Process Models and Estimation Methods in Cloud, Artificial Intelligence and Mobile Software Development Scenarios: A Systematic Review

Sadhana Pandey
Department of Computer Science and Engineering,
Sage University
Indore, India

Abhay Kothari, PhD
Department of Computer Science and Engineering,
Sage University
Indore, India

ABSTRACT
The software development life cycle (SDLC) provides a systematic framework with specific deliverables at each phase of the software development. As the characteristics of each categories of software are different in many ways, it becomes crucial to study process model for such categories of software separately. Cost estimation is the fundamental area that chooses budgetary constraints related to these software applications which keep company to maintain accurate estimates for such application development to maintain their reputation in the market. In this research paper we have studied process model and estimation techniques for Artificial Intelligence application, cloud application and Mobile application in particular. The process model for artificial intelligence, cloud, mobile application have been studied and presented separately. We have also studied basic COCOMO model, intermediate COCOMO model as a fundamental work in cost estimation.

General Terms
Artificial Intelligence (AI) Application development process Model, COCOMO, Process Model, Cloud Application Development Process, Mobile Application development Process, Software Development Life Cycle (SDLC).

Keywords
Artificial Intelligence (AI) Application development process Model, COCOMO, Process Model, Cloud Application Development Process, Mobile Application development Process, Software Development Life Cycle (SDLC), Cloud Software Life Cycle Process (CSLCP) Model.

1. INTRODUCTION
The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible time. The goal of the SDLC is to produce superior software that meets all customer expectations and demands. The SDLC outlines a detailed plan with stages, or phases, that each encompasses their own process and deliverables. Processes in software engineering refers to the methods and techniques that are used to develop and maintain software. In software engineering, adaptability refers to a system's ability to adjust to changing requirements, technologies, and environments, without requiring much modifications or re-implementation. ISO/IEC/IEEE 12207 mainly provides processes that can be employed for defining, controlling, and improving software life cycle processes within an organization or a project. ISO//IEC/I/EEE 15288 document defines a set of processes to facilitate system development and information exchange among acquirers, suppliers, and other stakeholders in the life cycle of a system.

We have studied process model for AI, cloud and mobile application. The application of artificial-intelligence-(AI)-based methods within the context of complex systems presents new challenges within the product life cycle. The general approach is to build a component-wise development of the overall system including an AI component. This allows domain specific development processes to be in parallel fashion.

This can be summarized in the following two challenges for the use of ML methods in the development process of complex technical systems:

1. The ML component functionalities heavily depends on data from operating components and
2. The performance of ML components cannot be predicted in advance in such scenarios.

Popular and well-established process models like the Waterfall Model, Scrum, Crisp-DM and the V-Model do not address these challenges.

On the other hand, Small to medium-sized enterprises take advantage of the strengths of cloud computing. These enterprises require a software process model to produce reliable and quality cloud software, given their limited resources.

Different types of software need a different type of software process models. The selection of an unsuitable software process model results in insufficient software quality and increases its development Cost. The software engineering process should be tailored to cloud-based projects. Most of the research in the area of software process models for cloud-based applications only considers the development process area. In these models, attention is not paid to the characteristics, challenges, and standards of the cloud-computing environment [2]. In the same context mobile application development process is also different as requirements vary considerably. Mobile devices are characterized as a portable device, viewed as a personal device by its users, and require a network connection. So we have studied process model for each category of software and figured out their differentiating characteristics. We have also focused on estimation model of such software applications which can be used for all the software applications under our study and are used by software companies frequently.

2. LITERATURE REVIEW
This paper [1] seeks to explore six common methodologies: the Waterfall Model, the Spiral Model, Agile, Scrum, Kanban,and Extreme Programming. Specifically, a general explanation of each methodology, the history behind it, its unique features,

and general developer opinions of the methodology is discussed.

In this paper [2], the author A cloud software life cycle process model proposed, validated, and verified to handle the shortcomings of existing cloud software process models. A case study is used to illustrate all the activities required throughout the software life cycle of the discussed model. The proposed cloud software life cycle process model is a cyclic iterative prototyping model. It is compatible with levels two and three of the capability maturity model integration and extends the Egyptian software process improvement model to fit the cloud environment so far.

