Quality Extended Use Case Point (QUCP): An Improved Cost Estimation Method

Ibrahim Mohammad Ba'abbad¹; M. Rizwan Jameel Qureshi²

¹Department of Information Technology & King Abdul-Aziz University, Saudi Arabia

²Department of Information Technology & King Abdul-Aziz University, Saudi Arabia

¹eabad@stu.kau.edu.sa; ²rmuhammd@kau.edu.sa

DOI: 10.47760/ijcsmc.2021.v10i06.001

Abstract— *The quality of a product is one of the major interests of the manufacturing process in all industries. The software industry imposes to construct a project with several phases to ensure producing high-quality software. A software development company estimates time, effort and cost of the project during planning phase. It is important to have accurate estimations to reduce the risks of project failure. Several cost estimation methods are practiced in the software development companies such as Function Point (FP), Use Case Points (UCP), Constructive Cost Model I and II and Story Points (SP). UCP cost estimation method is taken in this research to improve the accuracy of its estimation. UCP estimation depends on the use case diagram of the proposed system. A use case diagram describes the main functional requirements of the proposed system. UCP partially considers non-functional requirements through the technical and environmental factors. There is a lacking in the UCP method to consider the importance of quality attributes in the estimating process. This paper proposes an extended version of the existing UCP method named Quality Extended Use Case Point (QUCP) method in which quality attributes are included to improve the accuracy of cost estimation. A questionnaire is used to validate the proposed QUCP method. It is found after data analysis that seventy five percentages of the participants are agreed that the proposed method will not only help to improve the accuracy of cost estimation but it will also enable a software development company to deliver high-quality products.*

Keywords— *Software Project Estimation, Cost Estimation, Use Case Point, Constructive Cost Model, High-Quality Software*

I. INTRODUCTION

Software cost estimation is a cumbersome task because it requires the estimator to predict cost accurately at the planning phase of the project. Some software cost estimation methods are proposed over the years such as Function Point Analysis (FPA), Cost Constructive Model (COCOMO) I & II, Use Case Points (UCP) and story point [1-5]. Some researchers combine them with novel approaches to generate more accurate estimations of software projects. There are several attempts to modify the UCP method to have more accurate estimations [6-8]. However, the quality of the released software at the planning phase is not considered. Poor quality software also increases delivery cost. The traditional UCP method depends on the use case diagram to describe the functional

requirements of object-oriented software. UCP counts the number of interactions between use cases and actors and it assigns numeric values on complexity scales. Although, non-functional requirements are partially considered through the technical and environmental factors, the lack of considering the quality of software project in the estimating process is still present. Our proposed method extends the traditional UCP technical factors in which they will describe the quality of the software wider. We generate measuring statements for the International Organization for Standardization (ISO) quality attributes [9]. We named our new approach as Quality Extended Use Case Point (QUCP) method. QUCP method helps to generate more realistic estimation since the quality of the software product has ensured. We distribute a questionnaire with twenty questions to the information technology (IT) departments' employees of different local and international companies as well as to IT graduate students in order to validate the proposed method goal achievement.

This research contains the following sections: section 2 presents a literature review for the improvements of related software project estimation methods. Section 3 defines the current problem to be addressed. Section 4 summarises the calculations of the traditional UCP method. Section 5 exhibits the proposed QUCP method. Section 6 covers the validation of the proposed solution.

II. RELATED WORK

As a new concept, machine learning techniques were used to enhance the estimation of the software project effort by BaniMustafa [1]. Naïve Bayes, logistic regression and random forests techniques were applied to the historical data set of ninety-three projects. Results are compared with the constructive cost model (COCOMO). Consider improving the COCOMO drivers and multipliers can provide more accurate results in the estimation effort process.

