A REVIEW ON COST ESTIMATION TOOLS FOR SOFTWARE DEVELOPMENT

Mohammed Rafi
Research Scholar
MUIT University
Lucknow

Dr. B.D.K.Patro
Professor
MUIT University
Lucknow

ABSTRACT

Prediction for certain times of duration and costs for each development project are still a problem in software industry. The estimate is an essential activity for managing successful software projects. This is very necessary for budget and project planning. The quality of the estimate is one of the factors that determine the success of the project and facilitates the elimination of risks. Inaccurate estimates often lead to project failure. In general, the results of the budget and delivery times are an essential concern for the software industry. In the previous research, much software effort, duration and cost estimation methods are available including the Algorithmic and Non- algorithmic model. In the initial stage, software cost estimation was complete manually by using simple thumb rules or estimating algorithms. In below taking review from different authors in order to understand research gap.

Keywords: *Cost estimation, prediction, effort, machine learning*

I. INTRODUCTION

Software Technology is a subject deeply worried about the evolution of big, well-organized software programs utilized in electronic computers. Including various preparation theories and methods that include not just scientific questions, like tools and technology for trained applications development but also administrative aspects like planning, staffing, business, preparation, tracking, and budgeting. Software technology is a complex procedure which simplifies the issue by producing an abstract design or perhaps connected to an iconic version. In comparison to science, SE closely connected with great small business clients. This well-known scientific technique of software development required user specifications once the input of a software product published as an outcome.

This Systematic and planned advancement in applications expansion finished with a sequence the function-based development version, various hybrid versions, object-oriented Models, and distinct prototype versions. The ordered set of actions Included extending software preparation mentioned as a process. Every stage of SDLC is a growth which needs limitations, resources and other jobs. They also inflict an arrangement and assist developers enhance associated exercises.

The degrees have a social event that's essential for a customer, particular practices point by point on what the plan does and when all of the client's requirements have fulfilled. After the customer recognizes the precious details dependent on the conclusion of this case, the item is made and analyzed.

The software built can be moved and introduced in the close of the purchaser.

The vital actions of a software procedure comprise the group of software requirements, software design, software construction, software testing, software maintenance, software configuration management, software engineering management, software engineering process, applications and software engineering procedures, software quality assurance and risk management.

Subsequent is the developmental narration of software cost estimation

1940-It was the start of the PC period and towards the digitalization of data.

Before 1970-Simple dependable guidelines and some basic calculations of estimation was utilized and henceforth the mistake rate in the forecast was high.

Around 1970-Researchers pondered the mechanization apparatuses for increasingly exact and quick expectation of expense and amid this period, the principal robotized machine for software cost opinion was fabricated. It was presented by Barry Boehm, named as COCOMO (Constructive Cost Model), and it was given in the book Software Engineering Economics by Bohem.

1977-Frank Freiman created PRICE-S business device for the United States for business purposes.

1979-Lawrence H. Putnam presented the SLIM (software life-cycle the executives)-second business instrument for the United States

1981-Barry Bohem underscored some valuable calculations for COCOMO display.

1982-In the book "controlling software ventures", Tom DeMarco presented an utilitarian measurement with some acquired highlights of Albrecht's capacity point metric.

1983-Ada software dialect was produced by the U.S. Branch of Defense (DoD) for decreasing the expense of growing expansive military software frameworks. Additionally, around the same time, check II work point metric was presented by Charles Symons.

1984-IBM completed a noteworthy correction in its capacity point metric.

1985-Capers Jones with his associates built up the SPQR/20 (software item quality and unwavering quality) device for estimation. Likewise, the idea of Function Point was reached out by Caper Jones, to incorporate the impact of computationally troublesome issues.

1990-Barry Boehm, at University of Southern California, began broadening and amending the current COCOMO demonstrate.

1991-Hans Koolen and Michel van Genuchten added as far as possible of cost estimation by presenting some of valuable apparatuses and techniques for software cost estimation.

1992-R. Betteridge utilized the stamp II work guide's measurements toward anticipate the expense of a few activities.

1993-New form of the COCOMO display which is COCOMO 2.0 showed up in 1994.

1997-Some of the current models were reexamined for expanding their precision rate.

1998 another model called MARCS was presented by Chatzoglou, for expectations of software exertion.

Around 2008-Caper Jones drove a few patent applications for presenting another technique for rapid software measuring strategy dependent on example coordinating. The benefit of the strategy is its relevance before necessities. This new technique is inserted with another cost estimation instrument called Software Risk Master (SRM).

