ISSN: 2455 - 9172

EIJO Journal of Engineering, Technology And Innovative Research (EIJO–JETIR)

Einstein International Journal Organization (EIJO)

Available Online at: www.eijo.in

Volume – 10, Issue –3 , May - June - 2025 , Page No. : 12 - 21

Software Effort Estimation Models: A Comparative Review

[1]Prachi Gupta, Assistant Professor, Department of Computer Science, Arya College of Engineering, Jaipur

[2]Shaizan Alam, Research Scholar, Department of Computer Science, Arya College of Engineering, Jaipur

[3]Shahnawaz Alam, Research Scholar, Department of Computer Science, Arya College of Engineering, Jaipur

Abstract

Software effort estimation is a crucial and foundational activity in software project management that significantly influences the success of software development initiatives. Accurate estimation plays a vital role in enabling effective project planning, budgeting, scheduling, resource allocation, and risk management. Errors in effort estimation can lead to cost overruns, missed deadlines, and compromised software quality, making it essential for project managers and stakeholders to adopt reliable estimation methods.

Over the years, numerous models and methodologies have been proposed and refined to enhance the accuracy, consistency, and applicability of software effort estimation. These models broadly fall into three major categories: algorithmic (model-based), non-algorithmic (expert-based or data-driven), and hybrid approaches. Algorithmic models, such as COCOMO and Function Point Analysis, rely on mathematical formulas and historical data. Non-algorithmic methods include expert judgment, analogy-based estimation, and machine learning techniques, which offer flexibility and adaptability to specific project contexts. Hybrid models combine elements from both categories to leverage the strengths and mitigate the weaknesses of individual approaches.

The paper also discusses the practical challenges faced in implementing these models, such as data availability, model complexity, domain-specific constraints, and organizational maturity. By synthesizing insights from academic literature, industrial case studies, and recent empirical evaluations, the paper identifies emerging trends and future research directions, including the potential of automated estimation tools and the role of continuous learning systems.

In summary, this review provides a comprehensive and insightful resource for both researchers and practitioners, guiding them in selecting and applying the most suitable software effort estimation techniques based on project requirements, organizational context, and technological advancements.

Keywords*:* Software Effort Estimation, COCOMO, Machine Learning, Analogy-Based Estimation, Expert Judgment, Function Point Analysis, Agile Estimation, Project Management, Hybrid Models, Software Development

Introduction

In the realm of software engineering, accurate effort estimation plays a critical role in determining the success or failure of a project. Despite advances in methodologies and tools, many software projects still suffer from cost overruns, schedule delays, and resource misallocation, often due to poor estimation practices. Effort estimation refers to the process of predicting the amount of human labor, usually measured in person-hours or person-months, required to complete various phases of a software development life cycle (SDLC), including planning, designing, coding, testing, and deployment.

Given the inherent complexity and uncertainty in software projects—stemming from evolving requirements, technological challenges, and human factors—estimation becomes a non-trivial task. Therefore, software effort estimation is not just a technical challenge, but also a strategic management concern that influences budgeting, staffing, and risk mitigation.

Over the decades, researchers and practitioners have proposed a wide variety of estimation models to improve prediction accuracy. These models range from intuitive expert-based techniques to data-driven statistical and machine learning approaches, and more recently, hybrid models that seek to combine the best of both worlds. Each model comes with its own set of assumptions, advantages, and limitations, making it essential to understand their applicability in different project contexts.

Classification of Estimation Models

Software effort estimation models can be broadly categorized into three major types based on their methodology and underlying principles:

Algorithmic Models

These are mathematical or formula-based models that use well-defined equations to calculate effort based on specific input parameters. The inputs typically include size metrics (like lines of code or function points), complexity factors, and other environmental variables. These models aim to provide repeatable and objective estimates, especially in traditional software development environments.

Non-Algorithmic Models

Non-algorithmic models do not rely on predefined equations. Instead, they are based on subjective judgment, analogies, or data-driven methods such as machine learning. These models are often more flexible and adaptable to different types of projects, particularly those with limited historical data or where qualitative insights are critical.