The agile software development methodology is used widely to enhance software delivery time. Cost estimations in agile rely on a variety of project attributes including historical data of the project or expert opinion. A new model based on Principal Component Analysis (PCA) was developed by Garg and Gupta [5]. PCA used to highlight the key attributes that affect costs estimation to gain further enhancement of the development process. PCA is working even when project historical data are not available. The statistical Toolbox of MATLAB software was used to establish the principal component list using the PCA model. More than seventy nine percentages of project costs are described by twelve principal components. It is suggested to generalize the proposed model to deal with multiple types of agile software development projects.

As another attempt, an algorithmic method was proposed to produce more accurate estimation in agile software development [10]. To compare the results, three scenarios are used that vary the consideration of effort estimation factors. As a limitation of the proposed method, the estimator still gives personal judgments to form some input information to the algorithmic proposed method.

Failure to understand the software project requirements is one of the most important reasons for inaccurate estimations. Non-functional requirements are rarely included in the project estimations. A hybrid method consisting of the Case Base Reasoning (CBR) method and COSMIC model is developed to improve the accuracy of software project estimations. CBR is an Artificial Intelligence (AI) model used to solve current problems by adapting previous solutions to similar problems. COSMIC is a functional sizing technique supported by the International Organization for Standardization (ISO). In the proposed method, it is assumed that functional requirements (FRs) are linked to non-functional requirements (NFRs). The proposed approach is needed to be validated through a real case study [11].

The global Software Development (GSD) model is used by global software development companies. GSD provides a huge number of multi-skilled personnel with short time-to-market intervals. However, making estimations for GSD projects is a hard task since the GSD model involves geographical, temporal, and socio-cultural factors. A scheduling-based method is introduced by Ramacharan and Roa [12] to produce a more accurate estimation for GSD projects. The proposed method provides calibration parameters to yields accurate estimations. MATLAB software was used to form a comparison between the new method, COCOMO II and SLIM models. A comparison between model-based and expert-based approaches is recommended.

Kurniawan, et al. [13] developed an analogy-based method to estimate the Owner Estimated Cost (OEC) of the software project. This method applied the previously experienced judgments of similar projects to the current project. Usual used algorithmic methods such as Use Case Point (UCP) method has some limitations of calculating the estimated cost since it depends on only fixed use case complexity that cannot consider uncertain conditions. This study aims to switch from the algorithmic method to the analogy method which has better performance in estimating OEC for a government agency. UCP method is used as an input parameter to the analogy-based method since it can be provided in the planning phase of the system development lifecycle (SDLC). A mathematical model was proposed with nine equations to calculate the OEC of software projects. The results show the estimated cost deviation is around seven percent on average from the actual cost. However, this study is limited to Indonesian regulations, it would be used as a unified standard analogy method over the world if it is distributed worldwide to follow the world regulations.

Several factors affect the process of cost estimation in the software development project and sometimes can cause project failure. Suliaman and Kadoda [14] tried to figure out those factors in Sudanese software development companies. Both qualitative and quantitative study methods are used. Interviews are conducted with professionals in software engineering companies in Khartoum. On the other hand, a survey was distributed among students at the Faculty of Mathematical Science, University of Khartoum. As a result, both technical and non-technical factors cause a software project to fail. Lack of professional practice and local standards on the software development process was the major reason for failure. More research is required to study the level of application of the local software development standards in Sudan.

Optimizing Software Cost Estimation (SCE) with accuracy is one of the challenges to software professionals. A hybrid technique was developed using a genetic algorithm and a Tabu search algorithm to optimize SCE. NASA projects' data set was used as input data. Gharehchopogh et al. [15] found that the hybrid technique ensures more accurate SCE compared to the normal COCOMO method as the number of projects increased. Reducing the execution time for the hybrid algorithm is the next step for developing further.

The Function Point Analysis (FPA) method is used to estimate software project size. However, it does not follow rapid technological development. A learning-based adjustment model is proposed to the FPA method by applying a Genetic Algorithm (GA) and analogy-based estimation. The analogy-based method is used to determine the coefficients for each one of the Key Influence Factors (KIFs) for the FPA method. Over six hundred projects of the International Software Benchmarking Standards Group (ISBSG) dataset are used as a case study to validate the proposed solution. Liu et al. [2] recommend more research to find alternative methods to select KIFs.

