ESTIMATING CHANGE EFFORT USING A COMBINATION OF CHANGE IMPACT ANALYSIS TECHNIQUE WITH FUNCTION POINT ANALYSIS

Jalal Shah
Universiti Teknologi Malaysia
54100, Jalan Semarak, Kuala Lumpur
+6017-3278645
engrjalalshah@yahoo.com

Nazri Kama
Universiti Teknologi Malaysia
54100, Jalan Semarak, Kuala Lumpur
+6013-3945451
mdnazri@utm.my

Nur Azaliah A Bakar
Universiti Teknologi Malaysia
54100, Jalan Semarak, Kuala Lumpur
+6019-6499394
azaliah@utm.my

ABSTRACT

Software effort estimation is one of the methods that can help software project manager in making decision whether to accept or reject requirement changes. Many methods have been developed and Function Point Analysis is one of the methods that is used for software maintenance phase. Looking from software development phase, FPA method faces a challenge on performing estimation when the non-developed software artifacts exists (some of the classes are fully developed; partially developed; and not developed yet). This research aims to develop a method that improves the estimation accuracy through combination of Function Point Analysis method with Change Impact Analysis technique. An evaluation was conducted using two selected case studies where a significant accuracy achievement is achieved.

CCS Concepts

• **Software and its engineering →Software design techniques**.

Keywords

Software Development Effort Estimation; Effort Estimation; Function Point Analysis; Change Impact Analysis; Software Development Phase.

1. INTRODUCTION

Change requests are occurring in all the stages of Software Development Life Cycle (SDLC). These changes are the main causes for a software project failure [1]. An effective change acceptance decision can help software project managers in accepting or rejecting changes [2] . Software Development Effort Estimation (SDEE) is one of the methods that may help the software project managers in accepting or rejecting change requests with a concrete reason. SDEE method predicts the amount of work that is required to implement a change request in Person per Month (PM) unit. SDEE can be performed in many ways such as: Expert Judgement [3], Estimation by Analogy [4], Function Point Analysis (FPA) [5], Source Lines of Code (SLOC) [2] and Regression Analysis [6]. However, this study only focused on FPA method.

ICSIE 2019, April 9-12, 2019, Cairo, Egypt
© 2019 Association for Computing Machinery
ACM ISBN 978-1-4503-6105-7/19/04…$15.00
https://doi.org/10.1145/3328833.3328836

Function Point Analysis is one of the most common effort estimation methods that is used for measuring the size and complexity of an SRC by calculating the number of function points. Commonly, FPA method is used for software maintenance phase where all the software artifacts are in consistent state or assumed that all the classes are fully developed. However, using FPA in Software Development Phase (SDP) is a challenging task. Since in SDP software artifacts are in inconsistent state as some of the classes are fully developed, some are partially developed and some of the classes are not developed yet.

Alternatively, software Change Impact Analysis (CIA) is one of the techniques that can help software project managers in understanding the actual status of software artifacts and moreover, it helps software project managers in knowing the consequences of an SRC on software artifacts.

Many researchers agreed that the combination of FPA with CIA can help software project managers for an accurate effort estimation during software development phase. Therefore, this research combines FPA method with CIA technique to propose a new software effort estimation method. The proposed method predicts the amount of required effort for a SRC in SDP.

This paper is structured as follows: Section (2) presents related work, section (3) describes proposed solution, section (4) presents evaluation process and section (5) presents conclusion and future work.

2. RELATED WORK

The four most related keywords involved in this research are Software Effort Estimation, Software Development Effort Estimation, Change Impact Analysis, Software Development Phase, and Function Point Analysis.

2.1 Software Development Effort Estimation

Software Development Effort Estimation (SDEE) is a process that predicts the amount of work and hours of work which are required to implement a software requirement change. Effort estimation can be done in many ways such as: Expert Judgement [7], Estimation by Analogy [4], Regression Analysis [8], Ontology Based Effort Estimation [9], COCOMO II, Use case Point [10], Source Lines of Code [11] and Function Point Analysis[11, 12].

