SOFTWARE REQUIREMENT CHANGE EFFORT ESTIMATION MODEL PROTOTYPE TOOL FOR SOFTWARE DEVELOPMENT PHASE

Jalal Shah[1], Nazri Kama[2], Nur Azaliah A Bakar[2] , Zuhaibuddin Bhutto[1] and Sohrab Khan[1]

[1]Department of Computer System Engineering & Sciences Balochistan, University of Engineering & Technology Khuzdar, Pakistan
[2]Department of Advanced Informatics, Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia

ABSTRACT

In software development phase software artifacts are not in consistent states such as: some of the class artifacts are fully developed some are half developed, some are major developed, some are minor developed and some are not developed yet. At this stage allowing too many software requirement changes may possibly delay in project delivery and increase development budget of the software. On the other hand rejecting too many changes may increase customer dissatisfaction. Software change effort estimation is one of the most challenging and important activity that helps software project managers in accepting or rejecting changes during software development phase. This paper extends our previous works on developing a software requirement change effort estimation model prototype tool for the software development phase. The significant achievements of the tool are demonstrated through an extensive experimental validation using several case studies. The experimental analysis shows improvement in the estimation accuracy over current change effort estimation models.

KEYWORDS

Software Change Effort Estimation, Software Requirement Changes, Change Impact Analysis and Software Development Phase.

1.INTRODUCTION

Software change effort estimation (SCEE) is the process of predicting the required amount of effort that is required to implement a software requirement change. SCEE can be done in software miantance phase and software development phase (i.e. requirement analysis, design, coding or testing). In software maintenance phase software artifacts are in consistent states means that all the classes are fully developed. Whereas, in software development phases software artifacts are not in consistent state such as: some of the class artifacts are fully developed, some classes are half developed, some classes are major developed, some classes are minor developed and some classes are not developed yet. It is important to manage the changes in the software to meet the evolving needs of the customer and hence, satisfy them [1]. Accepting too many changes causes delay in the completion of the software and it incurs additional cost. While, rejecting the software requirement changes may cause dissatisfaction to the customers [2, 3]. Therefore, it is important for the software project managers to manage these changes during software development by making an effective decision. One type of information that helps to make an effective decision is the prediction of the number of classes which were affected by these changes and it can be done by performing change impact analysis [4]. In this study a

DOI: 10.5121/ijsea.2019.10302

software change effort estimation model prototype tool is developed for software development phase by using change impact technique with software change effort estimation.

This paper is structured as follows: Section (2) presents related work, section (3) describes Proposed prototype tool, section (4) data collection, results and discussion and section (5) Presents conclusion and future work.

2. RELATED WORK

The five most related keywords involved in this research are Software Change Effort Estimation, Change Impact Analysis, Software Development Phase, Function Point Analysis and Constructive Cost Model II.

2.1. Software Change Effort Estimation

Software Change Effort Estimation (SCEE) is the process of predicting that how much work and how many hours of work are required to develop a software. Normally it describes in mandays or man-hours unit [5]. Several SCEE models have been developed such as: Expert Judgement [6, 7]; Estimation by Analogy [8]; Source Lines of Code [9] Function Point Analysis [3] and Regression Analysis [7]. According to Sufyan, et al. [10] generally software development teams preferred to use expert judgement effort estimation model instead of other estimation models because of its flexibility and simplicity. Furthermore, they mentioned that it is not sure whether the estimation results which were produced during this model are hundred percent accurate or not. Additionally, it is also observed that unstructured expert judgment is extremely unpredictable. According to Idri, et al. [8] and Sufyan, et al. [10], estimation by analogy predicts the amount of effort required for new software projects and uses the statistics of the similar software projects which were developed earlier. Furthermore, they have stated that because of simplicity and flexibility of analogy model it is used frequently as a hybrid model. Whereas, in hybrid model, two or more SCEE models are combined for the reason to improve the accuracy and performance such as particle swarm optimization (PSO), grey relational analysis (GRA), artificial neural network (ANN), principle component analysis (PCA), and rough set theory [11]. Source Lines of Code (SLOC) is the most ordinarily used metric to denote size of software during SCEE in algorithmic methods [9]. On the other hand, Hira, et al. [12] stated that estimating with SLOC is nearly impossible until the source code developed completely. Furthermore, they have stated that SLOC gives two different estimation results for different programming languages, because the number of lines of code in each language is different. Function Point Analysis (FPA) is also an algorithmic-based SCEE model [13]. It is used for measuring the size and complexity of a software by calculating the functionality, that system provides to its users [14]. While, in late 1970s Allan Albrecht acknowledged that measuring effort estimation by SLOC is insufficient. For that reason, in 1979 he introduced FPA method [15].