Hira and Boehm [6] improve cost estimation methods by merging Use Case Points (UCPs) and COCOMO II. It is an attempt to enhance the accuracy of cost estimation by proposing a hybrid cost estimation method. The relative difficulty factor is added helping to estimate the technical and environmental factors of a software project. The problem with the UCP method is that it does not consider verified technical and environmental factors. It also depends on the number of transactions of each use case rather than the complexity of the transactions themselves reducing the accuracy of estimation. The proposed method is taking advantage of verified COCOMO II parameters by plugin them into straight-forward UCPs method. A mathematical model is used consisting of a set of linear equations to prove the proposed model. After using a limited dataset to test the new model, a linear proportional relationship between the hybrid method and estimated effort was reported. The results will be further improved if a large dataset is used [6].

Cost drivers are one of the main input parameters of the Constructive Cost Model (COCOMO II). Major effects over cost estimates are driven by cost drivers. Bat Algorithm (BA) is used by Girotra and Sharma [16] to find the best values of fifteen cost drivers. NASA-93 dataset is used as validators. A comparison between the proposed algorithm, COCOMO and genetic algorithm is conducted. Considering other software engineering methods with different datasets can enhance the study purpose further.

It is a bit difficult to estimate the software project effort at the early development stages. Iterative models were proposed by Qi and Boehm [7] to make effort estimation more precise. The constructive cost model (COCOMO II) approach is combined with the use case points (UCPs) method to form the incremental hybrid model. Data collected from four projects is used to test the proposed models. However, more data with large scale are needed to validate the accuracy of the new models further.

Estimating the size of the software is mainly done by involving experts' opinions. To automate the software sizing process, Qi and Boehm [8] proposed the Detailed Use Case Points (DUCPs) method. DUCPs method directly counts the use case points from the sequence and class - unified modelling language (UML) - diagrams. Twenty-two historical projects were used to validate the proposed method. However, large data sets are needed to further improve the effectiveness of the DUCPs method.

Priorities of use cases are not accurately specified in the traditional use case point (UCP) method. Park et al. [3] proposed an improved UCP method. The proposed method is applied theoretically using vehicle supplies management system with twenty-two use cases. The improved UCP method suggests assigning weights to Include and Extend relations between use cases. A real implementation of the proposed method is needed to validate the accuracy of requirements priorities generated.

One of the most used methods to estimate the cost of software projects is the Use Case Point (UCP) method. Factors used in the UCP method possess some limitations regarding software project risks. Bagheri and Shameli-Sendi [4] consider that every software project is surrounding by eight aspects. Four of them represent the essentials aspects of a software development process which are: requirement aspect, design aspect, implementation aspect and test aspect. The other four are auxiliaries' aspects such as quality aspect, risks aspect, security aspect, and reporting aspect. In [4], the risk aspect is accomplished. An improved version of the UCP method called Risk Extended Use Case Point (RUCP) method is developed. Sixty risks are classified into eight domains of software project risks. The new sub-factors are included through the environmental factors (EFs) of the old UCP method. Five companies tested the proposed RUCP method. Quantitative results show that a

significant improvement of deviations between actual and estimated cost using the RUCP method comparing with the old UCP method. The remaining seven aspects of a software project are needed to study as future work. The limitations of existing work are shown in Table 1.

III. PROBLEM STATEMENT

Use Case Point (UCP) estimation method has some limitation since it depends only on the number of transactions of each use case. This fixed use case complexity cannot consider the software quality aspect to estimate the software project cost [4-6]. Following is the problem identified from the literature review to be addressed in this research.

- How to improve existing software projects estimation method while ensuring high-quality software to be delivered?

IV. THE PROPOSED SOLUTION

In order to explain our proposed solution as extend from the existing UCP method, we should explore the original method. UCP is first introduced in 1993 and it is used to estimate the software project resources using an object-oriented approach [17]. UCP is based on the use case diagram which is one of the static diagrams related to Unified Modelling Language (UML) technique.