These methods are divided into two categories: (1) Algorithmic and (2) Non-Algorithmic methods [6]. The Algorithmic methods are based on fixed and predefined statistical and mathematical equations[5]. While, the Non-Algorithmic methods are based on learning, understanding and analyzing previous software development projects and may include past personal experiences.

However, this study has focused on Function Point Analysis (FPA) method which is one of the most common algorithmic methods that is used for software size estimation by calculating the number of function points that the system provides to its end users. FPA method is developed by Allan Albrecht in 1979 and its fundamental objectives are: (1) independent of development technology, (2) simple to apply, (3) can be estimated from requirement specifications and (4) meaning full to end users [13]. A systematic literature review is conducted on software effort estimation by[14] and they stated that FPA is one of the most concrete and consistent estimation technique [12, 15].

In FPA method, user functional requirements are recorded into five types: (1) Internal Logical Files (ILF), (2) External Interface Files (EIF), (3) External Inputs (EI), (4) External Outputs (EO) and (5) External Inquiries (EQ) [16]. A user manual is introduced by International Function Point User Group (IFPUG) in 1994 [17] for the first time to calculate the number of Function Points (FPs) and later it is modified several times with the recent version as *4.3.1* in 2010 [18]. There are several steps introduced in IFPUG manual to calculate the number of FPs [16, 17]. Step 1: to calculate total number of Unadjusted Function Points (UFPs) by adding ILF, EIF, EQ, EO and EI with their respective complexity (low, average or high) as shown in Equation 1[17].

$$UFPs = ILF + EIF + EI + EO + EQ \tag{1}$$

Whereas,

UFPs: Unadjusted Function Points

ILF: Internal Logical Files

EIF: External Interface Files

EI: External Inputs

EO: External Outputs

EQ: External Inquiries

Step 2: to calculate Technical Complexity Factor (TCF) which is calculated by estimating the Degree of Influence (DI) for 14 elements of General Application Characteristics (GSC). A scale is used for DI; from zero (0) (no influence), to five (5) strongly influence. TCF can be calculated from Equation (2).

$$TCF = (TDI \times 0.01) + 0.65 \tag{2}$$

Whereas,

TCF: Stands for Technical Complexity Factor

TDI: Total Degree of Influence

Step 3: to calculate FPs which is the multiplication result of UFPs and TCF as shown in Equation (3) [16, 17].

$$FPs = UFPs \times TCF \tag{3}$$

Whereas,

FPs: Function Points

UFPs: Unadjusted Function Points

TCF: Stands for Technical Complexity Factor

Step 4: to calculate effort which is equal to work productivity multiplied by function points. Generally, the productivity is eight (8) hours worked per day as shown in Equation (4) [5].

$$SRCDE = P \times FPs \tag{4}$$

Whereas,

SRCDE:	Stands for Software Requirement Change Development Effort
P:	Stands for Productivity i.e. Eight (8) hours for One (1) Function Point
FP:	Stands for Function Points

2.2 Change Impact Analysis

Change impact analysis is the process of identifying potential consequences of a change, or estimating what needs to be modified to accomplish a change [19]. There are several CIA techniques that have been introduced in the literature such as: Use Case Maps (UCM), Class Interactions Prediction with Impact Prediction Filters (CIP-IPF) technique [20] Path Impact technique [21] and the Influence Mechanism technique [1].

These techniques are divided into two categories: (1) Static Impact Analysis (SIA) and (2) Dynamic Impact Analysis (DIA) [22]. The SIA technique analyzed the static information which is generated from software artifacts such as: requirements, design, class and test artifacts. On the other hand, DIA technique analyzed the dynamic information or executing code.

Existing CIA techniques are assumed that all classes in the class artifacts are fully developed; and secondly the class artifacts are used as a source of analysis, since it represents the final forms of user requirements [23]. Unfortunately, these assumptions are not practical for implementing CIA in the software development phase since some class artifacts are still not developed or partly developed [24].

