (CIA) is the method of finding possible consequences of a change, or "estimating what needs to be modified to accomplish a change" [7]. There are two types of IA techniques: (i) Static Impact Analysis (SIA) and (ii) Dynamic Impact Analysis (DIA). The SIA technique considers static information from software artifacts to produce a set of possible impact classes. On the other hand, the DIA technique considers dynamic information created by implementing the code to generate a set of potential impact classes. The studies show that the integration of the SIA technique with the DIA technique as a new approach. Also, these studies consider the fully and partially developed classes during the software development phase [8].

DOI: 10.5281/zenodo.3715382

3.2 Line of Code

Lines of code (LOC) or source lines of code (SLOC), is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code. LOC is the metric that measures the size of the project by counting the number of instructions in the developed program. While counting the number of source instructions, lines used for commenting the code and the header lines are ignored. To estimate the LOC count at the beginning of a project, one would have to make a systematic guess. So sometimes project divided into modules and sub-modules until sizes of different modules can be approximately predicted. Lines of code (LOC) only measures the volume of code, one can only use it to compare or estimate projects or programs that use the same language and is coded using the same coding standards. To change one is to change the volume of code. A better method to compare without regard to direct volume is to measure the complexity of the software. Lines of code could be defined either [9]: Physical SLOC, Logical SLOC. [10]

3.3 Function Point Analysis

Function Point Analysis (FPA) method was developed by Allan Albrecht in 1979. It is a technique of counting the size and complexity of a software system in terms of the functions that the system provides to its end users. The main goals of the FPA method are (i) independent of development technology, (ii) simple to apply, (iii) can be estimated from requirement specifications and (iv) meaning full to end-users.
Additionally, a systematic literature review was performed on EE by in which they specified that FPA is one of the most solid and reliable estimation techniques.

In the FPA technique, Function Points (FPs) of software are calculated by adding Unadjusted Function Points (UFPs) with Value Adjustment Factor (VAF). The procedure of calculating UFPs and VAF is given in the International Function Point User Group (IFPUG) manual.

$$FPs = UFP * VAF \quad (1)$$

Whereas,
FP stands for Function Points
UFP stands for Unadjusted Function Points.
UFPs is the sum of all functions i.e. External Interface (EI's),
External Output (EO's), External Queries (EQ's), Internal Logical Files (ILF's) and External Interface Files (EIF's) with its level of complexity (low, average and high).
VAF stands for Value Adjustment Factor
Value Adjustment Factor (VAF) can be calculated from fourteen
General System Characteristics (GSC) as shown in Equation

$$VAF = 0.65 + [(\Sigma ni=1Ci) *0.1] \quad (2)$$

Where:
i = GSC from 1 to 14.
Ci = degree of influence for each General System Characteristic.
Σ = summation of 14 GSC.
So, after getting the value of VAF from Equation (2) the final value of FPs can be calculated from Equation 1.

DOI: 10.5281/zenodo.3715382

3.4 Effort Estimation

Effort Estimation (EE) is the method of predicting how much work and how many hours of work is required to develop software. Normally it describes in man-days or man-hours unit [11]. According to Idri, et al. EE is one of the most interesting tasks for software project managers. Several EE models have been developed. Some of the most commonly used EE models are Expert Judgement; Estimation by Analogy; and Regression Analysis. However, before estimating the amount of effort for a change request, it is important to estimate its accurate size. It is still a challenging task for software project managers to estimate the accurate size of a change request. For this purpose, the two most common methods which are used are (i) SLOC and (ii) FPA. [5]

4. CASE STUDY

A case study on the Library system is being carried out. The objective of the problem was to find out that can function point analysis be used as the method for calculating the software size in terms of LOCs and possible errors. The raw requirements considered were:
A register of accredited users was maintained by an administrator, who could add, remove and change user privileges. Users got access to the system via the typical login/logout mechanism. A logged-in user could search for the books in the catalogs based on author names or titles of the books. A librarian could issue and return the books after verifying the member and book details.

The specifications of the library system were considered and measured. The specification of the system was given to students using an informal text. To support the computation of Function Points, the original specifications were translated into a data flow diagram. To remain as independent as possible from the UML-based techniques, the measurement was based on the data-flow diagrams.