Hybrid Models

Hybrid models integrate the strengths of both algorithmic and non-algorithmic approaches. They often use an algorithmic model as a baseline and refine the output using advanced techniques such as neural networks or fuzzy logic systems. The objective is to improve the accuracy of estimates while accommodating variability and uncertainty.

Algorithmic Models

Algorithmic models have been among the earliest and most widely adopted approaches in software effort estimation. They rely on quantitative metrics and mathematical expressions to predict effort, time, and cost. Two of the most prominent models under this category are COCOMO and Function Point Analysis.

COCOMO (Constructive Cost Model)

The Constructive Cost Model (COCOMO), developed by Barry W. Boehm in 1981, is a seminal model in software engineering. It estimates the effort required based on the size of the software, typically measured in thousands of lines of code (KLOC), along with various cost drivers related to product, personnel, and project attributes.

COCOMO comes in several variants:

- Basic COCOMO: Provides a rough estimate using only software size and a constant multiplier.
- Intermediate COCOMO: Incorporates additional factors like software reliability, team capability, and tools used.

- COCOMO II: An updated version designed for modern software practices, supporting object-oriented development and iterative models**.**
- COCOMO's transparency and historical calibration make it a preferred choice in organizations with structured development processes, although it can be limited by its dependency on accurate size estimation early in the project**.**

Function Point Analysis (FPA)

Function Point Analysis, introduced by Allan Albrecht at IBM, offers a method of estimation based on the functionality delivered to the user, rather than lines of code. It evaluates components such as:

- External Inputs (user data entry)
- External Outputs (reports and messages)
- Internal Logical Files (logical groupings of data)
- External Interface Files (data from other systems)
- User Inquiries (interactive queries)

FPA is particularly effective during the early stages of the SDLC, when the codebase has not yet been developed. It is language-independent and focuses on what the system does, making it useful in scenarios with well-defined user requirements**.**

Non-Algorithmic Models

Non-algorithmic models offer a flexible and adaptive alternative to formula-based estimation, often incorporating human expertise, past experiences, and machine learning to generate more context-aware predictions**.**

Expert Judgment

Expert judgment relies on the intuition, experience, and domain knowledge of software professionals to make estimates. This method is common in agile development, where formal metrics may be unavailable or difficult to apply.

Popular techniques include

- Planning Poker: A consensus-based approach where team members estimate effort using cards with numbers (e.g., Fibonacci series), followed by discussion and re-estimation.
- Delphi Technique: Experts independently provide estimates which are then anonymously discussed, and the process is repeated until convergence is reached.
- While highly intuitive and practical, expert judgment can be subjective and prone to bias, especially in teams with uneven experience levels.

Analogy-Based Estimation

Analogy-based estimation is grounded in the principle of case-based reasoning, where effort is estimated by identifying and comparing the current project with historically similar projects. The assumption is that projects with similar attributes (e.g., size, complexity, team composition) will require similar efforts.

This method requires a robust repository of past project data and mechanisms to measure similarity accurately. While it can be very effective, its success largely depends on data availability and the relevance of analogies.

Machine Learning-Based Estimation

Machine learning (ML) techniques are increasingly being applied to software effort estimation, thanks to their ability to learn patterns from historical data and make predictions even with complex, non-linear relationships.

Common ML techniques include:

- Linear and Non-linear Regression: For identifying relationships between input factors and effort.
- Artificial Neural Networks (ANNs): Useful for modeling intricate dependencies.
- Support Vector Machines (SVM): Effective in high-dimensional spaces with limited data.
- Decision Trees and Random Forests: Provide interpretable models that can handle both categorical and numerical data.
- ML-based models can adapt and improve over time as new data becomes available, but they require substantial training data, preprocessing, and validation to avoid overfitting and ensure reliability.

Hybrid Models

Hybrid models aim to combine the structured predictability of algorithmic methods with the adaptive intelligence of non-algorithmic techniques to create a more robust estimation framework.