II. REVIEW OF LITERATURE

Aprna Tripathi ; et.al [2012] Estimating software maintenance effort has always been a challenge for software professionals because it accounts for about half of overall development costs, for systematic and timely forecasting of the software maintenance effort using the SRS (Software Requirements) document of the proposed software. The result confirms that the proposed maintenance measure is complete and compares well with the various other measures in effect proposed in the past. The calculation of the estimate of the proposed maintenance effort implies a minimum of overhead costs compared to the others.

Ivan Tsmots et.al [2016] Requirements and design phases of hardware and software intellectual processing of intensive data streams in real time have been examined. Design principles of hardware and software tools have been proposed. Translation of processing algorithms into hardware and software tools has been described. The basic structure of parallel-flow system with data exchange through multiport memory has been proposed.

Passakorn Phannachitta [2017] Analogue-based software effort estimation (ABE) is a widely adopted method because of the precision offered and its intuitiveness. ABE calculates an estimated effort value for a new software project by adapting to the effort values of its previous similar projects. Precisely measuring the level of similarity between software project cases is an important process of EBA with regard to the fact that the recovery of similar projects is similar to the new project. However, none of our knowledge systematically evaluated and compared the similarity measures for the ABE process. In the present study, 6 similarity measures most often appeared in literatures over a period of 5 years up to the date of writing are systematically compared. Based on a comprehensive empirical experiment using 12 sets of industrial data including 952 project cases, as well as 5 robust performance measures and subjected to a robust statistical test method, we found that simple measures of similarity such as Similarity measurements of Euclid and Manhattan typically provide accurate

estimates for software effort estimation datasets. Although studies in other areas frequently discourage the use of these simple similarity measures; the results of this study support them as a crucial part of an ABE model.

Salma El Koutbi & Ali Idri [2017] Software development error the effort estimation is often studied to regulate the results of the stress estimation technique. The purpose of this article is to propose a quantum approach to treat model error regardless of the technique used to estimate the effort. To achieve this, we explored quantum theory and proposed a model based on an analogy with the problem of potential infinite wells. To evaluate the performance of our approach, we evaluate the classical estimation technique based on analogy on the COCOMO'81 dataset. The main conclusions are as follows: (1) the proposed confidence intervals are close to the theoretical values and (2) the distribution of the Sine-squared efforts represents an interesting alternative to the widely used Gaussian distribution since both distributions are statistically comparable. These results provide an interesting way for further research and investigation.

Kan Qi & Barry W. Boehm [2017] Use case analysis has been widely adopted in modern software engineering due to its ability to capture the functional requirements of a system. This is often done with a UML use case model that formalizes interactions between actors and a system in the requirement extraction procedure, and with the explored architectural alternatives and user interface details specified in the analysis iteration and next planning. On the other hand, to facilitate decision making in software management, effort estimation models are needed to provide estimates of the effort required at the beginning of the project, which, however, provides little information to accurately assess the complexity of the system. To solve this dilemma, an incremental approach to integrating the information available during the first iterations is desirable to provide estimates of the multiple efforts to maintain the balance between utility and accuracy. The author proposed an effort estimation model that incorporates two sub-models to provide an estimate of the effort at two points for the first iterations of a project based on use cases. Our proposed model is lightweight because its size metrics are set to be counted directly by the artifacts of the first iterations. To further calibrate the model, particularly considering the limited number of data points available, we have also introduced a standardization framework in our model calibration process to reduce noise from stress data. By calibrating the proposed sub-models with the data points collected from 4 historical projects, we have shown that the sub-models correspond to the data set and that the next-phase model is superior to the initial phase model. Because it corresponds to the data set better and shows less uncertainty in the calibrated parameters.

Safa Mohammed Ahmed Suliman & Gada Kadoda [2017] Software project management is a central topic in software engineering courses as it explains how to plan, implement, control, monitor, and evaluate software projects. The development

of theories in software metrics and prediction models is based on the broader field of project management, but also attempts to overcome the inherent difficulties of measuring an immaterial object such as software. This paper is part of a research into the factors that influence the estimation of costs and time required for software projects that continue to be a problem for software development organizations. The study described in this article explored the technical and non-technical factors considered critical by Sudanese IT professionals and, if left unmanaged, can lead to cost and time overload or, in some cases, lead to the failure of the project. Using a mixed-method approach, the research project was initially informed through a qualitative study that explored the types of problems that the estimation process faces from the point of view of different levels of staff. This part of the study revealed a number of factors that can be broadly classified as technical factors, for example: the skills of those involved in the estimation process and non-technical factors such as the high level of uncertainty in the local economic context. The second part of the study focused on one of the key factors, the training and experience of the software project staff, using the survey method to determine the extent to which the software engineering curriculum is aligned with the skills required in the software market., especially those related to estimation. The recommendations of this study on reducing estimation errors, whether for business or academia, are preliminary and can only reflect the local approach. However, they also drew on the abundant literature on costing techniques and case studies in similar and more advanced contexts. The problem of predicting software effort and estimation models are a thorny problem in the field of software engineering since the concept of "software crisis" and the field itself, in response to the crisis, are appeared at the end. It still seems to somebody that "after forty years of experience, the term" software engineering "indicates nothing but a vague and largely unsatisfied aspiration" [2]. This study develops our understanding of the issues facing one of the country's young professions, while contributing to the global body of research on the development of techniques to manage the complexity of software engineering over more established engineering disciplines.