Regression analysis effort estimation uses mathematical approaches for measuring effort estimation. Furthermore, it uses two variables such as: (i) SLOC and (ii) FPA for software size measurement. On the other hand, some regression analysis models use different parameters such as: operating system or programming language for an independent variables [16, 17]. The benefit of using regression analysis model for effort estimation is its accuracy in measurement. Among other regression analysis models the most famous one is Constructive Cost Model (COCOMO) II presented by Boehm [18]. Some earlier researchers have stated the significance of this method for estimating effort [12, 19].

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 [20].The motivation behind CIA study is to identify the implications of change on software artifacts. The change can be in any form such as addition, modification or removal of existing or new software artifacts. Whereas, the information of affected artifacts can help software project managers in taking effective decisions regarding to the change. As, Asl and Kama [21] stated that change impact analysis can help project managers in decision making, that whether to accept or reject the software requirement changes based on predicted consequences.

Several CIA techniques have been developed such as: Use Case Maps (UCM) technique [22], Class Interactions Prediction with Impact Prediction Filters (CIP-IPF) technique [21], Path Impact technique [5], Influence Mechanism technique [19], and SDP Change Impact Analysis Framework (SDP-CIAF) [23]. These techniques are divided into two categories named as: Static Impact Analysis and Dynamic Impact Analysis. Static Impact Analysis (SIA) technique considers static information from software artifacts to produce a set of possible impact classes. Some of the common SIA techniques are: Use Case Maps (UCM) technique [22] and Class Interactions Prediction with Impact Prediction Filters (CIP-IPF) technique [21]. Whereas, Dynamic Impact Analysis (DIA) techniques considers dynamic information created by implementing the code to generate a set of potential impact classes [24]. Some of Common DIA techniques are: Path Impact technique [5] and the Influence Mechanism technique [19].

3. PROPOSED MODEL

The study has developed a prototype tool to systematize the entire processes of SCEE model. The Software Change Effort Estimation Model Prototype Tool (SCEEMPT) has four main steps as follows:

- Step 1: Change Request Form
- Step 2: Unadjusted Function Points
- Step 3: Change Impact Analysis
- Step 4: Effort Estimation

Step 1: Change Request Form

In this step the required data will be provided from the user. The Change Request Form (CRF) in the SCEEMPT requires the important change request attributes which include the change request name, change requester name, change request type, change request receiving date, daily working hours and affected requirements. A running identification number will be assigned to every project name change request name automatically by the SCEEMPT. The CRF is shown in Figure 1.