Examples and Approaches

- COCOMO + Neural Networks: Use COCOMO to generate a baseline estimate and then apply neural networks to adjust the output based on historical project deviations.
- Fuzzy Logic + Expert Judgment: Apply fuzzy logic to handle uncertainty in expert estimates, especially when precise numerical input is not possible.
- Ensemble Techniques: Combine multiple ML models (e.g., regression + decision trees) to enhance accuracy through voting or averaging mechanisms.
- Hybrid models are particularly beneficial in dynamic and uncertain environments, where no single method provides sufficient accuracy. They also help in bridging the gap between historical data and human intuition, offering a practical middle ground.

Recent Examples

A comprehensive study conducted in 2023 involving 60 software development projects across two major Indian IT firms—Infosys and Tata Consultancy Services (TCS)—provided compelling empirical evidence supporting the effectiveness of hybrid software effort estimation models. The research aimed to evaluate the accuracy, scalability, and adaptability of traditional and modern estimation techniques across a diverse range of project types, including both enterprise-level applications and agile-driven modules.

The study compared various estimation models, including standalone algorithmic models like COCOMO II, non-algorithmic models based on machine learning, and hybrid approaches that integrated both. Among the hybrid configurations, the model that combined COCOMO II with Random Forest Regressors emerged as one of the most effective. This particular hybrid model utilized the deterministic foundation of COCOMO II to generate initial baseline estimates and employed Random Forest—a powerful ensemble learning technique—to fine-tune the estimates using historical project data, project-specific contextual variables, and non-linear interactions among multiple factors.

The results were notable: the Mean Magnitude of Relative Error (MMRE)—a commonly used metric for evaluating estimation accuracy—was reduced by up to 30% when using the hybrid model compared to standalone COCOMO II or machine learning models alone. The hybrid approach not only improved estimation precision but also demonstrated better generalization across project sizes and domains, making it a viable option for large organizations managing a portfolio of heterogeneous software projects.

In parallel, the study also explored the integration of agile estimation techniques—such as Planning Poker and story point estimation—with machine learning models trained on sprint data and team velocity metrics. This combination proved particularly effective in highly iterative and rapidly changing project environments, where traditional estimation techniques often struggle to maintain relevance. The fusion of qualitative expert input with quantitative learning models enabled the estimation framework to dynamically adjust to changes in scope, resource availability, and technical complexity, resulting in more real-time, responsive forecasting.

Overall, the findings highlighted a significant shift toward hybrid estimation models as the preferred choice in modern software development settings. By leveraging the structure and interpretability of algorithmic models and the adaptive intelligence of machine learning, hybrid approaches offer a balanced and robust solution for effort estimation in both traditional and agile methodologies.

Figure 1:

Opportunities and Benefits

Improved Accuracy: Especially with AI/ML Models

One of the most significant advantages of incorporating AI and Machine Learning (ML) models into software effort estimation is their ability to dramatically improve the accuracy of predictions. Traditional estimation techniques, such as expert judgment or simple algorithmic models, often rely on static inputs and assumptions that may not reflect the dynamic nature of modern software development projects. In contrast, AI/ML models excel at capturing complex, non-linear relationships between various project attributes, including team performance, coding practices, technological complexity, and customer requirements.

By leveraging large datasets from previous projects, machine learning algorithms, such as Random Forests, Neural Networks, and Support Vector Machines, can learn from past experiences and identify patterns that might not be

immediately apparent through traditional methods. This enables these models to provide highly accurate estimates, even in environments where historical data may be incomplete or where there is considerable uncertainty. Furthermore, these models can be continuously improved as more data is gathered, further enhancing their predictive capabilities over time. This ability to adapt and refine predictions makes AI/ML-powered models highly effective in improving software project forecasting accuracy.

Better Resource Management: Accurate Predictions Lead to Optimal Team Allocation

Accurate effort estimation plays a crucial role in resource management by ensuring that the right amount of human resources is allocated to a project at the right time. When project effort is estimated with precision, it allows for optimal allocation of developers, testers, designers, and other team members according to the workload demands throughout the project lifecycle. Overestimating effort may lead to overstaffing, wasting valuable human resources, while underestimating effort could result in delays and burnout.

