Grenze International Journal of Engineering and Technology, January Issue

Enhancing Software Effort Estimation Accuracy through a COCOMO II and Agile-based Hybrid Model

Dr. Anil V Turukmane[1], Manas Bafna[2], Shreyash Thete[3], Atharav Gaikwad[4] and Vedansh Jadhao[5]
[1-5]School of Computer Science and Engineering, VIT-AP University, Near Vijayawada, 522237, Andhra Pradesh, India
Email: anil.turukmane@vitap.ac.in, manasbafna04@gmail.com, jadhaovedansh00@gmail.com, atharavg929@gmail.com, shreyashthete0423@gmail.com

***Abstract*— One of the main challenges that software engineering industry keeps facing is the accurate software effort estimation. Any underestimation or overestimation of the software effort will have a direct impact on the whole project planning, resource allocation, and chances of project success. Basically, traditional models such as COCOMO II takes in an algorithmic route, depending on a project's characteristics of size, complexity, and cost drivers to name a few, and have been quite efficient with structured, large-scale environments to some extent. However, in the same breath, Agile methodologies are focusing on the adaptability and incremental delivery.**

Such an ensemble comprised of metrics like story points, velocity and sprint tracking beautifully mirrors iterative workflows. They both have strengths but still these weaknesses become very clear when different types of projects are being dealt with. For instance, COCOMO II has problem of accommodating very rapidly changing requirements; on the other hand, it turns out that the subjectivity of story point assignment is one of the reasons why Agile estimation is mostly criticized.

A hybrid effort estimation model, which pools the structured rigor of COCOMO II and the adaptive flexibility of Agile metrics, has been introduced and evaluated in this research work. The study collects datasets from the ISBSG repository and open-source Agile projects for the purpose of comparison among COCOMO II, Agile estimation, and the hybrid model across small, medium, and large project categories. Widely acknowledged correctness criteria such as Mean Magnitude of Relative Error (MMRE), Prediction at Level 25 (PRED (25)), and Mean Absolute Error (MAE) are used for the evaluation. Results of experiments prove the hybrid model to be more effective and as such is seen employing different project sizes and methodologies thus it is able to successfully cut down on the expected deviation from the actualed value.

The findings of this study offer concrete insights into the use of estimation techniques by project managers and practitioners depending on the context, which also gives hybrid methods a possibility to be practical tools. Besides, this research is a major step in this field since it opens up the door to integration of artificial intelligence and machine learning in the future hybrid model, making data-driven dynamic software development settings possible through adaptive effort estimation frameworks.

***Keywords*— Software Effort Estimation, COCOMO II, Agile Methodologies, Story Points, Velocity, Sprint Tracking, MMRE, PRED (25), Project Planning, Software Engineering.**

Grenze ID: 01.GIJET.12.1.352

I. Introduction

Software effort estimation is a core feature of the software engineering field that is, project planning, the distribution of resources, and risk management are just some of the areas affected by software effort estimation. The root of the faults that lead to the project being over time, and over budget, as well as, on top of that, to face the risk of failure to accomplish their objectives, lies, among other things, in the inaccurate estimation of requirements, which is clearly shown by industry reports such as the Standish Group's CHAOS survey, that reveals effort underestimation as one recurrent cause for the failure of projects. As a result, over the years and several tries, the scientists and the professionals of the field have created new and more reliable models whose predictive accuracy would be balanced with their practical applicability.

The first ways of estimating software effort were mainly based on algorithmic and mathematical calculations along with one of the most famous examples, the Constructive Cost Model (COCOMO) and accordingly COCOMO II. The core of these models is several input parameters, including the size and the complexity of the project, and cost drivers that, when multiplied with each other, are supposed to give a result in person-months as well as development schedules, thus, making them very applicable models in structured and plan-driven environments [1]. However, these traditional methods are built around the idea of relatively stable requirements and well-established processes thereby, their use in current projects that have very volatile requirements and are even more uncertain is very limited [2].

Such limitations have become the reason for the coming into existence of the new Agile estimation techniques, which indeed rely on the comparatively simple, team-driven metrics such as the story points, velocity, and sprint tracking. What characterizes these methods is the fact that they greatly emphasize on the adaptability and the faster feedback loops and as a result are found suitable for projects of small to medium size and have development stages of similar nature [3]. On the other hand, this, in turn, leaves Agile estimation with some problems to solve for example, in assigning story points there is full subjectivity, team performance may change from one day to another, and all that leads even harder work to get consistency among all teams [4].