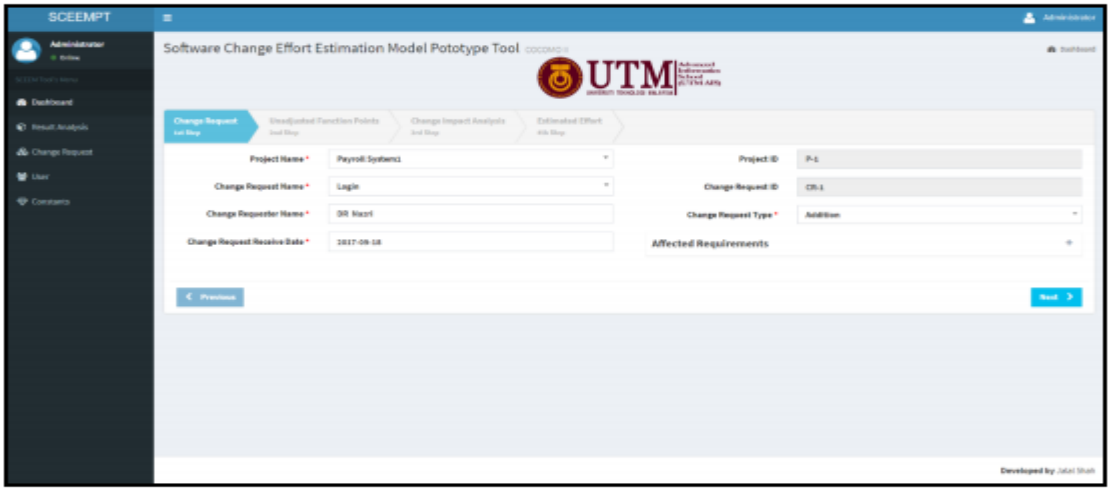

Figure 1 Change Request Form

Unadjusted Function Points

In this step user will enter the number of data functions such as External Inputs, External Outputs, Internal Logical Files, External Interface Files and External Quires with the level its complexity. These data functions will be selected from the user requirement by using the rules as described in IFPUG Manual [15]Then the system will calculate the total number of Unadjusted Function Points which is the sum of all data functions as shown in Figure 2.

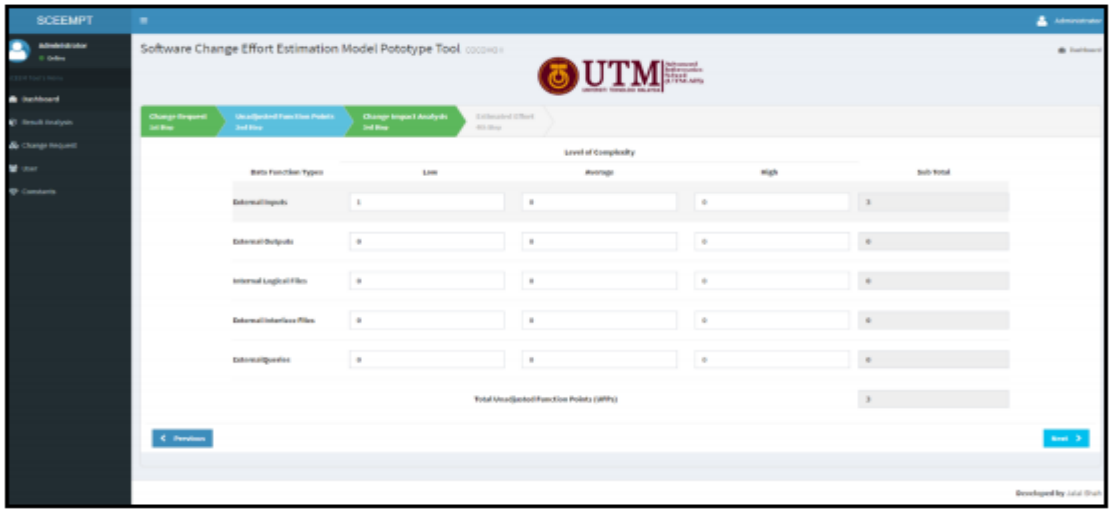

Figure 2 Calculating Total number of Unadjusted Function Points

Step 3: Performing Change Impact Analysis

In this step the task of Change Impact Analysis (CIA) will be performed. During this step user will browse the required file from the source code. The system will read the development status of code such as: Fully Developed, Half Developed, Major Developed, Minor Developed and Not Developed with the help of CIA and a multiplier is assigned for each development state as shown in Table 1 and then the size of change request will be calculated by using Equation 2 or Equation 3 for different type of change requests. In the next step the amount of effort will be estimated by using Equation 4.

Table 1 Development Status Multiplier

Development State	Symbol	Multiplier	Completion %
Fully Developed	FD	1	100 %
Major Developed	MD	0.75	75%
Half Developed	HD	0.50	50%
Minor Developed	Min D	0.25	25%
Not Developed	ND	0	0%

Basri, et al. [5] has used a marking technique to identify the code status. Whereas, this study has modified [5] marking technique and developed a new marking technique; which identifies the development state of the code and the name of programing language in which the source code isdeveloped. The marking technique is described as: [Special Tagging <LanguageTag> CodeStatus </LanguageTag>], where <LanguageTag> is used for the programming language in which the code is developed. Initially, a technique developed by Jones in the 1980s and named as backfiring or conversion ratio [25]. In conversion ratio technique the number of UFPs will convert into the number of source lines of code. This study is using the conversion ratio developed by Quantitative Software Management (QSM) [26]. As a result, system will provide the status and size of the particular change request as shown in Figure 3.