With improved accuracy in project timelines and effort estimates, teams can make data-driven decisions about resource allocation. For instance, if a project requires more resources during the development phase and fewer during testing, accurate predictions allow for the dynamic reassignment of team members to meet evolving needs. In addition, AI/ML-based models can predict team productivity trends based on past performance data, leading to more informed decisions about resource scheduling, team composition, and workload distribution. This level of precision and foresight ensures that projects remain on track and within budget while maximizing productivity and minimizing resource waste.

Dynamic Estimation: Hybrid Models Adapt to Changes in Scope or Technology

One of the standout features of hybrid estimation models—which combine traditional techniques with advanced AI/ML approaches—is their ability to offer dynamic estimation capabilities. In today's software development landscape, where agile practices and continuous delivery are increasingly prevalent, project scopes and technology stacks can change rapidly. Traditional estimation methods often struggle to keep pace with such dynamic environments, leading to estimation inaccuracies and project delays.

Hybrid models, on the other hand, are inherently flexible and can quickly adjust to changes in scope, requirements, or technology. For example, when a project shifts to a new framework or adopts a new software development tool, the hybrid model can incorporate this information and recalibrate estimates accordingly. Furthermore, these models can incorporate real-time data, such as team velocity or coding progress, to make ongoing adjustments to predictions. By leveraging the power of machine learning algorithms to continuously learn from project evolution, hybrid models can provide a much more adaptive and responsive approach to effort estimation, helping project managers navigate uncertainty and make timely adjustments.

Risk Reduction: Enhanced Visibility into Project Timelines and Costs

Accurate estimation of effort, time, and cost is instrumental in risk management. Hybrid models, particularly those enhanced by AI and machine learning, provide enhanced visibility into a project's potential risks by offering more accurate predictions and forewarning of potential issues. With reliable estimates in hand, project managers can identify budget overruns, timeline delays, and resource shortages early on, allowing them to take corrective actions before these issues snowball into larger problems.

For example, if a hybrid model predicts a significant discrepancy between the initial project estimates and actual progress, it can flag this discrepancy as a potential risk. By adjusting estimates dynamically, these models provide a more real-time view of project performance, allowing teams to mitigate risks by adjusting schedules, increasing resources, or changing priorities. The ability to foresee issues in advance reduces the likelihood of cost overruns, schedule slippages, and resource misallocations, ultimately improving project outcomes and ensuring that the project delivers on time and within budget.

Tool Support: Integration into Tools Like JIRA, Trello, and Microsoft Project

A significant benefit of modern estimation techniques, particularly AI/ML-based and hybrid models, is their ability to seamlessly integrate with popular project management tools like JIRA, Trello, and Microsoft Project. These tools are widely used in both agile and waterfall software development environments to track progress, assign tasks, and manage team collaboration. By integrating effort estimation models into these platforms, teams gain the advantage of having real-time, data-driven insights directly within their workflow.

For example, with an AI-enhanced estimation model embedded within JIRA, project managers can instantly receive updated predictions about task completion times, resource requirements, and overall project progress. This integration enables a more efficient feedback loop, where the estimation model continuously learns from task completion data and adjusts future estimates accordingly. Similarly, integration with tools like Microsoft Project allows for seamless synchronization of project timelines and resources, enabling dynamic updates to effort predictions as new information is entered. This tight integration streamlines the effort estimation process, reducing administrative overhead and ensuring that project managers have the most up-to-date and accurate information at their fingertips. By connecting estimation models directly to project management tools, teams can ensure that predictions are constantly refined as new data becomes available, helping to maintain accuracy and relevance throughout the software development lifecycle.

Challenges

Data Dependency: Machine Learning Models Require Extensive Historical Data

A key challenge for machine learning (ML)-based effort estimation models is their dependency on large volumes of historical data. These models rely heavily on past project data to train algorithms, identify patterns, and make predictions. Without sufficient high-quality historical data, the performance of ML models can be significantly compromised. Small datasets or incomplete project records may not capture the full spectrum of project variables, leading to inaccurate or biased predictions. This is particularly problematic in organizations or industries where project data may not be readily available or well-organized, or when teams lack a history of similar projects.