UCP aims to estimate the cost of developing a software project by estimating the effort needed to complete the project. This estimation is based on the complexity of the project which can be examined by four factors.

TABLE I
SUMMARY OF LIMITATIONS

Paper Title	Limitation
Predicting Software Effort Estimation Using Machine Learning Techniques [1]	Improving the COCOMO drivers and multipliers can provide more accurate results in the estimation effort process.
A Learning-Based Adjustment Model with Genetic Algorithm of Function Point Estimation [2]	More research is needed to find alternative methods to select KIFs.
Improving Use Case Point (UCP) based on Function Point (FP) Mechanism [3]	A real implementation of the proposed method is needed to validate the accuracy of requirements priorities generated.
Software Project Estimation Using Improved Use Case Point [4]	Requirement aspect, design aspect, implementation aspect, test aspect, quality aspect, security aspect, and reporting aspect of a software project are needed to study as future work.
PCA based cost estimation model for agile software development projects [5]	Dealing with multiple types of agile software development projects is suggested to generalize the proposed model.
Combating Use Case Points' Limitations with COCOMO(R) II and Relative Difficulty [6]	The results will be further improved if a large dataset is used.
A Light-Weight Incremental Effort Estimation Model for Use Case Driven Projects [7]	More data with large scale are needed to validate the accuracy of the new models further.
Detailed Use Case Points (DUCPs): A Size Metric Automatically Countable from Sequence and Class Diagrams [8]	Large data sets are needed to further improve the effectiveness of DUCPs method.
Effort, Duration and Cost Estimation in Agile Software Development [10]	As a limitation of the proposed method, estimator still gives personal judgments to form some input information to the algorithmic proposed method.
Analyzing the non-functional requirements to improve accuracy of software effort estimation through Case Based Reasoning [11]	The proposed approach is needed to be validated through a real case study.
Scheduling Based Cost Estimation Model: An Effective Empirical Approach for GSD Project [12]	A comparison between model-based and expert-based approaches is recommended.

Development of Analogy-Based Estimation Method for Software Development Cost Estimation in Government Agencies [13]	The study is limited to Indonesian regulations, it would be used as a unified standard analogy method over the world if it is distributed worldwide to follow the world regulations.
Factors that Influence Software Project Cost and Schedule Estimation [14]	More research is required to study the level of application of the local software development standards in Sudan.
A New Approach by Using Tabu Search and Genetic Algorithms in Software Cost Estimation [15]	Reducing the execution time for the hybrid algorithm is the next step in this study.
Tuning of Software Cost Drivers using BAT Algorithm [16]	Considering other software engineering methods with different datasets can enhance the study purpose further.

- Unadjusted Use Case Weights (UUCW)
- Unadjusted Actor Weights (UAW)
- Technical Complexity Factor (TCF)
- Environmental Complexity Factor (ECF)

Software companies aim to provide high-quality software even before writing the first line of code. If not, unplanned costs will appear when poor-quality software delivered. Hence, estimating the high-quality software projects cost will decrease the future cost for a software company and customer as well. As we explored the original UCP method in the previous section, it is noticed that the thirteen technical factors (TFs) are partially considering the quality aspect.

