*International Journal of Engineering & Technology*, 7 (4.28) (2018) 151-154

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# An Empirical Study with Function Point Analysis for Requirement Changes During Software Development Phase

**Jalal Shah*[1], Nazri Kama[2], Amelia Zahari[3]**

*[1,2,3]UTM AIS*

**Corresponding author E-mail: engrjalalshah7@gmail.com*

**Abstract**

Software Requirement Changes (SRC) take place at any stage during the development of a software system. Allowing for too many changes may increase in price and period of the development of a software system. On the other hand, denying changes may increase consumer unhappiness. Software Change Effort Estimation (SCEE) is one of these techniques that can help software development team in accepting or rejecting a change. At present many SCEE techniques have been introduced and Function Point Analysis (FPA) is one of them. FPA commonly used for SCEE during the early phases of software development cycle. Our previous works have shown that it is a challenging task to implement FPA technique in Software Development Phase (SDP) due to the inconsistent states of software artifacts such as (1) some are fully developed, (2) some are partially and (3) some are not developed yet. Hence, an empirical study is conducted on FPA to analyze the capability of the FPA technique to support SCEE in the context of software development phase. From the study, we found that the FPA technique is not able to present the: (1) current state of software artifacts; and (2) impact of SRC on software artifacts. As a result, we recommended in our future works that the integration of FPA with Impact Analysis (IA) technique that can overcome the limitations and potentially giving higher accuracy of SCEE results.

***Keywords***: *Software Change Effort Estimation; Function Point Analysis; Software Requirement Changes; Software Development Phase*.

## 1. Introduction

Software Requirement Changes (SRCs) may occur at any time and phase of the Software Development Life Cycle (SDLC)[1] Accommodating many SRCs may increase the development period and budget of the software. While rebuffing SRCs may raise client disappointment. Hence, it is very crucial for a Software Development Team (SDT) to change the requirements and take the best decisions for the success of software projects. SCEE technique is one of the inputs that can assist and support SDT in taking best decisions during software development phase [2, 3].

Software Change Effort Estimation is the process of predicting how much work and how many hours of work are required for a particular change request implementation [4, 5]. The purpose of the software effort prediction is to evaluate the volume of effort and time required in implementing the particular SRC [4-6].

There are two types of SCEE techniques that have been widely used are: (1) Algorithmic and (2) Non-algorithmic techniques. Algorithmic techniques that are usually used for SCEE are: Constructive Cost Method (COCOMO II)[7], FPA [8] and Use-Case Points (UCP) [9]. Alternatively, previous studies highlighted that Non-Algorithmic techniques such as Expert Judgement (EJ) [10], Analogy Based Estimation (ABE) [11] and Delphi [12]. Although several extensions have been developed based on the current effort estimation techniques [13-16]. But, these extensions are still lacking in considering the SCEE during SDP.

According to Sufyan, et al. [17], in software development phase, requirements might change due to the dynamic nature of the software projects. These changes will give an impact to software development team in controlling the software effort estimation. Furthermore, software development phase includes an important factor i.e. inconsistent states; that need to be considered in estimating the required effort. The inconsistent states are: (1) the existence of partially developed artifacts; (2) the existence of artifacts that have been developed conceptually but not practically been implemented; and (3) the existence of fully developed artifacts. On the other hand, the failure of these considerations will lead to inaccuracy of estimation and results in project delay or customer dissatisfaction. Therefore, SCEE is a challenging task for SPT for SRCs during SDP [18].

In this study, we have used FPA technique in an empirical study for SRCs during SDP. The results of the empirical study are highlighted the key problems which are faced by using FPA technique for SRCs during SDP.

This study is structured as: Section (2) Literature Review, section (3) Methodology, section (4) Discussion, and section (5) Conclusion and Future Works.

## 2. Literature Review

There are two most related keywords involve in this research are: Software Change Effort Estimation and Function Point Analysis.

### 2.1 Software Change Effort Estimation

SCEE is a technique that forecasts the amount of work and the number of hours which are required for the implementation of a SRC. The results of SCEE can be used in project plans, budgets, iteration plans, investment analyses, bidding rounds and risk management [16]. Several types of SCEE techniques are discussed

in the literature. While, some of them are very famous and used widely such as; (1) EJ [9]; (2) ABE[16]; (3) FPA[8]; (4) Regression Analysis [7, 19]; and (5) Model Based [20].