Software metrics offer a quantitative basis for predicting the software development process. In this paper the author [3] discussed the different software metrics and how these metrics have an impact on software quality and reliability. Effort estimation is an integral part of software project management. There is a lot of research in effort estimation in the form of models, techniques, methods, and tools. Although a variety of estimation techniques have been applied in an ASD context, ranging from expert judgment to Artificial Intelligence techniques, those used the most are the techniques based on some form of expert-based subjective assessment. These techniques are expert judgment, planning poker, and use case points method.[4] Most of the techniques could not result in good prediction accuracy values. It was observed that there is little agreement on suitable cost drivers for ASD projects. Story points and use case points, which are close to prevalent requirement specifications methods (i.e. user stories, use cases), are the most frequently used and considered size metrics. The author [5] used Clustering approach to form consistent project groups and Support Vector Regression (SVR) to predict the effort required for development. In this paper[6] the author worked on cost estimation, fuzzy logic and applied with three membership functions such as Triangular, Trepezoidal and Bell. This work has been tested on 5 projects. When the parameters for triangular function are executed, values for embedded 1, semidetached 2 are enhanced as compare to organic, semi detached 1, embedded 2. After the evaluation of trapezoidal function, values of semidetached 1 and embedded 1 are more as compare to others. The results show that all three membership functions vary by 1-2 % in the effort estimation where as if compared; COCOMO 2 Effort varies by 5-10 % from Fuzzy output. In paper[7], the author has reviewed various papers on COCOMO model. In this paper, the author[8] has studied the COCOMO'81 model, specifically its intermediate version and proposed the use of fuzzy sets rather than classical intervals in the COCOMO'81 model specifically. For each cost driver and its associated linguistic values, they have defined the corresponding fuzzy sets. In this paper [9], comparison of deepnet, neuralnet, support vector machine and random forest algorithms were carried out and the results show that random forest outperforms other algorithms because of its robustness and capacity to handle large datasets. Evaluation metrics deliberated discussed are Mean Absolute Error, Root Mean Squared Error, Mean Square Error and R-Squared. The author [10] studied the existing techniques in term of accuracy, usage, and suitability. Therefore, their comparison could facilitate and help the project managers to distinguish and compare among techniques to choose the optimum technique as per project type and requirements. Also, they have proposed a model to help a researchers, and project managers, by combining three existing estimation techniques to improve the accuracy. The challenges that are being anticipated, and covered by using the proposed model include the errors that can result due to the single approach failure. An accuracy in software cost estimation has a direct impact on company's reputation and also affects the software investment decisions in the long run. Accurate cost estimation can minimize the unnecessary costs and increase the productivity and efficiency of the company considerably. The author[11], identified the existing methods of software cost estimation prevailing in the market and analyzing some of the important factors impacting the software cost estimation process. In the paper[12], presents a systematic survey about software cost estimation in agile software development. The paper deals with the current estimation schemes used in software development other than agile estimation, so that these schemes may be useful in the agile development environment. In this paper[13], The main objective of this paper is to provide an overview of software cost estimation models and summarize their strengths, weakness, accuracy, amount of data needed, and validation techniques used. The findings show, in general, neural network based models outperforms other cost estimation techniques. In [14] presents the design and implementation of a software cost estimation tool integrated into a mobile application developed using Flutter. This tool includes various techniques for software cost estimation, including expert judgment, function point analysis, 3D point analysis, and the COCOMO model. The tool's efficacy is assessed using case studies and contrasts with other software cost estimation methods currently in use. The outcomes show that the app can produce trustworthy and precise cost estimates, which makes it an important resource for software development projects. [15] The work carried out in this paper contributes to the field of software engineering to calculate the overall efforts of mobile application development. research work uses a hybrid approach using the concepts of CPEEM and Machine learning technique. The framework uses Mobile Functional Factors as input parameters for the proposed approach where CPEEM and Machine learning technique can be used to calculate the size and efforts of mobile application development. These efforts are compared with the real mobile applications' actual efforts to see whether the proposed approach is efficient or not.