Additionally, data variability—such as changes in project size, scope, or technology stack—can create challenges for ML models to generalize across diverse projects. These models perform best when they can recognize patterns across many similar instances, and the lack of consistent historical data can limit their ability to deliver reliable estimates. This dependency on data raises concerns about data quality and the need for data cleaning and preprocessing before feeding the data into machine learning algorithms. Furthermore, the integration of diverse data sources and formats from multiple projects can introduce complexity in terms of data harmonization and ensuring that it is ready for analysis.

Expert Availability: Expert Judgment is Limited by Availability and Bias

Despite the advancements in machine learning and algorithmic models, expert judgment remains a common method for effort estimation, especially in cases where historical data is scarce or when dealing with unique projects. However, expert judgment has inherent limitations that can affect its reliability. One of the most significant challenges is expert availability—access to domain experts with the necessary experience and understanding of the specific project context is often limited. In many cases, these experts may not be available on-demand, especially when tight deadlines require quick estimations.

Moreover, **bias** is a well-known issue in expert judgment. Experts, like all humans, are susceptible to various cognitive biases, such as overconfidence, anchoring, or availability bias (where the expert is influenced by the most readily recalled information). These biases can lead to systematic errors in estimation, resulting in overly optimistic or pessimistic predictions. Experts may also rely on personal experiences that may not be representative of the current project, introducing further inaccuracies. Even with the best intentions, subjectivity and individual interpretation can influence estimations, leading to discrepancies in how different experts estimate the same project. These limitations highlight the need for supplementing expert judgment with objective, data-driven approaches when possible.

Model Complexity: Hybrid Models Can Be Difficult to Implement and Maintain

Hybrid models that combine both algorithmic techniques (such as COCOMO or Function Point Analysis) and machine learning methods offer a powerful approach to software effort estimation, but they come with their own set of challenges, primarily due to model complexity. These hybrid approaches, while more accurate and adaptive, require significant technical expertise to implement and maintain effectively.

For instance, combining algorithmic models with machine learning models often requires customizing machine learning pipelines, integrating multiple data sources, and continuously refining the model based on new data. This involves expertise in both software engineering and data science to ensure that the two models work seamlessly together. The complexity is further compounded by the need to maintain these models over time, as both the software development landscape and machine learning algorithms evolve.

Additionally, as hybrid models evolve and refine their predictions, they require constant monitoring, evaluation, and calibration to ensure their continued accuracy. This can be resource-intensive, requiring regular updates to the training data, continuous fine-tuning of the models, and adapting the system to account for changing project dynamics. As a result, organizations need to ensure they have the proper infrastructure and skilled personnel to manage and maintain these models, which can introduce significant overhead and costs.

Tool Overhead: Integration with Project Management Tools Can Introduce Complexity

While the integration of effort estimation models into widely used project management tools such as JIRA, Trello, and Microsoft Project offers clear advantages in terms of accessibility and real-time insights, it also introduces certain complexities and overhead. First, the integration process can be technically challenging, requiring a high degree of customization to ensure the estimation models align with the project management tools' workflows and data structures.

For instance, in tools like JIRA, which are designed to track individual tasks and issues, the estimation model must be able to interpret the data entered into JIRA and feed it back into the system for continuous updates. This may require building custom APIs or using specialized middleware to ensure smooth communication between the estimation models and

project management tools. Moreover, integrating a hybrid estimation model could increase the computational load, requiring more resources for real-time calculations and updates.

On top of this, the integration process may result in tooling complexity, as project managers and team members may need to learn how to work with both the project management tools and the estimation models. The user interface (UI) and user experience (UX) design must also be adapted to allow stakeholders to easily interpret and act upon the estimated effort predictions without overwhelming them with technical details. This integration effort can lead to additional time and resource investment, especially when tools require frequent updates to accommodate new estimation features or model refinements.

Changing Requirements: Agile Environments Can Invalidate Prior Estimations

In the context of agile software development, project requirements are often subject to change over time. This fluidity poses a significant challenge to effort estimation models, particularly those relying on historical data or fixed inputs. Agile projects are characterized by their iterative nature, where features, priorities, and even project scope may evolve with each sprint or iteration. As a result, earlier estimations may quickly become outdated, rendering prior predictions less useful or even completely invalid.