Expert Judgement is a very famous technique mostly used for SCEE till today. Most of the SPT selects EJ due to its flexibilty and less complication as compare to other SCEE techniques. Currently, there is not a single SCEE technique that clamis for hundrade percent precise results [13].

Whereas, ABE technique uses the data or information which is experienced from related projects. The knowledge behind ABE attainment for SCEE is based on the previous information of inter-related projects. Probably, due to simplicity and flexibility the ABE is frequently used as hybrid model for combination with other techniques to increase its accuracy such as, Particle Swarm Optimization (PSO), Grey Relational Analysis (GRA), regression and rough set theory [11, 21].

Source Lines of Code (SLOC) measures the size of software. In SLOC technique number of lines of cod are countined for estimating required effort. However, SCEE can be possible with SLOC once the code developed for said SRC. Some studies [3, 22] specify that the moment of SLOC will be decreced for large size development projects and also providing different SCEE values for different programming languages. Therefore, it is a challenging task for SPT to get accurate SCEE results with SLOC [9]. Later on Allan Albrecht introduced FPA technique and tried to solve SCEE problems that were faced during SLOC tecchnique [23, 24].

Normally, Software size can be measured with Source Lines of Code (SLOC), Function Point Analysis (FPA) and Use Case Point (UCP), [15]. However, in this study we will try to identify the problems of FPA technique when it is used for SRCs during SDP. Therefore, a brief description of FPA is given in section 2.2.

### 2.2 Function Point Analysis

Function Point Analysis (FPA) method is developed by Allan Albrecht in 1979. It measures the size and complexity of a software by calculating its functionality [23, 25]. The main goals of FPA method are: (1) independent of development technology, (2) simple to apply, (3) can be estimated from requirements specifications and (4) meaning full to end users [8]. Furthermore, a systematic literature review is conducted on software effort estimation by [25] in which they stated that FPA is one of the most useable and reliable estimation technique.

In FPA technique, Function Points (FPs) of a software are calculated by adding Unadjusted Function Points (UFP) with Value Adjustment Factor VAF [26] as shown in Equation 1.

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

Where
FP stands for Function Points
UFP stands for Unadjusted Function Point
VAF stands for Value Adjustment Factor

Value Adjustment Factor (VAF) can be calculated from fourteen General System Characteristics (GSC) [26] as shown in Equation 2.

$$VAF = 0.65 + [(\sum^{n}_{i=1} Ci) * 0.1] \quad (2)$$

Where:
i = GSC from 1 to 14.
Ci = degree of influence for each General System Characteristic.
$\sum$ = summation of 14 GSC.
So after getting the value of VAF from Equation 2 the final value of FPs can be calculated [26].

## 3. Methodology

This section describes the method that has been adopted for the calculation of SCEE from the selected case study. During this process three key elements are considered for the assessment of the case study. These elements are; (I) Case Selection (II) Change Requests and (III) Results and Findings.

### 3.1 Case Selection

Course Registration System (CRS) software is a development project which selected as a case study from postgraduate students of software engineering at Advanced Informatics School (AIS), at Universiti Teknologi Malaysia (UTM). During this study five cases are selected from different SDLC stages during SDP as shown in the Table 1.

**Table 1:** Case Studies

| Cases | Stages | States of Software artifacts |
|---|---|---|
| C1 | Analysis | Software Requirements Specification is completed |
| C2 | Design | Software design is completed. |
| C3 | Coding | Some classes are partially developed. |
| C4 | Testing | All classes are developed |
| C5 | Deployment | All classes are fully developed. |

### 3.2 Change Requests

We have selected fifteen change requests (CR) in five cases with three Change Request Types (CRTs) i.e. Addition, Deletion and Deletion in SDP as shown in Table 2.

**Table 2:** Change Requests

| CRTs | Case1 | Case2 | Case3 | Case4 | Case5 |
|---|---|---|---|---|---|
| CRT1-Addition | CR1 | CR4 | CR7 | CR10 | CR13 |
| CRT2-Deletion | CR2 | CR5 | CR8 | CR11 | CR14 |
| CRT3-Modification | CR3 | CR6 | CR9 | CR12 | CR15 |

## 4. Results and Findings

We have followed the rules of IFPUG manual [26] for calculating function points. As we are using three Case types i.e. addition, deletion and modification, for fifteen change requests. The values of UFP for the change requests is shown in Table 3.