In this paper [16], research study compares various machine learning techniques for estimating effort in software development, focusing on the most widely used and recent methods. Random Forest Regression algorithm performs well on the given dataset tested along with various Regression algorithms, including Support Vector, Linear, and Decision Tree Regression. In this paper [17], The main objective of this research is to investigate the role of fuzzy logic technique in improving the effort estimation accuracy using COCOMO II by characterizing inputs parameters using Gaussian, trapezoidal and triangular membership functions and comparing their results. NASA (93) dataset is used in the evaluation of the proposed Fuzzy Logic COCOMO II. It is found that Gaussian function is performing better than trapezoidal function and triangular function, as it demonstrates a smoother transition in its intervals, and the achieved results were closer to the actual effort. In this paper [18] investigation started with a survey that targeted software professionals, and then they conducted multiple-case study approach involving four different software development companies in Palestine. Expert-based estimation models are the mostly applied models especially within agile environments. Multiple improvements were required to be done on expert-based models to formalize the process. In this paper [19] a simplified genetic algorithm based model is proposed. A simplified genetic algorithm is used for optimizing the parameters of the basic COCOMO model. The author found COCOMO with simplified GA tuned parameters gives an improved estimation compared to basic

COCOMO. [20] In this paper, different reviews are made clear to propose a way cosmic an appropriate method that can be used to size mobile application in a fast and accurate way. The review paper has provided a layout of different size matrices and cost estimation models. In [21] the author study aims to present a systematic literature review (SLR) to investigate the trends of the articles published in the recent one and a half decades and to propose a way forward in this domain. This systematic literature review has proposed a three-stage approach to plan (Tollgate approach), conduct (Likert type scale), and report the results from five renowned digital libraries. It is concluded that ANN; and COCOMO are the most popular techniques followed by Ensemble and FPA. Also, ANN has outperformed several ML and Non-ML techniques.

The paper [22] aims to explore project management activities and techniques for estimating project size. Overall, software project management involves managing, allocating, and timing resources to develop software that meets requirements and required to be delivered within budget and schedule. This paper highlights the significance of employing efficient estimation methods to achieve successful software project management. Estimation plays a critical role in the software development process, as it helps project managers to determine the resources and time required for completing the project .In [23], findings

Shows in general, neural network based models outperforms other cost estimation techniques. However, no one technique fits every problem and they recommended practitioners to search for the model that best fit their needs. In this paper[24] the author concluded that An effective development model can help improve competitive advantage and shorten release cycles, which is vital in the fast paced environment of mobile app development. In this paper[25] the author has explained Process model for AI Systems Engineering (PAISE®) contributes to the aim of integrating AI development into development contexts of increasingly complex systems

3. IDENTIFIED CHARACTERISTICS

As each software application is different in some way or the other so, we have identified characteristics of each category of software under the study. Our aim is to study the differences and similarities of these categories of software and to assess how these impact their development process as well as respective estimation techniques. The importance of finding a systematic approach for the development of intelligent systems making use of novel AI and machine learning methods is widely recognized. So the identified characteristics of AI software application are adaptability, perception, computer vision and machine learning.

On the other hand, the cloud applications also differ from other categories of software. The cloud based applications require on demand self service, broad network access, resource polling, measured service, elastic scalability and ubiquitous access. On the other hand, if we talk about software application that run on mobile device, it demands a well designed user interface, cross device compatibility, security, live streaming and some essential offline functionality. Distinguishing aspects for mobile application are network connectivity concerns, hardware limitations (e.g., screen sizes and battery power), portability, reliance on sensors for many applications, user movement across multiple locations.

Here it is crucial to note that some of the characteristics may be required in all categories of software applications and some of them may differ for each categories of application. so it becomes obvious to study their process models separately

4. PROCESS MODEL FOR AI

The application of artificial-intelligence-(AI)-based methods within the context of complex systems poses new challenges within the product life cycle considerably. The process model for AI systems engineering, PAISE®, addresses these challenges by combining approaches from the disciplines of systems engineering, software development and data science. Now a days, Machine learning (ML) algorithms are advancing to the practical forefront as a subset of artificial intelligence (AI). The ML algorithm programs a software for a given use case by analyzing so-called training data and identifying patterns and correlations. The functions of the developed software are therefore largely determined by the training data. The above explanations must be concluded in the following two challenges for the use of ML methods in the development process of complex technical systems:

1. The ML component functionalities essentially depend on data from operating components and
2. The performance of ML components is not predictable in advance in such cases.