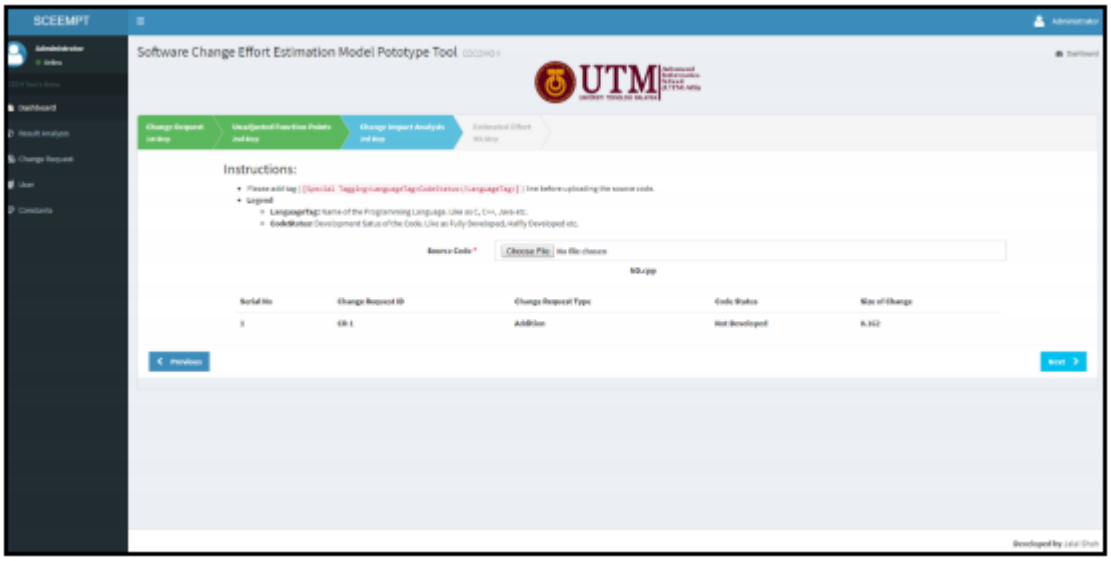

Figure 3 Performing Change Impact Analysis

Step 4: Effort Estimation

In this step the amount of effort for a specific change request implementation will be estimated after receiving the required information such as Value of A (constant), Scale Drivers Exponent B and Value of Effort Multipliers, Development Phase and change request completion date. The Value of A (constant), Scale Drivers Exponent B and Value of Effort Multipliers will be calculated by using COCOMO II [27] as shown in Figure 4.