**Table 3:** Value of Unadjusted Function Points

| Function Types | Function Complexity | Complexity Total | Function Type Total |
|---|---|---|---|
| ILF | 1 Low * 7<br>1Average *10<br>0 High *15 | = 7<br>= 10<br>= 0 | 17 |
| EIF | 1 Low * 5<br>1 Average *7<br>0 High *10 | = 5<br>= 7<br>= 0 | 12 |
| EI | 1 Low *3<br>0 Average *4<br>0 High *6 | = 6<br>= 0<br>= 0 | 6 |
| EO | 0 Low *4<br>2 Average *5<br>0 High *7 | = 0<br>= 10<br>= 0 | 10 |
| EQ | 1 Low *3<br>1 Average *4<br>0 High *6 | = 3<br>= 4<br>= 0 | 7 |
| UFPs = | | | 17+12+6+10+4=52 |

After calculating UFPs the next step is the calculation of VAF. VAF is calculated from the fourteen GSCs with its degree of interference by using the rules of IFPUG [26] as shown in Table 4.

**Table 4: General System Characteristics**

| General System Characteristics | Degree-of Inference | General System Characteristics | Degree-of Inference |
|---|---|---|---|
| 1.Data Communications | 3 | 8.Online Update | 2 |
| 2.Distributed Data Processing | 2 | 9.Complex Processing | 0 |
| 3.Performance | 3 | 10.Reusability | 2 |
| 4.Heavily Used Configuration | 2 | 11.Installation Ease | 3 |
| 5.Transaction Rate | 2 | 12.Operational Ease | 2 |
| 6.Online Data Entry | 2 | 13.Multiple Sites | 3 |
| 7.End-User Efficiency | 3 | 14.Facilitate Change | 3 |
| Total Degree of Influence (TDI) =32 | | | |

VAF = (TDI * 0.01) + 0.65
VAF= (32* 0.01) + 0.65 =0.97
FP= UFP* VAF
FPs= 52 *0.97 = 50.44

Whereas Development Projects Function Point shown in Equation (3).

$$DPF = (FPs + CEP) * VAF \quad (3)$$

Where:
DFP is the development project function point count
UFP is the unadjusted function point count
CFP is the function points added by the conversion unadjusted function point count

DPF= (49+3) * 0.98 = 51.94 FPs

## 5. Discussion

To review the results of analysis of the empirical study, we have identified the limitations in FPA technique which are: (1) FPA technique cannot trace a requirement change in software artifacts (2) FPA cannot predict the impact of a requirement change on software artifacts.

### 5.1 Function Point Analysis Technique cannot trace a requirement change in software artifacts

In software development phase software artifacts are in inconsistent states. Before, estimating the accurate amount of required effort for implementing a change request it is necessary to know the actual states of the software artifacts. Requirements traceability can help SDT in knowing the actual states of software artifacts. Whereas, FPA technique is not using any proper method for requirements traceability. Therefore, during the implementation of SRCs in software artifacts we faced the problem of requirements traceability

### 5.2 Function Point Analysis Technique cannot predict the impact of a requirement change on software artifacts

However, before accepting or rejecting a change request, it is necessary for SDT to know the impact of the particular SRC on software artifacts. While, FPA technique cannot predict the impact of SRC on software artifacts. Therefore, it becomes critical for SPT to accept or reject a change request while using FPA technique for requirements change during software development phase.

## 6. Conclusion

This paper presents an empirical study on the capability analysis of FPA technique to calculate the SCEE during SDP. Usually, FPA technique used for early phase SCEE when the requirements are predefined. The novelty of this study is that we have implemented the FPA method for SRCs during SDP in the existence of inconsistent states of software artifacts. We have selected a small case study namely Course Registration System (CRS) to analyze the capability of estimation performed by the FPA technique. Our results have shown that there are some challenges facing by the FPA technique which are: (1) Tracing of requirement changes in software artifacts and (2) Impact of requirement changes on software artifacts. Therefore for future work, we intend to integrate the FPA technique with software change impact analysis technique. The selected software change impact analysis technique should be able to consider the inconsistent states of software artifacts in its implementation.

## Acknowledgements:

The study is financially supported by Contract Research Grant (Vote No: 4C124) and Research University Grant (Vote No: 16H68) under Universiti Teknologi Malaysia.