Popular and well-established process models like the Waterfall Model, Scrum, Crisp-DM and the V-Model do not address

these challenges .The Waterfall Model, originally defined for the domain of software development in 1970, consists of a fixed number of phases that run through in a predefined sequence with clearly pre-defined results. While this process model supports good time planning during development, it actually lacks iterative elements that allow an explorative approach. So, challenge 2 is not addressed. For ML-components compatibility severely depends on the quality of the data that was used during the component's development. Thus, challenge 1 is met neither by the V-Model nor Scrum.

Standards such as ISO 12207 and ISO/IEC 15288 are paving the way for the development of increasingly complex systems. The standard ISO/IEC 15288 describes the life cycle processes of a system developed according to the established disciplines of systems engineering and software engineering.

The Process Model for AI Systems Engineering views the development of a product as a system that could be decomposed into subsystems which can be either software (e. g., ML algorithms) or hardware (e. g., mechanical parts).

The process model consists of seven phases and these are arranged in a waterfall-like structure. As We can see in figure 1, The first two phases, Goals & Problem Specification and Requirements & Solution Approaches, adopt the processes "Business or Mission Analysis Process", the "Stakeholder Needs & Requirements Definition Process" and the "System Requirements Definition Process" from the standard ISO/IEC 15288. Overall project goals are defined, product requirements are derived and first ideas of how to approach the problem are developed till here. The artifact of role distribution is initialized during the phase Requirements & Problem Understanding to have a clear distribution of responsibilities. Functional Decomposition must initially be specified including their interfaces. The phase Handover covers the "Transition process" from the standard ISO/IEC 15288 where the product is required to be transferred from the development team to the organizational units that realize operation and maintenance. the characteristics of the phases Functional Decomposition, Development Cycle and Operation & Maintenance are according to the application of AI methods in addition to general aspects from the "Design Definition Process", "Implementation Process", "Validation and Verification

Process" and "System Analysis Process" from standard ISO/IEC 15288.

During the phase of Functional Decomposition, the functions of the overall system are initially distributed onto subsystems. The result is hierarchical subsystem specification with well-defined interfaces in most of the cases.

In the context of AI systems engineering, it is essential to incorporate data sources into the system model. Data sources are considered to be subsystems or enabling systems that provide data for development and/or for operation thus significantly influence the functionality of AI components.

An iterative development cycle is supported by checkpoints to synchronize component development. It allows switching between an exploratory approach on the one hand and a goal-oriented approach on the other. By iterating through the cycle, the maturity of the components and that of the overall system is continuously increased. At checkpoints the (partial) integration and evaluation of components with respect to requirements is done. Checkpoints therefore serve as a point of synchronization of all components. For ML components, this means evaluation against validation metrics that commonly assess the component's functioning within the overall system.

The data provisioning procedure has the purpose to generate, prepare and evaluate training, test and validation datasets. The data requirements comprise technical aspects relevant to the accomplishment of the AI component's tasks or purpose. Examples are the amount of data (how many measurements are available), its quality (e. g., how much missing or incorrect information) and its representativeness (whether the training data represents the data that will be generated at runtime). The data provisioning procedure is based on the V-Model. Each development step has a testing and verification step at the same level of detail. The procedure for ML component development is based on the V-Model as well. The goal of the procedure is the encapsulation of an ML model (i. e., the data-driven part) into a component. This facilitates the substitution of data sources and related enabling systems in order to be able to iteratively integrate and validate results within the checkpoints.

This approach creates an organizational interface between the classical data science discipline and systems engineering in particular. Domain knowledge is required to be incorporated at this point in order to ensure the correct functionality of the component.

Afterwards, the selected ML method is implemented as a specific model architecture with well defined hyperparameters. Examples of hyperparameters are the number of neurons, layers in artificial neural networks, learning rate as well as the definition of the loss function, also denoted as local cost function. As soon as all requirements are met, the exit of the development cycle is triggered and the last two phases Handover and Operation & Maintenance follow.

The monitoring of the ML component functionalities is essential for reliable operation of AI-based systems. Changes in the data processed during operation can degrade the performance of AI subsystems over time. If a model update is required, a new training data set is collected and processed following the data provisioning procedure. In the next step, the ML component development procedure is applied in order to re-adjust the ML model. Finally, the updated component is reintegrated into the overall system, tested and must put into operation.