TABLE II
QUALITY ATTRIBUTES, MEASURING STATEMENTS & WEIGHTS

	Quality Attributes	Weight	Measuring Statements		Effect Weights
1	Functional Suitability	2	1	Will the software function as per customer objectives?	+1
			2	The software gives correct and precise results as expected.	+1
2	Performance Efficiency	1	3	The software can function as expected by utilizing all available resources.	+1
			4	Will response time of a transaction in the software be less than 0.1 sec?	+1
			5	The software process less than the required number of operations as its maximum capacity	-1
3	Compatibility	1	6	Can the software integrate properly with other related systems?	+1
			7	The software distributes reliable information among multiple connected systems.	+1
4	Usability	1.5	8	The user interface of the software is easy to understand and use by a normal user.	+1
			9	The user-friendly software interface helps the user to be efficient and productive.	+1
			10	The contents of the software user interface are not reflex the process reality.	-1
			11	The software help center guides the user to perform processes by describing all functions.	+1
5	Reliability	1.5	12	Will the software able to deliver services whenever the user needs them?	+1
			13	Expected input errors by users are handled.	+1
			14	Expected hardware errors are handled.	+1
			15	Uptime of the software is more than ninety percent (90%).	+1
			16	When attacks occur, the software performance level is remaining as expected.	+1
6	Security	2	17	The software ensures a protected environment for data manipulation against attackers.	+1
			18	Authentication is provided by the software between source and destination ends.	+1
			19	Will the software block unauthorized access to system data?	+1
			20	No security technologies are applied to the software.	-1
7	Maintainability	1.5	21	Software source code is reusable to extend the functionality.	+1
			22	Will the software be implemented in modular-coding approach?	+1

			23	The structure of the software source code is neat and readable.	+1
			24	Software source code has weak documentation with unmanaged structure.	-1
			25	Will the software testing tools be available for the software lifetime?	+1
8	Portability	0.5	26	The ability to install the software system in the specified software/hardware environment	+1
			27	The software system cannot be adapted to other software/hardware environments.	-1
			28	The functional software source code is dependent on the hardware system.	-1

However, if we can extend these thirteen TFs, the extended factors will surely increase the precision of the estimation process. We named our new approach as Quality Extended Use Case Point (QUCP) method. Software quality means that the delivered software fits with its initial purpose that was given by the customer. Eight general quality attributes stated by the International Organization for Standardization (ISO) as ISO 25000 software product quality standard [9]. To understand what we meant by each ISO quality attribute, let us give a further explanation of each one of them. Functional suitability as a quality attribute ensures that the software functions correctly as per customer requirements. Security is one of the most important services that a customer needs in the software product. Security means that the data of the system are safe against attacks with authorized access to them. Authentication between communication ends is also maintained. Usability of the software aids the user to be productive since it ensures the easiness of understanding and using the user interface. The user guide centre is the ultimate helper to the software user in which it provides rich details about all functionalities of the software.

Fig. 1 ISO 25000 Software and Data Quality [9]

Reliability of software means that the software services are available whenever the user needs them. Especially, when there is a problem in the system, the system has a recovery plan in which all services are provided as usual. Maintainability of software means the source code is reusable in the software future to fix bugs, adapt it in new environments or simply to extend the functionality. Fast response is the ultimate need of the customer in which the software response time should be at the minimum whereas the transactions capacity should be at the maximum. The performance efficiency and powerfully deals with this issue. Compatibility of the software system with other systems becomes one of the important services to a special type of software such as enterprise resource planning (ERP) systems. In an ERP system, a group of integrated subsystems need to be compatible to exchange the exact information among all of them. Portability of software ensures that the software can run over independent environments of both hardware and software environments. Fig. 1 shows the software product quality attributes by ISO. Weights are judged and assigned according to the importance of each quality attribute. Table 2 shows the list of quality attributes with weights.

We generate measuring statements for each attribute in which to have a comprehensive list that describes the high quality of software. Each statement has effect weight that indicates the effect of the statement onto overall quality extended technical complexity factor (QTCF). This effect can be either positive (+1) or negative (-1) which will be multiplied by the score given by the estimator. The estimator will give a score between zero (means irrelevant factor) to five (means essential factor) to each measuring statement. Then, summing all statements weighted values produce the quality extended technical factor (QTF) that is used in eq. 1 to find out QTCF. Table 2 shows a detailed list of quality attributes with measuring statements and affect weights. The

original UCP technical factors list will be mapped into the new quality extended list. Table 3 shows each old TF with its corresponding QUCP measuring statements.