However, studies [5, 22, 23] shows that the integration of these two techniques: (1) SIA and (2) DIA will be a good method for CIA during software development phase. A framework that has been developed for software development phase integrated SIA and DIA and considers all partially developed classes, fully developed classes and not developed classes. The framework is named as: Software Development Phase Change Impact Analysis Framework (SDP-CIAF) [23]. The SDP-CIAF as the following features; (1) It provides SIA which covers the software artifacts traceability from requirement until development phase as well as source code, (2) It includes dynamic analysis of the source code, (3) It considers the inconsistent states of the software artifacts (4) It is used for software development phase.

Hence, this study has selected SDP-CIAF framework as a CIA technique and combined it with FPA method to propose a new software effort estimation method which could be used to measure the amount of effort that is required to implement a software requirement change or changes during software development phase

3. PROPOSED METHOD

The new proposed method is the combination of two techniques, which are: (1) SDP-CIAF and (2) FPA method. The new proposed method may help software project managers in an accurate effort estimation during software development phase and it also helps software project managers in an effective change acceptance decision by accepting or rejecting software requirement changes.

This method has four main steps such as: Step 1: Software Requirement Analysis, Step 2: Calculating Function Points, Step 3: Performing Change Impact Analysis and Step 4: Estimating Required Effort as shown in Figure 1.

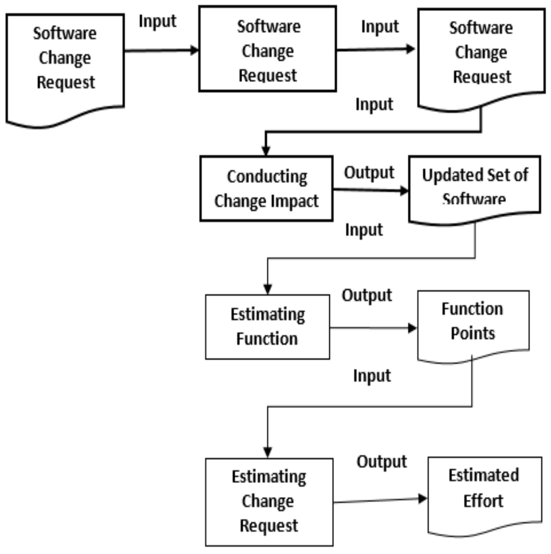

Figure 1. Software Development Effort Estimation Method

Phase 1: Software Change Request Analysis
It starts with a software change request form, having the following information: software change request name, software change request ID, software change request receiving date and software change request type (Addition, Deletion or Modification). At the end of this phase a document is generated and named as: software change request specifications which will be used as an input for Phase 2. This process will be repeated for each software change request.

Phase 2: Conducting Change Impact Analysis
In this phase the process of Change Impact Analysis (CIA) will be done. The process starts by checking the consequences of a software change request on software artifacts. Furthermore, it also checks the status of the class artifacts which may be Partially Developed (PD), Fully Developed (FD) or Not Developed yet and finally an updated set of software artifacts are generated which will used as input in phase 3. The process will be repeated for each software change request.

Table 1 shows the Code Development Status Multiplier (CDSM) for software requirement changes.

Table 1. Code Development Status Multiplier

S/NO	Development Status	Symbol	CDSM
1	Not Developed	ND	0.00
2	Partially Developed	PD	0.50
3	Fully Developed	FD	1.00

Phase 3: Estimating Function Points
In this Phase the number of Function Points will be calculated from Unadjusted Function Points and Technical Complexity Factor by using Equation (3). This phase will use the updated set of software artifacts as an input and identified five (5) type of functions such as: ILF, EIF, EI, EO and EQ by using the IFPUG manual rules [16, 18] and calculated UFPs and TCF using Equation (1) and Equation 2 respectively [25]. The process will be repeated for each software change request.

Phase 4: Estimating Change Request Development Effort
In this Phase the amount of effort that is required to implement a software change request will be estimated by using two inputs: (1) number of FPs and (2) updated set of software artifacts. Whereas, the effort is equal to productivity multiplied by number of function point [11]. For example, if the productivity is 8 hours for 1 function point. Then number of efforts will be equal to Productivity multiplied by FPs as shown in Equation (4). The process will be repeated for each software change request.

