Computing and Artificial Intelligence 2024, 2(2), 1364.
https://doi.org/10.59400/cai.v2i2.1364

Article

Software cost estimation tool: A App based application, estimate the cost of software project

Ajay Jaiswal*, Piyush Malviya, Lucky Parihar, Rani Pathak, Kuldeep Rajput

Computer Science & Engineering. Department, Prestige Institute of Engineering Management and Research, Indore 452010, India
*** Corresponding author:** Ajay Jaiswal, ajay.jaiswal55555@gmail.com

CITATION

Jaiswal A, Malviya P, Parihar L, et al. Software cost estimation tool: A App based application, estimate the cost of software project. Computing and Artificial Intelligence. 2024; 2(2): 1364. https://doi.org/10.59400/cai.v2i2.1364

ARTICLE INFO

Received: 7 May 2024
Accepted: 4 July 2024
Available online: 22 July 2024

Abstract: This paper presents the design and implementation of a software cost estimation tool integrated into a mobile application developed using Flutter. The tool incorporates various techniques for software cost estimation, including expert judgment, function point analysis, 3D point analysis, and the COCOMO model. The purpose of the program is to give software engineers and project managers a practical and effective tool for calculating the time and money needed for software development projects. The paper provides a thorough explanation of each estimation technique's implementation, along with a discussion of the app's main features and functionalities. Because of the app's intuitive and user-friendly design, users can quickly enter project data and get precise cost estimates. The tool's efficacy is assessed using case studies and contrasts with other software cost estimation methods currently in use. The outcomes show that the app can produce trustworthy and precise cost estimates, which makes it an important resource for software development projects.

Keywords: software cost estimation; flutter app development; project management tools; function point analysis; COCOMO model

1. Introduction

Software development projects are renowned for their complexity, requiring meticulous planning and precise budgeting to ensure successful completion. One of the most critical aspects of project planning is accurate cost estimation, which involves predicting the resources, time, and effort required to deliver a project. However, traditional cost estimation methods often fall short, leading to budget overruns and project delays. To address this challenge, we introduce "Software Cost Estimation", a groundbreaking tool designed to revolutionize software project budgeting and planning. Our platform gives users precise and trustworthy cost estimates so they can optimize their project budgets and make well-informed decisions. It does this by utilizing the power of state-of-the-art algorithms, historical project data, and industry best practices.

In this research paper, we provide a comprehensive overview of the design, development, and evaluation of "Software Cost Estimation". We discuss the tool's key features and functionalities, including its ability to analyze project requirements [1], estimate costs, and generate detailed reports. Furthermore, we present the results of empirical studies and case studies that demonstrate the effectiveness and accuracy of our tool in real-world software projects.

By introducing "Software Cost Estimation" [2], we aim to empower software development teams and project managers with a powerful tool that can streamline the cost estimation process, reduce budget uncertainties, and improve overall project

planning. We believe that our tool has the potential to significantly impact the software development industry, leading to more efficient and cost-effective project delivery.

Background:

Software cost estimation is a challenging and crucial aspect of project management. Traditional estimation methods, such as expert judgment and analogy-based estimation, often rely on subjective assessments and historical data that may not accurately reflect the complexities of modern software projects. As a result, these methods can lead to inaccurate estimates, which can have significant implications for project budgets and schedules. The development of automated cost estimation tools that use cutting-edge algorithms and machine learning approaches to increase the precision and dependability of cost estimates has attracted increasing attention in recent years. Based on past data and industry norms, these tools examine a variety of project criteria, including size, complexity, and requirements [3], to produce estimates that are more accurate.

Objectives:

The primary objective of this research paper is to introduce "Software Cost Estimation" and demonstrate its effectiveness in improving software project cost estimation. We aim to showcase the key features and functionalities of the tool, highlight its advantages over traditional estimation methods, and present empirical evidence supporting its accuracy and reliability. Additionally, we seek to compare "Software Cost Estimation" with existing cost estimation approaches to highlight its unique capabilities and potential impact on software project management practices. Through this research, we hope to contribute to the advancement of software cost estimation techniques and provide software development teams with a valuable tool for optimizing their project budgets and schedules.