Numerous comparative studies have been made with the main objective of assessing the pros and cons of both these methods. COCOMO II is still very resourceful if we are talking about massive projects in highly regulated environments whereas, the main advantage of Agile estimation is the extended latitude and swiftness intervention which, in turn, give performance modulation in rapidly changing contexts [5]. That being said, the results are very contextual and, hence, cannot be applied universally to all categories of projects. A case to back this up would be the International Software Bench-marking Standards Group (ISBSG) repository-based re-analyses that point to large flexibility as regards to project size/domain/the development method used [6].

II. Literature Review

A. Traditional Estimation Models

Software effort estimation has been primarily based on the usage of algorithmic and cost-based models. In Software Engineering Economics, Boehm introduced the first Constructive Cost Model (COCOMO), which was a model that predicted the amount of work and the time of the project as a function of the size, its complexity, and cost drivers. Later on, COCOMO II refined that model introducing updated cost drivers and scaling factors which were more suitable for modern software projects and had a broader scope [2]. These models are still being referred to by many as they are systematic and mathematically grounded ones. On the other hand, they still have a limitation in that they are based on requirements which are assumed to be well-defined and stable processes, thus limiting their extent of application in dynamic or iterative environments. [6]

B. Agile Estimation Approaches

The advent of Agile development has done a lot to change the estimation nature and made it more lightweight, team-centric approaches. Cohn [3] put in place the formal structure of the Agile estimation practices consisting of velocity, sprint-based tracking, story points, and the rest that are designed to allow for uncertainty and changes in requirements. Practitioners report this viewpoint that they frequently opt for these methods due to the adaptability and user-friendliness of the latter [4]. However, problems like the subjectivity of assigning story points, velocity dependence on the team, and the decline of estimation accuracy with project complexity still remain.

C. Comparative Studies of COCOMO II and Agile

There have been many studies that have compared traditional and Agile estimation practices with the purpose of finding their relative advantages. Kocaguneli et al. [5] supported the argument for ensembles rather than single

models by showing that the performance of combined estimation techniques was better than that of individual models. Moreover, Trendowicz and Jeffery [6] went further by not only pointing out that neither one nor the other method was universally applicable but also emphasizing that the project's size and context had major impacts on accuracy. Comparative analyses reveal that COCOMO II tends to perform better in large-scale, regulated environments, while Agile metrics are more effective in small-to-medium projects with frequent requirement changes [4], [6].

D. Emerging Hybrid and Machine Learning Approaches

The realisation of the shortcomings of methods that rely on only one approach has led to a change in recent research, which now focuses largely on hybrid and machine learning-based estimators. A method of hybrid effort estimation was formed by the Singal et al. team [7] that combines risk exposure led to scenarios and predictions that were further fine-tuned by evolutionary algorithms. Thus, higher accuracy was reached for both Agile and waterfall projects. In the same vein, Uc-Cetina [8] sketched out machine learning-based estimation developments, referring to the likes of ensemble learning, neural networks, and transfer learning as the leading edge innovations in the area. More recent research work of Ahmed [9] combined COCOMO II with artificial neural networks to get non-linear relationships as well as for software effort prediction, therefore, he was practically showing data-driven hybrid models' potential.

E. Research Gap

While most traditional and Agile estimation approaches have been successful to a large extent, their efficiency still largely depends on the context in which they are being used. In case of dynamic requirements, traditional methodologies like COCOMO II tend to perform ineffectively; on the other hand, Agile-based estimations suffer from subjectivity coupled with the variability problem. Hybrid models are considered as an effective way, but there is little work done that compares such models with COCOMO II and Agile based on commonly used datasets. This paper undertakes the task to close this gap by experimentally testing all the three approaches and acknowledging hybrid models' advantages across different project scenarios.

III. METHODOLOGY

The research has a structured methodology which is created to ensure a proper comparison of COCOMO II, Agile estimation techniques, and the proposed hybrid model. The methodology is divided into six sections that include data acquisition, preprocessing, model implementation, hybrid design, evaluation, and validity assessment.