$$QTCF = 0.6 + (0.01 \times QTF) \quad (1)$$

TABLE III
MAPPING TCF OF UCP TO QUCP MEASURING STATEMENTS

Technical Factor (UCP)	Corresponding QUCP Measuring Statements
T1: Distributed systems	Compatibility (software integrates properly...)
T2: Specific performance objectives	Functional Suitability (function as per customer objectives...)
T3: End-user efficiency	Usability (user-friendly interface helps the user to be efficient...)
T4: Complex internal processing	Performance Efficiency (software process less than the required number of operations...)
T5: Code re-usability	Maintainability (code is reusable to extend the functionality)
T6: Easy to install	Portability (ability to install the software system in ...)
T7: Easy to use	Usability (interface of the software is easy to ...)
T8: Portability	Portability (software system cannot be adapted in other...)
T9: Easy to change	Maintainability (structure of the software source code is neat...)
T10: Concurrency	Compatibility (software distributes reliable information...)
T11: Security features	Security (no security technologies are applied ...)
T12: Access for third parties	Security (the software block unauthorized access ...)
T13: Requires special user training facilities	Usability (help center guides the user to perform processes ...)

V. VALIDATION

An online questionnaire, with twenty questions, was used to validate the proposed method goal achievement. The close ended questions were evaluated by the Information Technology (IT) experts of local and multinational companies. The questions should be responded using a likert scale ranging from one to five. MS Excel is used as an analysis tool. Table 4 shows a cumulative analysis of collected questionnaire responses. As stated, only 3% of the participants strongly disagreed that the QUCP method helps to produce more accurate software projects estimation by ensuring high-quality software delivered, where 7% of the participants disagreed upon their respective goal. 16% show a neutral response against our goal. On the other side, 34% have agreed that the QUCP method helps to generate more realistic estimation since the quality of the software product has ensured. This is the same for the remaining 41% since they strongly agreed upon the research goal. To sum up, 75% of the sample participants are agreed that the proposed QUCP method will help to produce more accurate estimation ensuring high-quality software will be delivered. The graphical representation of the cumulative analysis is shown in fig. 2.

TABLE IV
CUMULATIVE ANALYSIS OF THE PROPOSED SOLUTION

Q. No.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Q1	3.3	3.3	6.7	56.7	30
Q2	3.3	10	13.3	30	43.3
Q3	0	26.7	26.7	33.3	13.3
Q4	6.7	6.7	13.3	36.7	36.7
Q5	0	13.3	10	36.7	40
Q6	10	6.7	30	30	23.3
Q7	3.3	3.3	20	46.7	26.7
Q8	3.3	0	13.3	46.7	36.7
Q9	0	6.7	20	33.3	40
Q10	3.3	0	6.7	16.7	73.3
Q11	0	3.3	3.3	26.7	66.7
Q12	0	13.3	16.7	20	50
Q13	0	3.3	26.7	33.3	36.7
Q14	3.3	3.3	16.7	43.3	33.3
Q15	0	6.7	3.3	16.7	73.3
Q16	3.3	3.3	13.3	33.3	46.7

Q17	3.3	6.7	10	33.3	46.7
Q18	3.3	3.3	10	33.3	50
Q19	3.3	3.3	20	33.3	40
Q20	6.7	6.7	30	36.7	20
Total	56.7	130	310	676.7	826.7
Avg.	2.8%	6.5%	15.5%	33.8%	41.3%

Fig. 2 The cumulative analysis of four goals

VI. CONCLUSION AND FUTURE WORK

An improved cost estimation method as an extended method to the traditional UCP method is proposed in this research. Quality Extended Use Case Point (QUCP) method considers the quality attributes of the software to generate more realistic estimation in the planning phase of the system development life cycle. A questionnaire technique is used to validate the proposed method goal achievement. As a result, seventy-five percent of the participants confirm that the proposed QUCP method helps to produce more accurate estimation ensuring high-quality software will be delivered. As future work, further validation of the QUCP method is recommended via industrial case studies in which the QUCP method is used to estimate the cost of diverse range of software projects to generalize the results. The QUCP based cost estimations should be compared to the actual costs of projects and the error difference should be examined.