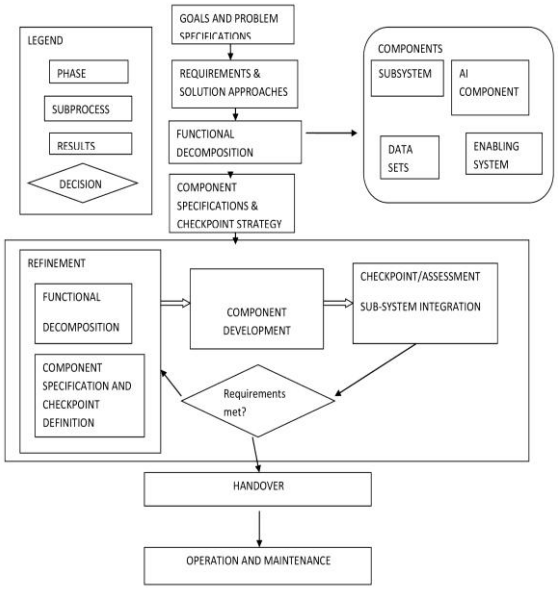

Figure 1: The Overall Structure Of The Process Model For AI System Engineering

Process model for cloud application

CSLCP (A cloud software life cycle process (CSLCP) model) model takes into consideration the risks and challenges, including the standards of cloud computing. Hence, the CSLCP model enhances the quality of cloud software development. The CSLCP model conforms to the quality software process model (2.2). CSLCP model consists of eight process areas of the second and third levels of the CMMI (Capability Maturity Model Integration) and nine sequential phases.

The nine phases are exploration, alternatives and decision making, planning, analysis, design, implementation, deployment, maintenance, and retirement phase. The description of these phases is as follows.

Exploration phase. The goal is to explore the development environment. That is to have a comprehensive view of the project's givens and to elicit the initial requirements. The collected information will be used in the subsequent phases; for planning and analysis of the project and making the needed decisions to develop the software.

Alternatives and the decision-making phase. The development in a cloud environment involves many crucial decisions. The aim of this phase are defining the development alternatives, studying them, and making suitable development decisions. These decisions are to be used in the following phases.

Planning Phase. The aim of this phase is to plan for all the development tasks in all the process areas. The planning phase includes estimating the needed resources and the size and skills of the development team in advance. It involves preparing the development schedule, the review plan, configuration management plan, and quality assurance plan.

Analysis phase. The aim of this phase is to analyze and specify both the elicited functional and non-functional requirements. In addition to that, the requirements are transformed into high level architecture components.

Design phase. In this phase, the analyzed requirements are converted to detailed architectural components representing the

functional behaviors of the software to be developed.

Implementation phase. The aim of this phase is to begin coding the designed artifacts with a programming language that is supported by the cloud provider and to test the completed artifacts subsequently.

Deployment phase. The phase has several goals; deploy the developed artifacts on the operating platform of the cloud provider to be used by the end-user, and to specify and provide the SLA of the developed SaaS/FaaS.

Maintenance phase. if the development team could handle the arising issues, then the issues are cycledback to the analysis phase

Retirement phase. The needs and the business environment change rapidly. Consequently, in some cases, some software components and features are no longer needed. Therefore, the development team is getting rid of any software component that is no longer needed.

The eight process areas of the CSLCP model are project management(PM), requirements (R), product development (PD), risk management (RM), verification ‘‘peer review” (V), quality assurance (A),configuration management (CM), and security (S).

Applying a structured process model such as the CSLCP model can help SMEs to develop any type of cloud-based software at low cost, good quality. The CSLCP model described and specified what to do to achieve this goal.

Process model for Mobile application

Mobile software development consist of clear goals and practices in order to be successful, however, this kind of software bears several limitations not present in desktop computing that make the mobile ecosystem a particular environment.

For instance, wireless communication problems (availability, variability, intermittence), mobility issues (autonomy, localization), the variety of platforms and technologies, the limited capabilities of terminal devices (low power supplies, small-sized user interfaces), and strict time-to-market requirements.

Mobile-D was the first attempt to incorporate Agile for the development of mobile applications. Mobile-D was introduced in 2004 by Abrahamsson et al. [26] as a development methodology inspired on Extreme Programming, Crystal Methodologies and Rational Unified Process (RUP). It is recommended to be used by a small, co-located team, working in a short development cycle. Mobile-D encourages iterations, after which a functional product is created. Actual Agile activities within the methodology include: Test-Driven Development, Continuous Integration, Pair Programming, etc.