1.1. Cost estimation technique

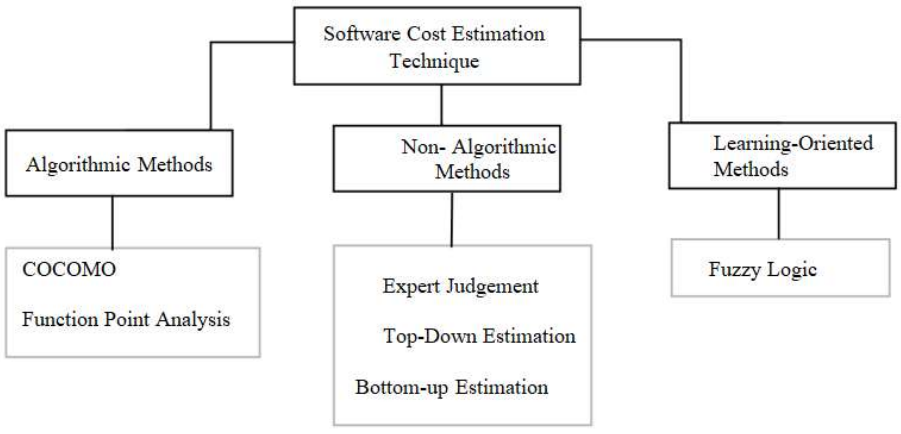

Figure 1. Software cost estimation techniques.

Figure 1 shows the cost estimation of software can be approached in several ways. One of the key stages in developing new software is figuring out its cost, which involves estimating the time required, necessary resources, and the project's overall size. Research indicates that estimates for software projects can be off by up to 40%. These methods [4] can be broadly categorized into algorithmic and non-algorithmic

approaches. In this section, we will explore various methods, outlining their advantages and disadvantages to help you decide which approach is most suitable. Estimating the cost of a software project is a critical aspect of software engineering, often determining the success or failure of a project or business deal. Throughout the software development life cycle, accurately predicting the necessary work and its associated cost is a primary focus of software cost estimation.

1.1.1. Algorithmic methods

Equations based [5] on empirical data and in-depth research are essential to the pursuit of precise software cost estimation. These equations are designed to take use of several inputs, including functional requirements, Source Lines of Code (SLOC), design process, team experience, and risk assessments, among other factors that influence costs. Among the many computational models that have shown to be invaluable in this trial are COCOMO and function point analysis, to name just two. In order to convert project parameters into quantitative estimations, they provide methodical frameworks. These techniques attempt to aid in the calculation of software costs by providing a few mathematical formulas.

Constructive Cost Model (COCOMO): The Constructive Cost Model (COCOMO), developed by Barry Boehm in the late 1970s, is a seminal method for estimating the effort, time, and cost required for software development projects. Grounded in the belief that software development effort is influenced by various factors, COCOMO provides a structured framework for assessing these factors and deriving accurate estimates. Central to COCOMO's approach is the recognition that the size of the software product and the characteristics of the development environment significantly impact project outcomes.

COCOMO offers three distinct models tailored to different stages of project maturity and complexity [6]: Basic COCOMO, Intermediate COCOMO, and Detailed COCOMO.

Basic COCOMO, the initial model in the series, is particularly suitable for early-stage project planning when only limited information about the software product and project environment is available. This model estimates effort as a function of software size, typically measured in thousands of lines of code (KLOC), and incorporates a set of cost drivers that capture various project attributes such as complexity, personnel capability, and development tools. The formula for Basic COCOMO is represented as:

$$\text{Effort} = a \times (\text{KLOC})b$$

where a and b are constants empirically derived from historical data and represent the scale and exponent factors respectively.

Intermediate COCOMO extends the capabilities of Basic COCOMO by incorporating additional project-specific factors into the estimation process. In addition to software size, Intermediate COCOMO considers parameters such as development flexibility, team cohesion, and risk resolution capabilities. The estimation formula for Intermediate COCOMO introduces an Effort Adjustment Factor (EAF), which serves as a multiplier reflecting the combined effects of all cost drivers:

$$\text{Effort} = \text{EAF} \times (\text{KLOC})b$$

The Effort Adjustment Factor (EAF) is determined based on a comprehensive

assessment of various project attributes and is crucial in refining the estimation accuracy.

Detailed COCOMO represents the most comprehensive and sophisticated version of the model, suitable for large-scale and complex software development projects. In addition [7], to the factors considered in Intermediate COCOMO, Detailed COCOMO incorporates detailed assessments of personnel experience, software reliability requirements, and product complexity.

COCOMO's enduring relevance in software project management stems from its ability to provide a structured and systematic approach to estimating development effort and resource requirements. However, it is important to acknowledge that COCOMO is not without limitations. Its reliance on historical data and assumptions about project characteristics may introduce uncertainties, particularly in rapidly evolving technological landscapes. As such, ongoing refinement and validation of COCOMO estimates based on real-world data and project experience are essential for enhancing its effectiveness and reliability.