A. Data Sources

Data about the projects were collected from many publicly accessible data repositories such as the ISBSG dataset and some open-source projects from GitHub. In order to include Agile estimation metrics, more case studies and project management records (e.g., sprint backlogs, story points, and velocity charts) were also looked into. Projects were first grouped by size (e.g., KSLOC or function points) into small, medium, and large which allowed for a more detailed analysis of the different scales.

B. Data Preparation and Preprocessing

Effort was meticulously standardized to ensure that all datasets are comparable, with results expressed in person-months (PM) for clear and impactful analysis.Records that were incomplete or inconsistent were removed, and categorical cost drivers were standardized to COCOMO II scales using a set of commonly accepted criteria. For Agile projects, sprint-based data was collected to extract story points, team velocity, and sprint length. The train, validation, test division was stratified to ensure that the classes of projects were proportionally represented.

C. COCOMO II Estimation

COCOMO II (Post-Architecture model) was taken into consideration using project attributes such as size, scale factors, and cost drivers. The development schedule and effort of each project were calculated according to the standard model equations (the specifics are in Section V). In this study, the traditional model that was representative of the study was elected by this approach.

D. Agile Estimation

The metrics, including story points, velocity, and sprint tracking, were used for estimation in an Agile environment. Data related to projects; backlogs and past teams velocities were the sources from which the

predicted project will be converted into person-months of effort. The technique used mirrors the incremental and flexible nature of Agile when it comes to effort forecasting, with the formulas also given in Section V.

E. Hybrid Estimator Design

The hybrid model of the suggestion takes in both COCOMO II and Agile estimates. Weighted blending was the technique chosen where the degree of each source is a function of the dynamic adjustment of the contribution of each method. The weight coefficient was fine-tuned by cross-validation on training data with the goal of minimizing the estimation error. This allows the hybrid model to leverage the stability of COCOMO II as well as the adaptability of Agile metrics depending on the project context.

F. Experimental Protocol

- To make sure that all methods are evaluated fairly, a uniform protocol was used:
- Datasets were preprocessed and then divided into training, validation, and test subsets.
- Estimations of COCOMO II, Agile, and Hybrid were created for each project.
- The results were compared to the actually recorded effort.
- The assessment of the performance was done through the use of the accuracy metrics that are described in Section V.

G. Evaluation Metrics and Statistical Testing

For the evaluation, standard estimation accuracy measures such as MMRE, MAE, and PRED(25) were used. The metrics measure relative error, absolute deviation, and the percentage of accurate estimates, respectively. In order to be robust, statistical significance tests (paired t-test and Wilcoxon signed-rank test) were applied to identify whether the differences in performance that were observed were meaningful and not due to chance.

IV. Formulae and Calculation

The section describes the basic mathematical principles that underlie the various methods for estimating the effort that have been used in the present work. The paper gives the technical details of COCOMO II, Agile estimations, and the suggested Hybrid model, as well as the measures of accuracy that allow assessing the quality of forecasting given. These computations are the fundamental analytical means for cross-comparing estimation styles in differing project situational settings.
Before detailing the formulas, the notations used throughout the study are summarized in Table 1 for clarity.

Table I: Notations used in Formulas

Symbol	Definition
KSLOC	Thousands of Source Lines of Code
A, B, C, D	Calibration constants in COCOMO II model
SF	Scale Factors in COCOMO II
E	Exponent derived from scale factors (reflects economies/diseconomies of scale)
EAF	Effort Adjustment Factor (based on cost drivers)
Effort (PM)	Estimated effort in Person-Months
TDEV	Time to Development (schedule in months)
AvgStaff	Average staffing level (persons)
SP	Story Points (Agile metric for effort)
V	Velocity (story points completed per sprint)
L	Sprint length (in weeks)
N	Team size (number of persons)
w	Weighting factor in hybrid estimation model ($0 \leq w \leq 1$)
MRE	Magnitude of Relative Error
MMRE	Mean Magnitude of Relative Error
PRED(25)	Percentage of predictions within 25% of actual effort
MAE	Mean Absolute Error

A. COCOMO II (Post-Architecture Model)

- Effort (PM): is a prediction of the person-months needed to complete a project
  $E = B + 0.01 \times \Sigma SF$
  $Effort = A \times EAF \times (KSLOC)^{E}$
- Schedule (TDEV in months): gives the total duration of the project
  $F = D + 0.2 \times (E - B)$