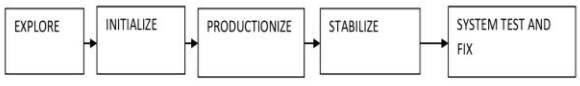

Figure 2 Phases of Mobile-D Software Development Process

5. ESTIMATION MODEL FOR AI, CLOUD AND MOBILE APPLICATION

As per review on estimation techniques of these above mentioned categories of software, basic COCOMO were found to be used in initial stage of projects and later intermediate COCOMO would be used to estimate cost and effort in such scenarios.

COCOMO Models (Constructive Cost Model)

This family of models was proposed by Boehm. The models have been widely accepted in practice. In COCOMO, the size S code is given in thousands of LOC (KLOC) and the effort is in person-month.

1. *Basic COCOMO* : In this model based on software complexity, three sets of {a, b} are used.
   - For simple, well-understood applications, the values are a = 2.4, b = 1.05
   - For more complex systems, a = 3.0, b = 1.15
   - For embedded systems, a = 3.6, b = 1.20
2. *COCOMO Intermediate And Detailed COCOMO* : In the intermediate COCOMO, an estimate of the nominal effort is obtained by using the power function with three sets of {a, b} with coefficients 'a' that are slightly different from that of the basic COCOMO.
   - For simple applications, a = 3.2, b = 1.05
   - For more complex systems, a = 3.0, b = 1.15
   - For embedded systems, a = 2.8, b = 1.20

$$PM = A \times Size^{E} \times \prod_{i=1} EM_i$$

$$\text{Where, } E = B + 0.01 \times \Sigma_{i=1} SF_i$$

The size of the application must be scaled according to the following five scale factors:

- Precedence (PREC)
- Flexibility of development (FLEX)
- Architecture / Risk Resolution (RESL)
- Team cohesion (TEAM)
- Maturity of the process (PMAT)

Cost factors are the characteristics of software development that affect the execution of a project considerably. Unlike scale factors, cost factors are chosen based on their fundamental principles of linear impact on effort. The main components of the project costs include:

- Effort costs
- Travel and training expenses.
- Hardware costs

Among the components of project costs, labor costs are the most difficult to estimate and control the administration costs and have the most significant impact on total costs.

6. CONCLUSION

In this paper, we have studied characteristics of three categories of software namely artificial intelligence, cloud and mobile applications. We have also reviewed process models as well as the estimation methods for these kinds of software projects. Since the characteristics are different they cast their effect on process model and estimation methods. As we can see in the process model for AI, functional decomposition may be

required repeatedly but in the case of cloud application development it is not required as much. In the same way, if we compare AI application to that of cloud application it has been observed that SLA (service level agrement) activities are required in only cloud application and not in AI application. Therefore, we can conclude that some activities may be mandatory in one kind of application and may be optional for other kind of application development. So the study of process model for above mentioned application development leads to minimized version of process model which would be our future work.

7. REFERENCES

[1] Risener, K. (2022). A Study of Software Development Methodologies. Computer Science and Computer Engineering Undergraduate

[2] Amira A. Alshazly, Mustafa Y. ElNainay , Adel A. El-Zoghabi, Mohamed S. Abougabal, (2020) Ain Shams Engineering Journal. A cloud software life cycle process (CSLCP) model.(6).

[3] J.Rashid, 2 T.Mahmood, 3 M.W.Nisar. *Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan.* A Study on Software Metrics and its Impact onSoftware Quality. *2313-7770.*

[4] Usman, M., Mendes, E., Weidt, F., Britto, R. (2014) Effort estimation in agile software development: a systematic literature review. In: *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*(pp. 82-91).

[5] *Ekrem Kocaguneli, Ayse Tosun, Ayse Bener.* AI-Based Models for Software Effort Estimation. Conference Paper *in* Conference Proceedings of the EUROMICRO · September 2010.