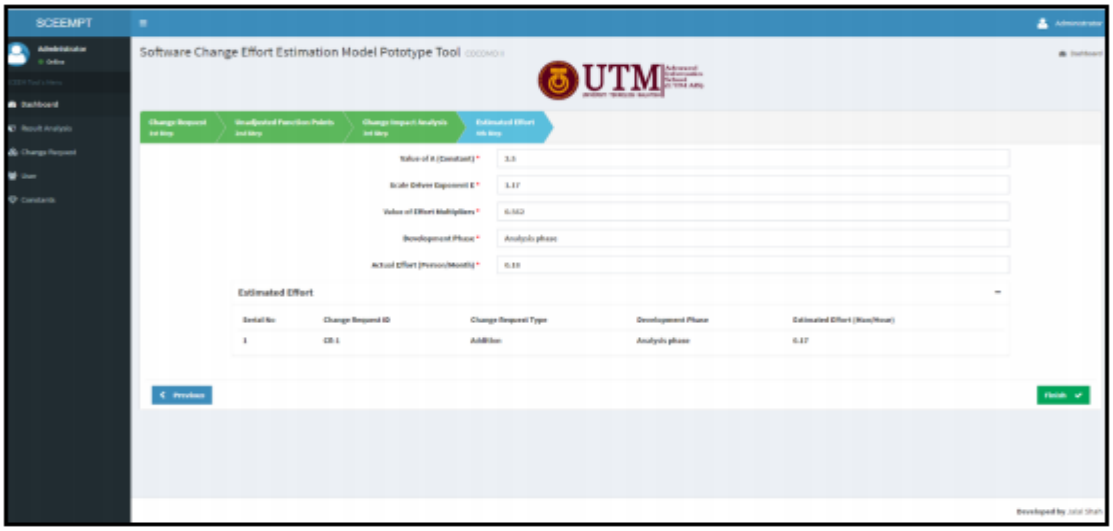

Figure 4 Effort Estimation

4. DATA COLLECTION

In this study four software projects such as: (i) Payroll System, (ii) Vending Machine Control System, (iii) On Board Automobile System and (iv) Software Requirement Change Effort Estimation Prototype Tool are selected as case selections and the following documents have been collected during the data collection.

- Change Request Form
- Software Requirement Specifications Document
- Software Design Document
- Source Code
- Progress Report (used for actual amount of effort)

5. RESULTS AND DISCUSSIONS

The applicability of the newly developed software effort estimation model prototype tool is checked by performing an experiment.

The hypothesis of the experiment are:

H: The SRCEEM is not applicable to estimate the change effort for software requirement changes during software development phase.

H2: The SRCEEM is applicable to estimate the change effort for software requirement changes during software development phase.

Table 2 shows the results of the software projects which were selected as case selections. Whereas; the estimated effort which was produced by SCEEM, actual implementation effort which was recorded during the development of software requirement change(s) and MRE value (percentage of discrepancy between estimated effort and actual implementation effort) sorted by the Project ID and Change Request (CR) ID. A total number of 50 Change Requests (CR) that had been introduced from the case selection software projects during development phase.

Table 1 Case Selection Software Projects Experiment Results by SCEEM

Project ID	Change Request ID	Change Request Type	SRCEEM Estimated Effort Man/Month	Actual Effort Man/Month	MRE
P-1	CR-1	Addition	0.160	0.18	0.111111
	CR-2	Addition	0.163	0.185	0.118919
	CR-3	Addition	0.150	0.13	0.153846
	CR-4	Modification Addition	0.160	0.188	0.148936
	CR-5	Modification Deletion	0.031	0.032	0.03125
	CR-6	Addition	0.224	0.302	0.258278
	CR-7	Deletion	0.233	0.25	0.068
	CR-8	Addition	0.224	0.199	0.125628
	CR-9	Modification Addition	0.160	0.18	0.111111
	CR-10	Addition	0.163	0.23	0.291304
P-2	CR-11	Addition	0.156	0.19	0.178947
	CR-12	Modification Addition	0.156	0.22	0.290909
	CR-13	Modification Deletion	0.21	0.22	0.045455
	CR-14	Addition	0.156	0.18	0.133333
	CR-15	Modification	0.219	0.19	0.152632
		Addition			
	CR-16	Deletion	0.097	0.09	0.077778
	CR-17	Modification Addition	0.156	0.13	0.2
	CR-18	Modification Addition	0.219	0.18	0.216667
	CR-19	Addition	0.219	0.199	0.100503
	CR-20	Addition	0.156	0.13	0.2
	CR-21	Addition	0.219	0.188	0.164894
	CR-22	Addition	0.156	0.14	0.114286
	CR-23	Addition	0.156	0.188	0.170213
P-3	CR-24	Addition	0.318	0.18	0.766667
	CR-25	Modification Addition	0.318	0.355	0.104225
	CR-26	Addition	0.318	0.2	0.59
	CR-27	Modification Deletion	0.198	0.189	0.047619
	CR-28	Addition	0.445	0.27	0.648148
	CR-29	Modification Addition	0.318	0.28	0.135714
	CR-30	Addition	0.319	0.411	0.223844
	CR-31	Modification Addition	0.318	0.28	0.135714
	CR-32	Modification Addition	0.445	0.55	0.190909
	CR-33	Modification Addition	0.455		