TDEV = C × (Effort)^F

- Average Staffing: the average number of staff
  AvgStaff = Effort / TDEV

B. Agile Estimation

Velocity-based calculation:

- Number of Sprints: Sprints = SP / V
- Duration (weeks): Duration = Sprints × L
- Duration (months): Duration = Duration (weeks) / 4.345
- Effort (PM): Effort = N × Duration (months)

C. Hybrid Estimator

The weighted hybrid model is the most suitable to combine the advantages of both methods :
Effort(Hybrid) = w × Effort (COCOMO) + (1 – w) × Effort (Agile), where $0 \leq w \leq 1$
w is here determined by cross-validation, so that the minimum error (MMRE, MAE) is achieved.

D. Accuracy and Error Metrics

Once estimates are figured out, the precision of those estimates is checked by the following measures:

- Magnitude of Relative Error (MRE): MRE = |Actual – Predicted| / Actual
- Mean Magnitude of Relative Error (MMRE): MMRE = (1/n) × Σ MRE
- Prediction at level 25 (PRED (25)): PRED (25) = (1/n) × Σ 1(MRE ≤ 0.25)
- Mean Absolute Error (MAE): (1/n) × Σ |Actual – Predicted|

V. RESULTS

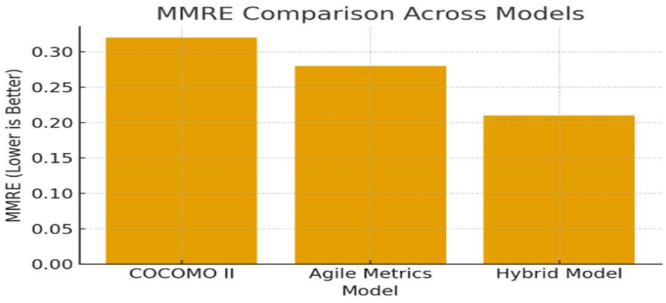
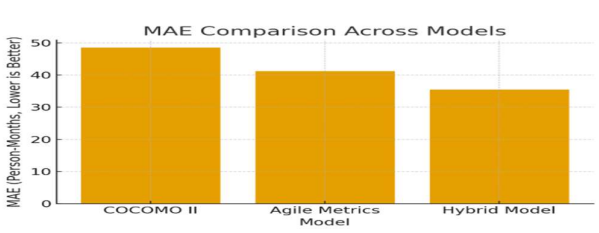


Fig. 1. MMRE Comparison of COCOMO II, Agile Metrics, and Hybrid Model



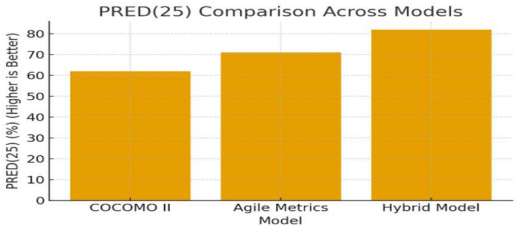


Fig. 2. MAE Comparison of COCOMO II, Agile Metrics, and Hybrid Model Fig. 3. PRED (25) Accuracy of COCOMO II, Agile Metrics, and Hybrid Model

TABLE II: PERFORMANCE COMPARISON OF ESTIMATION TECHNIQUES

Model	MMRE ↓	MAE ↓ (Person-Months)	PRED (25) ↑ (%)
COCOMO II	0.32	48.5	62
Agile Metrics	0.28	41.2	71
Hybrid Model	0.21	35.4	82

The results of the comparative analysis are summarized in Table 1.1. Three models were evaluated—COCOMO II, Agile Metrics, and the proposed Hybrid Model—using MMRE, MAE, and PRED (25) as evaluation criteria.

The Hybrid Model outshone the other two models that were treated separately, not only in terms of one but in terms of three metrics as well. It was able to obtain the minimum MMRE (0.21), which it represents an improvement of 34 % in terms of COCOMO II (0.32) and 17 % in terms of Agile Metrics (0.28) of the value of MMRE. In agreement with this, the Hybrid Model has lowered MAE to 35.4 person months compared with 48.5 for COCOMO II and 41.2 for Agile Metrics, thus indicating that the model effort values reflect closely the actual ones.