4. EVALUATION PROCESS

This section defines the method for conducting the evaluation process. During the evaluation process four main elements which have been considered are: (1) Case selection, (2) Data Collection, (3) Evaluation Metric, and (4) Evaluation Results.

4.1 Case Selection

To evaluate the results of new proposed method two case studies are selected i.e. Vending Machine Control System (VMCS) and Ticket Dispensing System (TDS). The selected software's are developed by the team of 5 (five) experienced members. The members are Master of Software engineering students and having industrial experiences.

Table 2. Data Collection

Project ID	SRC ID	SDP	SRC Type
P-1	CR-1	Analysis	Addition
	CR-2	Analysis	Addition
	CR-3	Analysis	Addition
	CR-4	Design	Deletion
	CR-5	Design	Modification
	CR-6	Coding	Addition
	CR-7	Coding	Modification
	CR-8	Coding	Deletion
	CR-9	Design	Modification
	CR-10	Design	Addition
P-2	CR-11	Analysis	Addition
	CR-12	Analysis	Addition
	CR-13	Analysis	Addition
	CR-14	Analysis	Deletion
	CR-15	Analysis	Modification
	CR-16	Coding	Addition
	CR-17	Coding	Modification
	CR-18	Testing	Addition
	CR-19	Testing	Modification
	CR-20	Testing	Deletion

4.2 Data Collection

During two case studies 20 (Twenty) software requirement changes have been introduced from both software projects. These changes are mostly introduced from all phases of software development life cycle. Later, these changes have been analyzed and a change request specification document is derived.

Table 2 is showing the data collection with Software Requirement Change ID, Software Development Phase (SDP) and Software Requirement Change (SRC) Type i.e. Addition, Deletion and Modification and Modification Type CRC is sub divided into two categories i.e. (Modification-Addition and Modification-Deletion).

4.3 Evaluation Metric

An evaluation metric named as Magnitude of Relative Error (MRE) is used for the evaluation of new proposed effort estimation method. It has calculated a rate of the relative errors in both cases of over-estimation or under-estimation as shown Equation (6) [26, 27].

$$MRE = \frac{Actual\,Estimated\,Effort - Estimated\,Effort}{Actual\,Estimated\,Effort} \qquad (6)$$

4.4 Evaluation Results

Table 3. MRE Values Produced from the new Proposed Method

Project ID	Requirement Change ID	Estimated Effort Man per Hour (SDEEM)	Actual Effort Man per Hour	MRE %
P-1	CR-1	56	55	0.018182
	CR-2	80	78	0.025641
	CR-3	33	31	0.064516
	CR-4	00	00	00
	CR-5	24	27	0.111111
	CR-6	88	85	0.035294
	CR-7	16	14	0.142857
	CR-8	16	15	0.066667
	CR-9	24	23	0.043478
	CR-10	56	58	0.034483
P-2	CR-11	80	82	0.02439
	CR-12	112	109	0.027523
	CR-13	56	57	0.017544
	CR-14	48	46	0.043478
	CR-15	32	30	0.066667
	CR-16	24	28	0.142857
	CR-17	48	51	0.058824
	CR-18	56	54	0.037037
	CR-19	32	34	0.058824
	CR-20	80	83	0.036145

Table 3 shows the results of the case studies using the new proposed method i.e. SDEEM. The evaluation process focused on the results between the estimated efforts with the actual effort. According to [28] whenever MRE value increases the estimation accuracy decreases.

Table 4 shows the MRE values produced from the new proposed model and Table 4 shows the MRE values Produced from the existing Function Point Analysis method. Whereas, Table 5 shows the comparison between the MRE values produced from existing Function Point Analysis method and the new proposed method.