				0.6	0.241667
	R-34	Addition	0.318	0.199	0.59799
	CR-35	Modification Addition	0.319	0.399	0.200501
P-4	CR-36	Addition	0.318	0.365	0.128767
	CR37	Addition	0.445	0.395	0.126582
	CR-38	Deletion	0.445	0.413	0.077482
	CR-39	Addition	0.465	0.55	0.154545
	CR-40	Addition	0.318	0.388	0.180412
	CR-41	Addition	0.322	0.2	0.61
	CR-42	Addition	0.445	0.55	0.190909
	CR-43	Modification Addition	0.455	0.435	0.045977
	CR-44	Addition	0.318	0.412	0.228155
	CR-45	Addition	0.322	0.365	0.117808
	CR46	Addition	0.445	0.395	0.126582
	CR-47	Modification Addition	0.136	0.158	0.139241
	CR-48	Modification Deletion	0.234	0.31	0.245161
	CR-49	Modification Addition	0.19	0.222	0.144144
	CR-50	Addition	0.195	0.231	0.155844

To validate the hypotheses, the Mean Magnitude Relative Error (MMRE) of each case selection software project and overall case selection software projects were calculated. Table 4 shows the results of MMRE for each case selection software project and overall case selection software projects.

Table 4 MMRE Results of all Case Selection Projects

Project ID	MMRE	Overall MMRE	PRED (.25)
P-1	0.165138	0.202594	0.88
P-2	0.14676		
P-3	0.311786		
P-4	0.186692		

To recapitulate, the evaluation focused on comparing results between the estimated effort with

the actual effort. The MMRE and Percentage of Prediction, PRED (.25) are used as the comparison metric. According to (Huang et al., 2008), an acceptable MMRE value (or error rate) for software effort estimation is 25% or (.25).

The first experiment results show that the overall MMRE value produced by the SRCEEM is under 25%. This result can be concluded that the SRCEEM is applicable in estimating the software requirement change effort for requirement changes during software development phase and PRED (.25) value of all selected software projects produced by the proposed SRCEEM is 0.88. Which shows that the new proposed SRCEEM is 88% applicable for estimating the amount of software requirement change effort for SRC during SDP. Hence, the analysis of this experiment rejects the H1 and accepts the H2 of the hypothesis.

6. CONCLUSION AND FUTURE WORK

This research has proposed a new Software Change Effort Estimation Model Prototype Tool for software development phase which is the combination of Effort Estimation and Change Impact Analysis techniques. Whereas, for effort estimation this research selected COCOMO II model and for Change Impact Analysis it selected SDP-CIAF. Using CIA technique in effort estimation model the proposed model generated very good results as shown in Table 3. This is possible only when the development status (i.e. Fully Developed, Major Developed, Half Developed, Minor Developed and Not Developed) of a software requirement change can be traced accurately. Whereas, Table 4 shows that the MMRE value of each individual project and the overall projects which is less than 25% it means that the new proposed model is applicable for effort estimation of SRCs during SDP.

The results of this research are the part and parcel of our ongoing research to overcome the challenges of accurate effort estimation for SRCs during SDP. For future work, this research is aiming to conduct an empirical study to check the effort estimation accuracy by comparing the estimation results produced from existing effort estimation model with the new Software Change Effort Estimation Model Prototype Tool.

REFRENCES

[1] J. Shah and N. Kama, "Extending Function Point Analysis Effort Estimation Method for Software Development Phase," presented at the Proceedings of the 2018 7th International Conference on Software and Computer Applications, Kuantan, Malaysia, 2018.

[2] J. Shah, N. Kama, and N. A. A. Bakar, "ANovel EFFORT ESTIMATION MODEL FOR SOFTWARE REQUIREMENT CHANGES DURING SOFTWARE DEVELOPMENT PHASE," 2018.

[3] N. K. b. Jalal Shah*a, Amelia Zahari, "AN EMPIRICAL STUDY WITH FUNCTION POINT ANALYSIS FOR REQUIREMENT CHANGES DURING SOFTWARE DEVELOPMENT PHASE," in ASIA International Multidisciplinary Conference 2017, Johor Bharu, 2017.