Barry W. Boehm [2017] the previous objective, which included a model for estimating the cost of software (and planning), was no longer possible. The sources of variation in the nature of the software development and development processes, products, properties and personnel (PPPP) require a variety of models and estimation methods best suited to their situations. This technical briefing will provide a brief history of radical changes in software estimation methods, a summary of the sources of variation of PPPP software and their implications for estimation, a summary of the types of widely used or emergency estimation methods, one of the best estimation types for different types of PPPP and a process to guide the choices of an organization's estimation methods as their PPPP types evolve.

Muhammad Adnan & Muhammad Afzal [2017] the author concentrates on estimating that the applications attempt and comprehension direction of this Scrum methodology in training, hard activities in a effort circumstance. The suggested strategy enhances pc software attempt estimation and comprehension direction of applications jobs by emphasizing Scrum procedures and techniques which utilize the ontological version within a multi-agent estimation program. Additionally, it encourages vital job stakeholders to frequently record significant tacit expertise on specific conditions, at the kind of lessons learned throughout job creation. Numerous estimation program representatives get the existent knowledgebase and accomplish their particular inference things to do holistically utilizing the illustrative logic in compliance with certain requirements given from the scrum grasp and also providing a suitable quote at the shape of resources, time and course learned victory of prospective endeavors. To confirm our technique, an experiment predicated on a dozen Internet endeavors were ran utilizing the suggested estimation procedures, Delphi process along with preparation. The outcome gained by using the MMRE, PRED (x ray) analysis steps show the suggested approach delivers far more accurate quotes than Delphi and poker intending procedures.

Naoki Kinoshita ; et.al [2018] the author focuses on the problem that the accuracy of estimating the software development effort varies considerably from one software project to another. We propose a predictable classification method for software projects before estimating effort. In the proposed method, given a project to be estimated, we first evaluate whether the effort can be accurately assessed or not by identifying the project as "predictable" or "unpredictable". In the case of predictable projects, we estimate the effort. Otherwise, the estimate is avoided. Following an experiment to evaluate the effectiveness of the proposed method using six sets of sector data, (i) the residual and residual mean square variance proves to be an adequate measure for taking into account predictability; and (ii) the average absolute error is significantly reduced in five data sets, thus avoiding any estimate when a project belongs to the unpredictable class, demonstrating the effectiveness of the proposed method. Using the proposed method, professionals become aware of cases in which they can rely on esteem and cases where they cannot.

Suhyun Cha ; et.al [2018] the technical debt describes the long-term negative effect of choosing non-optimal solutions for short-term profit (for example, cost savings). Engineers of various disciplines, such as mechanics, electricity and software, develop and maintain automated production systems (APS). During their development and maintenance period, changes in one discipline can have a negative or unexpected impact on other disciplines due to interdependencies between disciplines. A list of manually derived activities can omit certain necessary activities and, as a result, TD is often introduced unintentionally. Furthermore, TD's interest due to insufficient knowledge sometimes exceeds short-term cost savings. Author submitted, a workflow is proposed to assist in the process of selecting the solution for the aPS domain.

The workflow includes the systematic tool for assessing the change effort that takes into account the multidisciplinary estimation and TD interest. KAMP4aPS is selected as a tool to estimate the effort of change by deriving a list of fine-grained activities for a change. Based on this detailed estimate and additional cost factors, it is possible to predict TD and choose the appropriate solution, adapted to the current business environment. An example of a change scenario in a laboratory-sized structure is presented to illustrate the proposed methodology.

Emtinan Mustafa & Rasha Osman [2018] Many factors influence the cost estimate of software projects. Environmental factors are specific cultural, social or technical factors in the country that affect the costs of software development. The author has analyzed the characteristics of 31 sets of cost estimate data to determine the integration of environmental factors into their cost characteristics. To analyze the heterogeneous attributes of data sets, we combined all attributes into six categories and 48 representative attributes. We observed that most datasets represent the organizational environments of Europe and North America. In addition, recent datasets have provided more diverse attributes and greater coverage of organizational factors (users, developers, and project attributes). However, environmental factors, namely cultural and social norms, were not represented. This limits the application of these datasets in environments where these factors have a significant impact on software development costs. The author stresses the need for further research on cultural and environmental factors and their impact on software cost estimation.