Table 4. MRE Values Produced from existing Function Point Analysis Method

Project ID	Requirement Change ID	Estimated Effort with FPA method	Actual Value of Effort	MRE %
P-1	CR-1	56	55	0.018182
	CR-2	80	78	0.025641
	CR-3	33	31	0.064516
	CR-4	15	00	#DIV/0!
	CR-5	24	27	0.111111
	CR-6	88	85	0.035294
	CR-7	16	14	0.142857
	CR-8	25	15	0.666667
	CR-9	24	23	0.043478
	CR-10	56	58	0.034483
P-2	CR-11	80	82	0.02439
	CR-12	112	109	0.027523
	CR-13	56	57	0.017544
	CR-14	60	46	0.304348
	CR-15	32	30	0.066667
	CR-16	24	28	0.142857
	CR-17	48	51	0.058824
	CR-18	56	54	0.037037
	CR-19	32	34	0.058824
	CR-20	97	83	0.168675

Table 5: Comparison between the MRE Values Produced from Existing Function Point Analysis Method and the new Proposed Method

Project ID	Requirement Change ID	MRE Values with SDEEM	MRE Values with FPA method
P-1	CR-1	0.018182	0.018182
	CR-2	0.025641	0.025641
	CR-3	0.064516	0.064516
	CR-4	00	#DIV/0!
	CR-5	0.111111	0.111111
	CR-6	0.035294	0.035294
	CR-7	0.142857	0.142857

	CR-8	0.066667	0.666667
	CR-9	0.043478	0.043478
	CR-10	0.034483	0.034483
P-2	CR-11	0.02439	0.02439
	CR-12	0.027523	0.027523
	CR-13	0.017544	0.017544
	CR-14	0.043478	0.304348
	CR-15	0.066667	0.066667
	CR-16	0.142857	0.142857
	CR-17	0.058824	0.058824
	CR-18	0.037037	0.037037
	CR-29	0.058824	0.058824
	CR-20	0.036145	0.168675

5. DISCUSSION

The effort estimation accuracy of the new Software Development Effort Estimation Method has been evaluated by selecting two case studies and from each case study ten software change requests have been introduced during software development phase.

To review the results of the new proposed method this study has come up with some solutions of the problems which were encountered in our previous studies [5] the solutions are: (1) Proposed method can identify the impact of software requirement changes on software artifacts and (2) Proposed method can predict an accurate amount of effort for software requirement changes as compared to existing FPA method during software development phase.

The results of the proposed method are compared with the results from existing FPA method as shown in Table 5 and observed that the new proposed method is providing more accurate effort estimation results compared to FPA method. The main reason of providing more accurate estimation result with the new proposed method is because of Change Impact Analysis (CIA) technique. The CIA technique helps software project managers in knowing the actual status of class artifacts which are beneficial for an accurate estimation. While the existing FPA is limited in knowing the actual status of class artifacts which might produce inaccurate effort estimation results.

For example, a class is fully developed or partially developed is replaced with a new change request which is not developed yet. So, the class which is developed partially or fully shows that some effort has been already taken for its implementation. For an accurate effort estimation, it is important to know that effort that has been taken already. The new proposed method can help software project managers in an accurate effort estimation while the existing FPA method is limited to know the amount effort that has been taken already. The new proposed method which is the combination FPA with CIA can help project managers in estimating the accurate amount of effort for software requirement changes during software development phase.

6. CONCLUSION AND FUTURE WORK

This study proposed a new software effort estimation method that combines between Function Point Analysis (FPA) and Change Impact Analysis (CIA). As discussed earlier, the FPA method predicts the amount of effort for software requirement changes in initial phase of Software Development Life Cycle (SLDC). Whereas, CIA technique identifies the consequences of a software requirement change on software artifacts, and the development status of the code for software requirement change. The evaluation results shows that proposed method is able to give higher accuracy on the amount effort for a software requirement change during software development phase compared to the independent FPA method (without CIA technique).

The results of this study are the part of our ongoing research to overcome the challenges of accurate effort estimation for software requirement changes during software development phase. For future work, this study is aiming to conduct intensive tests of this method by considering more software requirement changes from different case studies.

