Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 6 (2023)

A Study on Development of a Cost-Effective Software Maintenance Management Models

Ahmed Masih Uddin Siddiqi
Department of Computer Science Engineering, Mangalayatan University, Beswan, Aligarh, UP, India
Dr. Manoj Varshney
Mangalayatan University, Beswan, Aligarh, UP, India

Abstract:
The process of maintaining software encompasses a wide range of activities, such as the enhancement of capabilities, the correction of errors, the optimisation of performance, and the elimination of obsolete capabilities. In order to guarantee the normal operation of essential software tools and accurate cost estimation, it is vital to ensure that the relevant software tools are maintained. As a result of this, a number of scholars have gravitated towards the study of the many elements that influence the productivity of software development. Research on the development of cost-effective software maintenance management models is the primary objective of this study. This study utilized secondary data to examine the software maintenance activities conducted through the current software development environment. For the purpose of selecting the software development activities, a non-probability convenience and judgement sampling technique was utilised. This was due to the exploratory character of the study. In terms of the functional point, the recommended performance is coming in high, which is 85.65%, and it must be high in order to achieve higher productivity of the software product, which in turn raises the demand in the market and the functionality of the software that is developed. Also, it has been demonstrated that the amount of work required per individual is minimal, coming in at 4.88 percent. This is a factor that ought to be low in order to provide accurate cost estimation in terms of maintenance.

Keywords: *Software; Software Development; Software Maintenance models; Cost – Effectiveness.*

Introduction:
Software maintenance encompasses a wide range of activities, such as enhancing capabilities, rectifying errors, optimising performance, and eliminating outdated features. It is necessary to establish a system for assessing, managing, and implementing modifications in response to the expected occurrence of change (Singh et al., 2019). During software development, it is crucial to create a well-defined plan for maintenance activities, which is an essential component of software maintenance. This plan specifies the adjustments that need to be implemented. In order to account for any changes in requirements, it is necessary to incorporate the cost of software development into the budget (Pospieszny et al., 2019). This implies that the maintenance expenses would rise, not alone because of inadequate design, but also owing to alterations in customer environmental and expected requirements, for which the system has been created (Islam et al., 2023).
Approximately 60% of the workforce in software development organisations from foreign countries will be allocated to maintaining the existing software (Khezami et al., 2021). The percentage of software tools and personnel is continuously increasing on a daily basis. Users and developers commonly encounter software maintenance issues (Khezami et al., 2021). Prior planning and cost analysis by developers and users are crucial factors in accurately estimating software maintenance costs. The primary objective of this project is to investigate the development of cost-effective models for managing software maintenance.
A discussion of the previous literatures that are relevant to this research is presented in the following section.

Literature REVIEW:

The following table discussed about the past literatures related to development of a cost-effective software maintenance management models.

Table 1: Related works

AUTHORS AND YEAR	METHODOLOGY	FINDINGS
Singh et al., (2019)	Explanatory based study implementation is used	Various approaches are used to estimate software maintenance costs due to its importance. Maintenance estimation models are separated by granularity level into Phase, Release, and Task levels. Most software costs can be evaluated by considering maintenance, a crucial software engineering task.
Andi, (2021)	In this work, a model was proposed and a brief description was provided on the concept of serverless cloud computing, its application in the information technology industry, and the advantages it offers.	It is possible to significantly cut down on the amount of time required for a system to be brought to market by utilizing this strategy, which also results in cost savings. Among the three groups that make up serverless architecture are Amazon Web Services Lambda, Microsoft Azure, and Google Cloud.
Machado et al., (2023)	For this study, a comprehensive literature analysis was conducted on the relevant works that were published between the years 1990 and 2020 in the field of software engineering. The search engines that were utilized were the most important ones.	Software maintenance metrics were identified in the product, process, and resource perspectives more broadly directed toward the product, and their application validation is done through technical experimentation of the measurement, which often does not match the software maintenance environment.
Pargaonkar, 2023	The study commences by doing a thorough theoretical analysis of classical Software Development Life Cycle (SDLC) models, such as the Waterfall, Iterative, and VModel through qualitative research approach.	Software development stakeholders and quality engineers can choose the best SDLC method for their projects by analysing each model's strengths and weaknesses.
Lima et al., (2023)	This study collected data from testing experts using convenience, purposive, and snowballing sampling methods in accordance with software engineering survey principles.	This study found that refactoring improves software testing team performance and automated test maintenance.