[4] N. K. Jalal Shah, Saiful Adli Ismail, "An Empirical Study with Function Point Analysis for Software Development Phase Method," presented at the 2018 7th International Conference on Software and Information Engineering (ICSIE 2018), Cairo, Egypt, 2018.

[5] 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.

[6] O. Fedotova, L. Teixeira, and H. Alvelos, "Software Effort Estimation with Multiple Linear Regression: Review and Practical Application," J. Inf. Sci. Eng., vol. 29, pp. 925-945, 2013.

[7] 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.

[8] 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.

[9] M. Kaur and S. K. Sehra, "Particle swarm optimization based effort estimation using Function Point analysis," in Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on, 2014, pp. 140-145.

[10] 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.

[11] 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.

[12] 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.

[13] F. Ferrucci, C. Gravino, and L. Lavazza, "Simple function points for effort estimation: a further assessment," presented at the Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, 2016.

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

[15] P. Vickers and C. Street, "An Introduction to Function Point Analysis," School of Computing and Mathematical Sciences, Liverpool John Moores University, Liverpool, UK, 2001.

[16] V. Anandhi and R. M. Chezian, "Regression techniques in software effort estimation using cocomo dataset," in Intelligent Computing Applications (ICICA), 2014 International Conference on, 2014, pp. 353-357.

[17] 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.

[18] B. W. Boehm, Software Cost Estimation with Cocomo II: Prentice Hall, 2000

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

[20] 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.

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

[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] Kama and M. Halmi, "Extending Change Impact Analysis Approach for Change Effort Estimation in the Software Development Phase," in WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering Series, 2013.

[25] J.-M. Desharnais and A. Abran, "Approximation techniques for measuring function points," in Proceedings of the 13th international workshop on software measurement (IWSM 2003), 2003.

[26] Q. S. Management. (2018, 10/04/2018). Function Point Languages Table (5th version ed.). Available: http://www.qsm.com/resources/function-point-languages-table

[27] 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.

AUTHORS

Jalal Shah, He is an Assistant Professor at Balochistan University of Engineering &Technology Khuzdar Pakistan. He graduated in Bachelor of Computer System Engineering from Balochistan University of Engineering & Technology Khuzdar Pakistan. Later, he obtained a Master Degree from Universiti Teknologi Malaysia (UTM) in Software Engineering. In 2018, he received a Doctorate in Software Engineering from Universiti Teknologi Malaysia (UTM.

Nazri Kama, He is an Associate Professor at Universiti Teknologi Malaysia (UTM) specializing in software engineering. He graduated in Bachelor in Management Information System from Universiti Teknologi Malaysia. Later, he obtained a Master Degree from the same university in Real-time Software Engineering. In 2011, he received a Doctorate in Software Engineering from the University of Western Australia in Australia.

Nur Azaliah Abu Bakar, PhD is a Senior Lecturer at Advanced Informatics Department, Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia. She graduated with a Bachelor (Information Technology) in Information Systems Engineering from Multimedia University (MMU) Malaysia (2000). She then obtained her Masters in Information Technology from Universiti Teknologi Mara (UiTM) in 2004. In 2017 she was awarded a Doctor of Philosophy degree in Information Technology (Enterprise Architecture) by Universiti Teknologi Malaysia (UTM).

Zohibuddin Bhutto He is an Assistant Professor at Balochist an University of Engineering &Technology Khuzdar and pursuing his PhD degree from South Korea. He graduated in Bachelor of Software Engineering from Mehran University of Engineering & Technology Jamshoro pakistan. Later, he obtained a Master Degree from Mehran University of Engineering & Technology Jamshoro pakistan in IT.

Sohrab Khan. He is an Assistant Professor at Balochistan University of Engineering and Technology Khuzdar, specializing in information systems. He graduated in Bachelor's in Computer Engineering from Sir Syed University of Engineering and Technology Karachi. Later he obtained a Master Degree from Stockholm University Sweden in Computer and System Sciences. In 2019, he received a Doctorate degree from Universiti Teknologi Malaysia (UTM).