7. ACKNOWLEDGEMENTS

This research project is sponsored under the 4C150, Contract research that is awarded by Majlis Agama Islam Selangor to Universiti Teknologi Malaysia.

8. REFRENCES

[1] S. Basri, N. Kama, and R. Ibrahim, "COCHCOMO: An extension of COCOMO II for Estimating Effort for Requirement Changes during Software Development Phase," 2016.

[2] P. Agrawal and S. Kumar, "Early phase software effort estimation model," in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, pp. 1-8.

[3] S. Grimstad and M. Jørgensen, "Inconsistency of expert judgment-based estimates of software development effort," *Journal of Systems and Software,* vol. 80, pp. 1770-1777, 11// 2007.

[4] 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.

[5] 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.

[6] 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.

[7] M. Jorgensen, "Practical guidelines for expert-judgment-based software effort estimation," *IEEE software,* vol. 22, pp. 57-63, 2005.

[8] A. Hira, S. Sharma, and B. Boehm, "Calibrating COCOMO&reg; II for projects with high personnel turnover," presented at the Proceedings of the International Conference on Software and Systems Process, Austin, Texas, 2016.

[9] M. Adnan and M. Afzal, "Ontology Based Multiagent Effort Estimation System for Scrum Agile Method," *IEEE Access,* vol. 5, pp. 25993-26005, 2017.

[10] A. B. Nassif, L. F. Capretz, and D. Ho, "Calibrating use case points," presented at the Companion Proceedings of the 36th

International Conference on Software Engineering, Hyderabad, India, 2014.

[11] 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.

[12] 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.

[13] A. J. Albrecht, "AD/M productivity measurement and estimate validation," *IBM Corporate Information Systems, IBM Corp., Purchase, NY,* 1984.

[14] 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.

[15] 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.

[16] D. Garmus and D. Herron, "Function Point Analysis: Measurement Practices for Successful Software Projects pdf," 2001.

[17] D. St-Pierre, M. Maya, A. Abran, J.-M. Desharnais, and P. Bourque, "Full function points: Counting practices manual," *Software Engineering Management Research Laboratory and Software Engineering Laboratory in Applied Metrics,* 1997.

[18] J. J. Cuadrado-Gallego, P. Rodriguez-Soria, A. Gonzalez, D. Castelo, and S. Hakimuddin, "Early Functional Size Estimation with IFPUG Unit Modified," in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*, 2010, pp. 729-733.

[19] D. Kchaou, N. Bouassida, and H. Ben-Abdallah, "UML models change impact analysis using a text similarity technique," *IET Software,* vol. 11, pp. 27-37, 2017.

[20] Asl and Kama, "A Change Impact Size Estimation Approach during the Software Development," in *2013 22nd Australian Software Engineering Conference*, 2013, pp. 68-77.

[21] S. Basri, N. Kama, and R. Ibrahim, "A Novel Effort Estimation Approach for Requirement Changes during Software Development Phase," *International Journal of Software Engineering and Its Applications,* vol. 9, pp. 237-252, 2015.

[22] 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.

[23] N. Kama and F. Azli, "A Change Impact Analysis Approach for the Software Development Phase," presented at the Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference - Volume 01, 2012.

[24] N. Kama, "Integrated Change Impact Analysis Approach for the Software Development Phase," *International Journal of Software Engineering & Its Applications,* vol. 7, p. 9, March 2013 2013.

[25] D. Longstreet, "Function points analysis training course," *SoftwareMetrics. com, October,* 2004.

[26] N. Nurmuliani, D. Zowghi, and S. Powell, "Analysis of requirements volatility during software development life cycle," in *2004 Australian Software Engineering Conference. Proceedings.*, 2004, pp. 28-37.

[27] S. Sabrjoo, M. Khalili, and M. Nazari, "Comparison of the accuracy of effort estimation methods," in *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2015, pp. 724-728.

[28] M. Jorgensen and K. Molokken-Ostvold, "Reasons for software effort estimation error: impact of respondent role, information collection approach, and data analysis method," *IEEE Transactions on Software engineering,* vol. 30, pp. 993-1007, 2004.