## References:

[1] J. Shah and N. Kama, "Extending Function Point Analysis Effort Estimation Method for Software Development Phase," in *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, 2018, pp. 77-81.
[2] B. Sufyan, K. Nazri, H. Faizura, and A. I. Saiful, "Predicting effort for requirement changes during software development," presented at the Proceedings of the Seventh Symposium on Information and Communication Technology, Ho Chi Minh City, Viet Nam, 2016.
[3] J. Shah and N. Kama, "Issues of Using Function Point Analysis Method for Requirement Changes During Software Development Phase.," presented at the Asia Pacific Requirements Engeneering Conference, Melaka Malaysia, 2018.
[4] M. H. Asl and N. Kama, "A Change Impact Size Estimation Approach during the Software Development," in *Software Engineering Conference (ASWEC), 2013 22nd Australian*, 2013, pp. 68-77.
[5] C. Bee Bee, "Rework Requirement Changes in Software Maintenance," in *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, 2010, pp. 252-258.
[6] B. B. Chua and J. Verner, "Examining requirements change rework effort: A study," *arXiv preprint arXiv:1007.5126*, 2010.
[7] A. Hira, S. Sharma, and B. Boehm, "Calibrating COCOMO® II for projects with high personnel turnover," presented at the Proceedings of the International Conference on Software and Systems Process, Austin, Texas, 2016.
[8] A. Hira and B. Boehm, "Function Point Analysis for Software Maintenance," presented at the Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Ciudad Real, Spain, 2016.
[9] K. Usharani, V. V. Ananth, and D. Velmurugan, "A survey on software effort estimation," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 505-509.
[10] M. Jorgensen, "Practical guidelines for expert-judgment-based software effort estimation," *IEEE software*, vol. 22, pp. 57-63, 2005.
[11] A. Idri, F. a. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Information and Software Technology*, vol. 58, pp. 206-230, 2// 2015.
[12] R. Britto, V. Freitas, E. Mendes, and M. Usman, "Effort Estimation in Global Software Development: A Systematic Literature Review," in *2014 IEEE 9th International Conference on Global Software Engineering*, 2014, pp. 135-144.

[13] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," *Journal of Systems and Software*, vol. 118, pp. 151-175, 8// 2016.
[14] M. d. F. Junior, M. Fantinato, and V. Sun, "Improvements to the Function Point Analysis Method: A Systematic Literature Review," *IEEE Transactions on Engineering Management*, vol. 62, pp. 495-506, 2015.
[15] D. Kchaou, N. Bouassida, and H. Ben-Abdallah, "Change effort estimation based on UML diagrams application in UCP and COCOMOII," in *2015 10th International Joint Conference on Software Technologies (ICSOFT)*, 2015, pp. 1-8.
[16] B. Chinthanet, P. Phannachitta, Y. Kamei, P. Leelaprute, A. Rungsawang, N. Ubayashi, *et al.*, "A review and comparison of methods for determining the best analogies in analogy-based software effort estimation," presented at the Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, 2016.
[17] B. Sufyan, K. Nazri, A. Saiful, and H. Faizura, "Using static and dynamic impact analysis for effort estimation," *IET Software*, vol. 10, pp. 89-95, 2016.
[18] S. Basri, N. Kama, and R. Ibrahim, "COCHCOMO: An extension of COCOMO II for Estimating Effort for Requirement Changes during Software Development Phase," 2016.
[19] C. A. L. Garcia and C. M. Hirata, "Integrating functional metrics, COCOMO II and earned value analysis for software projects using PMBoK," presented at the Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil, 2008.
[20] I. Attarzadeh, A. Mehranzadeh, and A. Barati, "Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference on*, 2012, pp. 167-172.
[21] V. K. Bardsiri, D. N. A. Jawawi, A. K. Bardsiri, and E. Khatibi, "LMES: A localized multi-estimator model to estimate software development effort," *Engineering Applications of Artificial Intelligence*, 2013.
[22] B. W. Boehm, "Software cost estimation meets software diversity," presented at the Proceedings of the 39th International Conference on Software Engineering Companion, Buenos Aires, Argentina, 2017.
[23] A. J. Albrecht, "AD/M productivity measurement and estimate validation," *IBM Corporate Information Systems, IBM Corp., Purchase, NY*, 1984.
[24] M. Saroha and S. Sahu, "Tools & methods for software effort estimation using use case points model — A review," in *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, 2015, pp. 874-879.
[25] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Web Effort Estimation: Function Point Analysis vs. COSMIC," *Information and Software Technology*, vol. 72, pp. 90-109, 4// 2016.
[26] D. Longstreet, "Function points analysis training course," *SoftwareMetrics. com, October*, 2004.