Research Gap

Subsequently, certain new trends in maintenance management are acknowledged, which will assist academics in identifying areas where the current literature is lacking and in efficiently focusing their future research endeavors.

So the goal of this study is to conduct a research on the development of a cost-effective software maintenance management models. Researchers, maintenance professionals, and individuals in the maintenance industry will all gain valuable insights from this study, as it will enhance their comprehension of the crucial aspects of maintenance management.

Methodology

The proposed methodology utilizes MATLAB Bug Id reports to acquire datasets for performing sentiment classification based on severity. The development tool MATLAB has been utilized in this project. The dataset includes COCOMO, PUTNAM and PUCO (HYBRID). Additionally, there may exist alternative instruments that might be employed in this development, such as quantifying individual engagement and their level of interest. Eclipse, a widely used development platform, is utilized by developers worldwide. These developers employ technical terminology while creating bug reports. This helped in describing the severity level of the defect and developing a more precise lexicon.

Subsequently, the textual summary report obtained from the dataset acquisition undergoes pre-processing. Using a summary as a severity level forecast yields superior outcomes when compared to an emotional report's detailed explanation. The process involves tokenization, stop word elimination, and stemming. The first step is Tokenization, which aims to eliminate punctuation marks. The entire text is segmented into distinct tokens to facilitate the exploration of all terms in the document. The second step is stop word removal, which aims to eliminate common words such as articles, prepositions, conjunctions, adjectives, adverbs, and frequent verbs from the textual data. These words lack significant information as they are often employed words. As a result, the data (text) was reduced and the system's performance improved. The third technique employed is stemming, which aims to reduce words to their base forms.

Results And Discussions

The outcomes achieved with the suggested methodology. This full chapter is divided into three pieces. The initial step of the COCOMO model involves the utilization of SLOC, functional points, and efforts per person each month to calculate the results. The functional point percentage is 60.885%, while the set effort per person per month and SLOC (Source Lines of Code) are 12.0929 and 733, respectively. The second step of the PUTNAM model involves the utilization of the Tomcat dataset, which is first subjected to pre-processing. Subsequently, Principal Component Analysis (PCA) is employed to extract features from the pre-processed dataset. The Particle Swarm Optimization (PSO) algorithm is utilized to analyze the extracted features for the purpose of instance selection. These selected instances are then classified using the Linear Discriminant Analysis (LDA) technique. The COCOMO model quantifies the work required per person as 4.84% and measures the source lines of code achieved as 464. Additionally, it accounts for 3.92% of the function point. The PUTNAM model's performance evaluation yields a predicted object point of 27.5%, an effort requirement of 33.67% per person, 900 source lines of code, and an achievement of 66.32% of function point. Both COCOMO and PUTNAM possess their own strengths and weaknesses. Therefore, in the third phase, they are merged to leverage their respective benefits.

Table 1: Performance Comparison using different datasets

Model	COCOMO	PUTNAM	PUCO (HYBRID)
SLOC	463	902	576
Efforts Per Person	4.83 %	33.66 %	4.88 %
Functional Point	3.93 %	66.33 %	85.66 %
Relative Mean Square Error	0.21	0.25	0.002