Function Type	Low	Average	High	Total
EI	8*3	0*4	0*6	24
EO	3*4	0*5	0*7	12
EQ	2*3	0*4	0*6	6
ILF	3*5	0*7	0*15	15
ELF	1*7	0*10	0*10	7

FP = 64 * [0.65 + (0.01*48)]
FP = 72
Assumptions and Results - Past date indicates that one FP translates into 60 times of code (if an OOP language is to be used)
LOCs = 60 * 72 = 4320 (approximately)
The past project has found an average of 3 errors per function point during analysis and design reviews and 4 errors per function point during the unit and integration testing.
Thus, a possible number of errors in analysis and design reviews should be 3*72 i.e. 216. At the time of testing, a possible number of errors should be 4*72 i.e. 288. Thus the total possible number of errors should be 504.

Verification of Results - After implementation, it was found that lines of code are 4870, which is more than calculated LOCs by a value of 550. Errors found at the time of analysis and design reviews are 196 and errors

DOI: 10.5281/zenodo.3715382

found at the time of testing are 325. Thus total errors found are 521 which is more than calculated by a value of 17.

5. CONCLUSION

This paper represents an empirical study of the FPA technique to calculate software estimation. The above analysis and observations show that function points are an important tool to measure the probable errors at all the development stages. However, errors found maybe more if the development process is not matured, thus an indication to improve the process. Advantages of FPA: useful even for those users without external expertise, the size of software delivered is measured independent of language and technology tools. To verify its usefulness, it is no doubt that more experimental data from a wider range of application domains and environments should be collected to deepen the investigation into the trade-off between generic versus specific. While it is challenging to use the FPA approach for requirement changes during the software development phase. On the other hand, the FPA approach can be applied if it is integrated with change impact analysis. So as future work we recommend that the integration of the FPA approach with change impact analysis can be useful for effort estimation during the software development phase. [2]

References

[1] N. Kama and M. Halimi, "Extending Change Impact Analysis Approach for Change Effort Estimation in the Software Development Phase," *Advanced Informatics School, Universiti Teknologi Malaysia,* pp. 1-7, 2013.

[2] J. Shah and N. Kama, "Issues of Using Function Point Analysis Method for Requirement Changes During Software Development Phase," *Advanced Informatics School, Universiti Teknologi Malaysia,* pp. 156-163, 2018.

[3] A. B. Nassif, D. Ho and L. F. Capretz, "Software Estimation in the Early Stages of the Software Life Cycle," *International Conference on Emerging Trends in Computer Science, Communications and Information Technology,* pp. 1-13, 2010.

[4] H. Azath and R. Wahidabanu , "Efficient effort estimation system viz. function points and quality assurance coverage," *IET Software,* vol. 6, no. 4, pp. 335-341, 2016.

[5] J. Shah, N. Kama and S. A. Ismail, "An Empirical Study with Function Point Analysis for Software Development Phase Method," *ICSIE'18,* pp. 7-11, 2018.

[6] C. R. Symons, "Function Point Analysis: Difficulties and Improvements," *IEEE Transactions on Software Engineering,* vol. 14, no. 1, pp. 2-11, 1998.

[7] D. Kchaou and N. Bouassida, "UML models change impact analysis using a text similarity technique," *IET Digital Library,* vol. 11, no. 1, pp. 27-37, 2017.

[8] S. Basri and R. Ibrahim, "A Novel Effort Estimation Approach for Requirement Changes during Software Development Phase," in *International Journal of Software Engineering and Its Applications*, 2015.

[9] V. Nguyen, S. Deeds, T. Tan and B. Boehm11, "A SLOC Counting Standard," *Center for Systems and Software Engineering University of Southern California,* vol. 1, no. 1, pp. 1-15, 2007.

[10] S. Bhatia and D. J. Malhotra, "A Survey on Impact of Lines of Code On Software Complexity," *IEEE International Conference on Advances in Engineering & Technology Research,* pp. 1-4, 2014.

[11] M. Shahid and S. Ibrahim, "Change impact analysis with a software traceability approach to support software maintenance," *IEEE,* vol. 1, no. 1, pp. 1-6, 2016.

DOI: 10.5281/zenodo.3715382