[6] Rahul Kumar Yadav and S. Niranjan.(2017) Indian Journal of Science and Technology, Project Effort Estimation using COCOMO-2 Metrics with Fuzzy Logic. *Vol 10(29),*

[7] Gajender pal, Manish kumar, Kuldeep barala. IJRDO - Journal of Computer Science and Engineering. A review paper on cocomo model. ISSN: 2456-1843

[8] *ALI IDRI AND ALAIN ABRAN Laïla KJIRI.(2000).* Cocomo Cost Model Using Fuzzy Logic. *7th International Conference on Fuzzy Theory &Technoloy* Atlantic City, New Jersey, February 27 – March 3, 2000

[9] A G Priya Varshini *et al* 2021 *J. Phys.: Conf. Ser.* 1767 012019. .

[10] Mohammed Aljohani and Rizwan Qureshi.(2017). International Journal of Software Engineering & Applications (IJSEA), Vol.8, No.6. COMPARATIVE STUDY OF SOFTWARE ESTIMATION TECHNIQUES.

[11] Gangwani, D. and Mukherjee, S. (2015) Analyzing the Impact of Different Factors on Software Cost Estimation in Today's Scenario. *Journal of Software Engineering and Applications*, 8, 245-251

[12] Saurabh Bilgaiyan, Santwana Sagnika, Samaresh Mishra and Madhabananda Das. Journal of Engineering Science and Technology Review · August 2017. A Systematic Review on Software Cost Estimation in Agile Software Development

[13] Chirra, S.M.R. and Reza, H. (2019) A Survey on Software Cost Estimation Techniques. Journal of Software Engineering and Applications , 12, 226-248

[14] Jaiswal A, Malviya P, Parihar L, et al. Software cost estimation tool: A App based application, estimate the cost of software project. Computing and Artificial Intelligence. 2024; 2(2): 1364

[15] Ziema Mushtaq1,*, Sami Alshmrany2, Fahad Alturise3, and Tamim Alkhalifah. EAI Endorsed Transactions on Energy Web. Early Size and Effort Estimation of Mobile Application Development

[16] KRUTI LAVINGIA ∗, RAJ PATEL †, VIVEK PATEL ‡, AND AMI LAVINGIA.(2024). Scalable Computing: Practice and Experience, ISSN 1895-1767. SOFTWARE EFFORT ESTIMATION USING MACHINE LEARNING ALGORITHMS. Volume 25, Issues 2, pp. 1276–1285

[17] Ashita Malik, Varun Pandey, Anupama Kaushik. *I.J. Intelligent Systems and Applications,* 2013, 05, 68-75. An Analysis of Fuzzy Approaches for COCOMO II

[18] Zarour, A. & Zein, S. (2019). Software development estimation techniques in industrial contexts: An exploratory multiple case-study. *International Journal of Technology in Education and Science (IJTES), 3*(2), 72-84.

[19] Rohit Kumar Sachan∗, Ayush Nigam, Avinash Singh, Sharad Singh, Manjeet Choudhary, Avinash Tiwari and Dharmender Singh Kushwaha.(2016). Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016). Optimizing Basic COCOMO Model using Simplified Genetic Algorithm.

[20] Ziema Mushtaq, Abdul Wahid.(2018). IOSR Journal of Engineering (IOSRJEN) ISSN (e): 2250-3021, ISSN (p): 2278-8719. Cost Estimation for Mobile Application Development: Review.

[21] Chaudhary hamza rashid 1, imran shafi 2, jamil ahmad.(2023).IEEE ACCESS. Software Cost and Effort Estimation: Current Approaches and Future Trends. *Digital Object Identifier 10.1109/ACCESS.2023.3312716.*

[22] Khin shin thant, hlaing htake khaung TIN (2023). Innovare Journal of Engineering and Technology. Learning the efficient estimation techniques for successful software project management. Vol 11, 2023

[23] Chirra, S.M.R. and Reza, H. (2019) A Survey on Software Cost Estimation Techniques. Journal of Software Engineering and Applications , 12, 226-248.

[24] Ronald Jabangwe [a], Henry Edison [b], Anh Nguyen Du.(2018). Journal of Systems and Software Volume 145**,** November 2018, Pages 98-111. Software engineering process models for mobile app development: A systematic literature review.

[25] Constanze Hasterok and Janina Stompe.(2022). Automatisierungstechnik 2022; 70(9): 777–786. PAISE® – Process model for AI systems engineering

[26] Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M. Koskela, J., Kyllönen, P., Salo, O.: Mobile-D: An Agile approach for mobile application