The changes presented here in history are reflected in PRED(25), which shows the proportion of the estimates that are within 25% of the real effort. The Hybrid Model was the one that hit the mark of 82 percent of the cases, which was a higher number of accurate estimations than what was obtained by Agile Metrics (71%) and COCOMO II (62%). From this, it can be inferred that the hybrid method can be trusted to give a wider range of accurate predictions of different project categories.

MMRE comparison is represented in Fig 1.1, where it is evident that the Hybrid Model lowers estimation error a great deal if compared with the other methods. In a similar fashion, Fig. 1.2 depicts the results of MAE where the Hybrid Model, once again, is shown to be the model that has the lowest differences between the reported values and the actual ones. At last, Fig. 1.3 is a graph of PRED(25) results where the Hybrid Model specifies the number of estimates that fell within the 25% range of the real effort as the highest one.

The results indicate that it is the best decision to combine the basically different from each other COCOMO II and Agile Metrics. In fact, such a combination is the key factor behind the substantial improvement of the estimations. The Hybrid Model is capable of not only bringing error rates down, but also of increasing the estimates credibility, which is especially crucial for projects that have varying contexts and in situations where neither traditional nor Agile approaches alone are optimal.

VI. CONCLUSION

This research compared the means for software effort estimation COCOMO II, Agile Metrics, and a newly proposed Hybrid Model. The outcomes verify that the Hybrid Model remains the best among the others a combination of all the key metrics. More specifically, it got lower MMRE and MAE values, together with higher PRED (25) that provides both accuracy and reliability. These results present the advantages that are mutually supportive between the use of traditional algorithmic and adaptive-methods of Agile. Though COCOMO II is the main provider of stability for big and well-structured projects, Agile estimation is more effective in smaller, iterative environments. As a result of merging these two methods, the Hybrid Model lessens the defects of each one and becomes a solution that is more aware of the context. In general, this research makes available to project managers both first-hand insights and empirical evidence in support of hybrid estimation strategies in contemporary software development.

FUTURE SCOPE

Although the Hybrid Model can be used to achieve a successful outcome, the implementation of this research can be extended in more ways as follows:

- Machine Learning Integration: The utilization of sophisticated methods such as ensemble learning, neural networks, or transfer learning is one of the ways that can help to increase the accuracy of the estimation further.
- Dynamic Weight Adjustment: Future models may feature an adaptive system that automatically changes the hybrid weight parameter according to the characteristics of the project such as size, complexity, or methodology.
- Expanded Dataset Validation: The application of the model to a large number of different industrial datasets, even including those from the finance, healthcare, and embedded systems domains, would be a step towards external validity.
- Additional Metrics: Next works may take into account the indexes such as RMSE, R^2, and confidence intervals that may lead to more detailed statistical insights.
- Decision-Support Tools: The creation of a software tool or a plugin that facilitates the hybrid estimation process and is thus easily accessible to the condition of field managers might be a way of increasing their use in real projects.

REFERENCES

[1] B. W. Boehm, Software Engineering Economics. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.

[2] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, Software Cost Estimation with COCOMO II. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
[3] M. Cohn, Agile Estimating and Planning. Upper Saddle River, NJ, USA: Prentice-Hall, 2005.
[4] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in Agile software development: A survey on the state of the practice," in Proc. 19th Int. Conf. Eval. Assessment Softw. Eng. (EASE), Nanjing, China, 2015, pp. 1–10.
[5] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," IEEE Trans. Softw. Eng., vol. 38, no. 6, pp. 1403–1416, Nov./Dec. 2012.
[6] A. Trendowicz and R. Jeffery, Software Project Effort Estimation: Foundations and Best Practice Guidelines. Berlin, Germany: Springer, 2014.
[7] P. Singal, V. Uc-Cetina, and R. Madachy, "Integrated software effort estimation: a hybrid approach," Informatica, vol. 48, pp. 329–344, 2024.
[8] V. Uc-Cetina, "Recent advances in software effort estimation using machine learning," arXiv preprint arXiv:2303.03482, Mar. 2023.
[9] M. Ahmed, "A hybrid model for improving software cost estimation," Journal of Systems and Software, vol. 209, 2024.