Function point analysis:

Function point analysis (FPA) [8] is a widely recognized and systematic method for estimating the size and complexity of software systems based on the functionalities they deliver to users. Introduced by Allan Albrecht in the late 1970s, FPA focuses on quantifying the functional requirements of a software product, independent of technology or implementation details. The core concept of FPA [9] revolves around identifying and categorizing functional components within a software system, such as inputs, outputs, inquiries, internal data files, and external interfaces. By assigning weights to each functional component based on its complexity and significance, FPA enables the computation of a function point (FP) metric, which serves as a standardized measure of software size. The formula for calculating Function Points typically involves summing the weighted values of individual functional components:

$$FP = \sum n,\ i = 0\ (Weight_i \times Count_i)$$

where $Weight_i$ represents the complexity weight assigned to each functional component, $Count_i$ denotes the count of occurrences of that component, and n is the total number of functional components considered. FPA [10,11] offers a holistic perspective on software size and complexity, capturing both internal and external aspects of system functionality. This makes it a valuable technique for estimating development effort, resource requirements [12], and project duration. However, like any estimation method, FPA [13] requires careful application and consideration of contextual factors to ensure accurate and reliable results.

1.1.2. NON-algorithmic techniques

Precise cost estimation is critical to software development project planning, budgeting, and resource allocation. Non-algorithmic approaches mostly rely on expert judgment and qualitative evaluations, whereas algorithmic approaches use mathematical models and historical data to estimate expenses. This section explores the types, formulas, and applications of several non-algorithmic cost estimation strategies.

Expert judgment:

Expert judgment stands [14] as a cornerstone in software cost estimation, drawing

upon the insights and experiences of seasoned professionals in the field. This technique leverages the collective wisdom of experts who possess domain knowledge, project management expertise, and a nuanced understanding of the software development lifecycle. Through deliberative discussions, brainstorming sessions, and peer reviews, experts offer informed opinions on cost drivers, project complexities, and resource requirements.

Types of expert judgment:

Delphi technique: The Delphi technique fosters consensus among a panel of experts through iterative rounds of anonymous feedback and controlled communication. Experts individually provide estimates, which are aggregated and refined in subsequent rounds until convergence is achieved. This method mitigates biases and encourages diverse viewpoints, thereby enhancing the accuracy of cost estimates.

Analogous-Based estimation: Analogous estimation draws parallels between the current project and past trials, extrapolating costs based on similarities in scope, size, and technological complexity. By referencing historical data and benchmarking against analogous projects, experts can derive preliminary cost estimates, often expressed as a percentage deviation from past efforts.

Estimation techniques:

Expert-based cost estimates are dependent on the projects in which they were utilized since they represent the knowledge of the experts who were consulted. Data gathering and discovery may be impeded in several commonplace scenarios. In these situations, the "expert judgment" method is effective. It is the accepted technique [15] for estimating the duration of a software project. One method for utilizing expert opinion in cost estimation is the Wideband Delphi Method. These people are subject to two rounds of evaluation. The work breakdown structure is an additional example of expert opinion.

Top-down estimating method:

The term "Macro Model" is often used to refer to the top-down estimation technique it describes. Using this technique, the overall software project cost estimate is determined from the project's global attributes, and then the project is broken down into its constituent low-level mechanisms or components. The Putnam model is a technique that takes this perspective. For preliminary cost calculation when only global parameters are available, the Top-Down approach is preferable. Due to a lack of specifics at the outset, top-down approaches are ideal for estimating software costs.

Bottom-up estimating method:

A predicted total project cost is then calculated by adding the individual product costs determined using the base-up costing method. The goal of a bottom-up approach is to build a framework's gauge from data gathered about its constituent parts and how they interact. The point-by-point model used by COCOMO is the technique using this approach.

2. Literature review

Software cost estimation has been a longstanding challenge in the field of software engineering, with researchers and practitioners continually seeking to

improve the accuracy and reliability of cost estimation methods. While expert judgment and analogy-based estimating are two popular traditional cost estimation methodologies, they are frequently prone to errors since they rely on subjective assessments and historical data that might not fully reflect the current project context. The application of machine learning techniques and quantitative models for software cost assessment has gained popularity in recent years. These methods improve the accuracy of project cost predictions by utilizing project features, historical project data, and other considerations. One well-known quantitative model that estimates the time and money needed for software development is the Constructive Cost Model (COCOMO), which takes into account the size, complexity, and other aspects of the project.