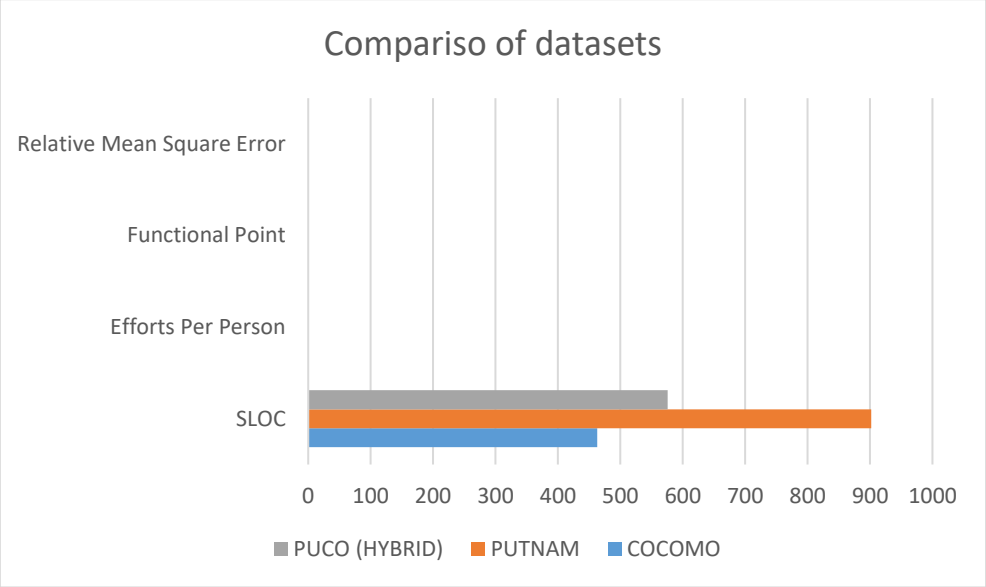

Figure 1: Performance Comparison using different datasets

When these three models are examined for 500 and 1056 records of the dataset, respectively, in terms of effort per person and functional point, it is demonstrated that the hybrid PUCO model's performance is superior to that of other models. When it comes to the Eclipse Bug Prediction Dataset, the model is evaluated for a total of 698 and 500 records respectively.

It is possible to calculate the Effort per person and functional point for each of the three models by taking into account 16 lines of code from the dataset that contains 500 and 998 records respectively. With regard to each and every set of lines of code, the findings of both models indicate that the suggested hybrid PUCO model produces a superior output in comparison to the other two models. It is possible for the number of source lines of code to be high or low depending on the software requirement, which must be produced in accordance with the user's point of view.

Conclusion:

The recommended performance, measured in terms of functional points, is very high at 85.65%. This high performance is crucial for achieving increased production of the software product, which in turn improves its demand in the market and enhances the functionality of the generated software. Additionally, the data reveals that the individual efforts are similarly minimal, amounting to 4.88%. This low level of effort is desirable for accurate cost estimation in relation to maintenance. Generally speaking, when working with the software development cost estimation platform, there are a lot of different possibilities that can be explored in the future. One example of this would be expanding this platform beyond its original purpose of providing a cost estimation platform to incorporate various types of maintenance, agile software development methodologies, and even the possibility of improving the approach to regression.

References:

[1] Andi, H. K. (2021). Analysis of serverless computing techniques in cloud software framework. *Journal of IoT in Social, Mobile, Analytics, and Cloud*, *3*(3), 221-234.

[2] Islam, M., Farooqui, N. A., Haleem, M., & Zaidi, S. A. M. (2023). An Efficient Framework for Software Maintenance Cost Estimation Using Genetic Hybrid Algorithm: OOPs Prospective. *International Journal of Computing and Digital Systems*, *14*(1), 1-xx.

[3] Khezami, N., Kessentini, M., & Ferreira, T. D. N. (2021). A Systematic Literature Review on Software Maintenance for Cyber-Physical Systems. *IEEE Access*, *9*, 159858-159872.

[4] Lima, D. L., Santos, R. D. S., Garcia, G. P., Da Silva, S. S., França, C., & Capretz, L. F. (2023, October). Software testing and code refactoring: A survey with practitioners. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 500-507). IEEE.

[5] Machado, J., Kemczinski, A., & Schroeder, R. (2023, May). Survey of Software Maintenance Metrics: A Systematic Literature Review. In *Proceedings of the XIX Brazilian Symposium on Information Systems* (pp. 332-339).

[6] Pargaonkar, S. (2023). A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. *International Journal of Scientific and Research Publications (IJSRP)*, *13*(08).

[7] Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, *137*, 184-196.

[8] Singh, C., Sharma, N., & Kumar, N. (2019). Analysis of software maintenance cost affecting factors and estimation models. *Int. J. Sci. Technol. Res*, *8*(9), 276-281.

[9] Singh, C., Sharma, N., & Kumar, N. (2019). Analysis of software maintenance cost affecting factors and estimation models. *Int. J. Sci. Technol. Res*, *8*(9), 276-281.