III. CONCLUSION

Each and every researcher tries improving the accuracy of the existing methods. Few researchers use the Algorithmic model or Non-algorithmic model while some of them use a combination of both. The SLOC or FOC are usual methods for estimating or calculate the software size. There are also other methods to calculate the size of the software, but it depends on the developer which method to use. According to previous studies, gaps have found in improving the accuracy of cost estimation of existing methods. The following are the gaps identified in the existing methods. Choice of an accurate method for calculating the size, Wrong selection of the software cost estimation method, Estimated cost is higher than the actual cost.

REFERENCES

1. *Chin-Yu Huang ; Jung-Hua Lo ; Sy-Yen Kuo ; M.R. Lyu, 1999, "Software reliability modeling and cost estimation incorporating testing-effort and efficiency", ISSN: 1071-9458 , Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No.PR00443), PP: 62-72.*

2. *CH.V.M.K. Hari ; Tegjyot Singh Sethi ; B.S.S. Kaushal ; Abhishek Sharma, 2011, "CPN-a hybrid model for software cost estimation", 2011 IEEE Recent Advances in Intelligent Computational Systems, PP: 902-906.*

3. *Chin-Yu Huang & M.R. Lyu, 2005, "Optimal release time for software systems considering cost, testing-effort, and test efficiency", ISSN: 0018-9529, Volume: 54 , Issue: 4 , PP: 583-591.*

4. *Damir Azhar ; Patricia Riddle ; Emilia Mendes ; Nikolaos Mittas ; Lefteris Angelis, 2013, "Using Ensembles for Web Effort Estimation", ISSN: 1949-3770, 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, PP: 173-182.*

5. *Debra Greenhalgh Lubas, 2016, "System and software cost correlation to reliability", 2016 Annual Reliability and Maintainability Symposium (RAMS), PP: 1-6.*

6. *Emtinan Mustafa & Rasha Osman, 2018, "An Analysis of the Inclusion of Environmental Cost Factors in Software Cost Estimation Datasets", 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), PP: 623-630.*

7. *Barbara Staudt Lerner ; Stefan Christov ; Leon J. Osterweil ; Reda Bendraou ; Udo Kannengiesser ; Alexander Wise, 2010, "Exception Handling Patterns for Process Modeling", **ISSN:** 0098-5589, Volume 36, Issue 2, PP: 162-183.*

8. *Iman Attarzadeh, Amin Mehranzadeh ; Ali Barati, 2012, "Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation", 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks, PP: 167-172.*

9. *Imam Kurniawan, Arry Akhmad Arman ; Sukrisno Mardiyanto, 2017, "Development of analogy-based estimation method for software development cost estimation in government agencies", ISSN: 2155-6830, PP: 1-6.*

10. *Jacky Wai Keung, Barbara A. Kitchenham ; David Ross Jeffery, 2008, "Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation", ISSN: 0098-5589, Volume: 34 , Issue: 4 , PP: 471-484.*

11. *Barbara Staudt Lerner ; Stefan Christov ; Leon J. Osterweil ; Reda Bendraou ; Udo Kannengiesser ; Alexander Wise, 2010, "Exception Handling Patterns for Process Modeling", **ISSN:** 0098-5589, Volume 36, Issue 2, PP: 162-183.*

12. *Prasadu Peddi (2020) "Effort Estimation Methods In Software Development Using Machine Learning Algorithms", ISSN NO:2347-6648, vol 9, issue 1, pp: 824-829.*

13. *Iman Attarzadeh, Amin Mehranzadeh ; Ali Barati, 2012, "Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation", 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks, PP: 167-172.*

14. *Imam Kurniawan, Arry Akhmad Arman ; Sukrisno Mardiyanto, 2017, "Development of analogy-based estimation method for software development cost estimation in government agencies", ISSN: 2155-6830, PP: 1-6.*

15. *Jacky Wai Keung, Barbara A. Kitchenham ; David Ross Jeffery, 2008, "Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation", ISSN: 0098-5589, Volume: 34 , Issue: 4 , PP: 471-484.*

16. *Prasadu Peddi (2020), "Software Development Effort Duration and Cost Estimation using Linear Regression and K-Nearest Neighbors Machine Learning Algorithms", Vol 11, issue 10 PP: 2278-307.*