ACKNOWLEDGEMENT

Firstly, I am extremely thankful to Allah who enabled and blessed me to conduct this research. Secondly, I would like to give special gratitude to my parents. Thirdly, I am grateful to my supervisor Prof. Dr. Rizwan J. Qureshi who helped me step by step to accomplish this research. Last but not least, my special tribute to all those people who participated in the survey.

REFERENCES

- [1] A. Banimustafa, "Predicting Software Effort Estimation Using Machine Learning Techniques," in *Proc. Comp Sci and IT*, 2018, p. 249-256.
- [2] J. Liu, Q. Du, and J. Xu, "A Learning-Based Adjustment Model with Genetic Algorithm of Function Point Estimation," in *Proc. High Performance Comp and Comm Smart City*, 2018, p. 51-58.
- [3] K. Park, SY. Moon, and RYC. Kim, "Improving Use Case Point (UCP) Based on Function Point (FP) Mechanism," in *Proc. Platform Tech. and Service*, 2016, p. 1-5.
- [4] S. Bagheri, and A. Shameli-Sendi, "Software Project Estimation Using Improved Use Case Point," in *Proc. SE Res, Manag and App*, 2018, p. 143-150.
- [5] S. Garg, and D. Gupta, "PCA based cost estimation model for agile software development projects," in *Proc. Industrial Engineering and Operations Management*, 2015, p. 1-7.
- [6] A. Hira, and B. Boehm, "Combatting Use Case Points' Limitations with COCOMO(R) II and Relative Difficulty," in *Proc. Asia-Pacific SE*, 2016, p. 353-356.
- [7] K. Qi, and B. Boehm, "A Light-Weight Incremental Effort Estimation Model for Use Case Driven Projects," in *Proc. Annual SW Tech*, 2017, p. 1-8.
- [8] K. Qi, and B. Boehm, "Detailed Use Case Points (DUCPs): A Size Metric Automatically Countable from Sequence and Class Diagrams," in *Proc. Workshop Modelling SE*, 2018, p. 17-24.
- [9] (2021) ISO 25000 Software and Data Quality website. [Online]. Available: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&limitstart=0>
- [10] M. Owais, and R. Ramakishore, "Effort, duration and cost estimation in agile software development," in *Proc. Contemporary Computing (IC3)*, 2016, p. 1-5.

- [11] F. Fellir, K. Nafil, and R. Touahni, "Analyzing the non-functional requirements to improve accuracy of software effort estimation through Case Based Reasoning," in *Proc. Intelligent Sys Theories and App*, 2015, p. 1-6.
- [12] S. Ramacharan, and KVG. Rao, "Scheduling Based Cost Estimation Model: An Effective Empirical Approach for GSD Project," in *Proc. Wireless and Optical Comm Networks*, 2016, p. 1-5.
- [13] I. Kurniawan, AA. Arman, and S. Mardiyanto, "Development of Analogy-Based Estimation Method for Software Development Cost Estimation in Government Agencies," in *Proc. Electrical Engineering and Informatics (ICEEI)*, 2017, p. 1-6.
- [14] SMA. Suliman, and G. Kadoda, "Factors that Influence Software Project Cost and Schedule Estimation," in *Proc. Computer Science and Information Technology (SCCSIT)*, 2017, p. 1-9.
- [15] FS. Gharehchopogh, R. Rezaii, and B. Arasteh, "A New Approach by Using Tabu Search and Genetic Algorithms in Software Cost Estimation," in *Proc. Application of Information and Communication Technologies (AICT)*, 2015, p. 113-117.
- [16] S. Girotra, and K. Sharma, "Tuning of Software Cost Drivers using BAT Algorithm", in *Proc. Computing for Sustainable Global Development*, 2016, p. 1051-1056.
- [17] G. Karner, *Resource estimation for objectory projects*, Objective Systems SF AB 17, 1993.