For example, in an agile environment, requirements may shift based on customer feedback or market changes, leading to a complete overhaul of previously estimated work. While hybrid models incorporating real-time data can help adjust estimations as the project progresses, they may still face difficulties when large scope changes occur, as these shifts may not be adequately reflected in the model's historical training data.

Furthermore, in agile environments, short-term estimations are typically made at the start of each sprint, but these estimates can become increasingly unreliable as the project scope changes or new features are introduced. While machine learning models can adapt to such changes over time, they may struggle to provide accurate predictions when the underlying assumptions of the project have dramatically shifted. To address this, continuous updates and refinements to the estimation model may be required, which can add complexity to the agile process. Therefore, hybrid estimation models need to be carefully designed to remain flexible and adaptive in the face of evolving project requirements.

References

1. Suri, P. K., & Ranjan, P. (2012). Comparative analysis of software effort estimation techniques. International Journal of Computer Applications, 48(21), 12-19.
2. Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2011). Data mining techniques for software effort estimation: a comparative study. IEEE transactions on software engineering, 38(2), 375-397.
3. Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: a comparative study. Neural Computing and Applications, 27, 2369-2381.
4. Tailor, O., Saini, J., & Rijwani, M. P. (2014). Comparative analysis of software cost and effort estimation methods: a review. Interfaces, 5(7), 10.
5. Tailor, O., Saini, J., & Rijwani, M. P. (2014). Comparative analysis of software cost and effort estimation methods: a review. Interfaces, 5(7), 10.
6. Tailor, O., Saini, J., & Rijwani, M. P. (2014). Comparative analysis of software cost and effort estimation methods: a review. Interfaces, 5(7), 10.

7. Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. Information and software technology, 54(1), 41-59.
8. Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An update on effort estimation in agile software development: A systematic literature review. IEEE Access, 8, 166768-166800.
9. Sharma, S., & Vijayvargiya, S. (2022). Modeling of software project effort estimation: a comparative performance evaluation of optimized soft computing-based methods. International Journal of Information Technology, 14(5), 2487-2496.
10. Rashid, C. H., Shafi, I., Ahmad, J., Thompson, E. B., Vergara, M. M., de la Torre Diez, I., & Ashraf, I. (2023). Software cost and effort estimation: Current approaches and future trends. IEEE Access, 11, 99268-99288.
11. Meenakshi, & Pareek, M. (2023, October). Software effort estimation using deep learning: a gentle review. In International Conference on Sustainable and Innovative Solutions for Current Challenges in Engineering & Technology (pp. 351-364). Singapore: Springer Nature Singapore.
12. Sinhal, A., & Verma, B. (2013). Software Development Effort Estimation: A Review. International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), 3(6).
13. MacDonell, S. G., & Shepperd, M. J. (2007, September). Comparing local and global software effort estimation models--reflections on a systematic review. In First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007) (pp. 401-409). IEEE.
14. Popović, J., & Bojić, D. (2012). A comparative evaluation of effort estimation methods in the software life cycle. Computer Science and Information Systems, 9(1), 455-484.
15. Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. (2022). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. Software: Practice and experience, 52(1), 39-65.
16. Saeed, A., Butt, W. H., Kazmi, F., & Arif, M. (2018, February). Survey of software development effort estimation techniques. In Proceedings of the 2018 7th International Conference on software and computer applications (pp. 82-86).
17. Varshini, A. P., Kumari, K. A., Janani, D., & Soundariya, S. (2021, February). Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation. In Journal of Physics: Conference Series (Vol. 1767, No. 1, p. 012019). IOP Publishing.
18. Rastogi, H., Dhankhar, S., & Kakkar, M. (2014, September). A survey on software effort estimation techniques. In 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence) (pp. 826-830). IEEE.
19. Gautam, S. S., & Singh, V. (2018). The state-of-the-art in software development effort estimation. Journal of Software: Evolution and Process, 30(12), e1983.
20. Wu, H., Shi, L., Chen, C., Wang, Q., & Boehm, B. (2016, October). Maintenance effort estimation for open source software: A systematic literature review. In 2016 IEEE international conference on software maintenance and evolution (ICSME) (pp. 32-43). IEEE.