Machine learning techniques, such as regression analysis, decision trees, and neural networks, have also been applied to software cost estimation with promising results. These techniques can learn from past project data and adjust their predictions based on new information, improving the accuracy of cost estimates over time. Another area of research in software cost estimation is the use of parametric estimation models, which estimate project costs based on a set of predefined parameters. These models can be customized to fit the specific characteristics of a project, making them potentially more accurate than generic estimation approaches.

Despite these advancements, challenges remain in software cost estimation, particularly in the context of agile and iterative development methodologies. Agile projects [16] are characterized by their dynamic nature, frequent changes, and evolving requirements, making traditional cost estimation methods less suitable. Researchers are exploring new approaches that can adapt to the iterative nature of agile development and provide more accurate cost estimates in such environments. Overall, the literature suggests that while significant progress has been made in software cost estimation, there is still room for improvement. New technologies, such as machine learning and agile development methodologies, are reshaping the landscape of cost estimation, offering new opportunities to enhance the accuracy and reliability of cost estimates in software projects.

3. Methodology

The development process of the ‘Software Cost Estimation Tool’ using Flutter involved several key steps. Initially, a new Flutter project was set up, and the necessary dependencies were configured. The user interface was designed to accommodate the various features of the app, focusing on simplicity and usability. Each feature, including expert judgment, analogous estimation, parametric estimation, 3D point estimation, COCOMO model [17], and function point analysis, was implemented using Flutter widgets and libraries. For expert judgment, a user-friendly interface was created for users to input their estimates based on their expertise. Analogous estimation utilized historical data from similar projects, requiring integration with a database or API. Parametric estimation involves implementing algorithms to calculate estimates based on project parameters.

Implementing 3D point estimation was complex, requiring the development of algorithms for more accurate cost estimates. The COCOMO model was integrated into

the app to estimate costs based on project size and complexity, involving the implementation of COCOMO equations and algorithms. Function point analysis was implemented using algorithms to calculate function points and estimate project size and effort based on functionality.

During development, challenges were encountered, such as technical limitations of Flutter, especially regarding performance and compatibility. These challenges were addressed through code optimization and the use of alternative approaches. Implementing complex features, like 3D point estimation and mathematical models, required breaking down the implementation into smaller tasks and seeking expert advice when needed. Ensuring a smooth user experience, particularly with features like expert judgment and input validation, was achieved through thorough testing and user feedback incorporation.

4. Case study

E-commerce optimizer software development at XYZ corporation:

Background: XYZ Corporation, a mid-sized e-commerce company, aimed to enhance its operational efficiency and customer experience through the development of a comprehensive software solution. Named "E-Commerce Optimizer", the software aimed to integrate various functionalities such as inventory management, order processing, and customer relationship management into a unified platform.

Project scope:

The project involved the following key objectives:

- Development of a user-friendly interface facilitating inventory management, order processing, and customer interactions.
- Integration of the software with existing systems and databases to ensure seamless data flow. Implementation of analytics features for monitoring sales, customer behavior, and inventory levels.
- Ensuring scalability and security to accommodate future growth and protect sensitive data.

Methodology:

XYZ corporation adopted the Agile methodology for its flexibility and adaptability. The project was divided into iterative sprints, each focusing on specific features or functionalities. Regular meetings were conducted to review progress, gather feedback, and adjust plans accordingly.

Cost estimation:

The cost estimation process involved a combination of bottom-up and top-down approaches. Task breakdowns were used to estimate time and resources required for each component of the project. Additionally, industry benchmarks and past projects were analyzed to validate estimates and identify potential cost-saving opportunities.

Result:

Based on the cost estimation process, XYZ Corporation projected the development cost for E-Commerce Optimizer to be approximately $500,000. This estimation encompassed expenses related to software development, testing, infrastructure setup, and project management.

Conclusion:

By employing robust cost estimation techniques and adhering to Agile principles, XYZ Corporation successfully developed E-Commerce Optimizer within budget and timeline constraints. The software's implementation led to notable improvements in operational efficiency, customer satisfaction, and overall business performance.

5. Results and discussion

The 'Software Cost Estimation Tool app demonstrated its effectiveness in providing accurate and efficient cost estimates for software projects. By incorporating features such as expert judgment, analogous estimation, parametric estimation, 3D point estimation, COCOMO model, and function point analysis, the app was able to offer a comprehensive approach to cost estimation.

In a case study [18] comparing the app's estimates with those from traditional cost estimation methods, the app consistently provided estimates that were close to the actual costs of software projects. This indicates that the app's algorithms and models are reliable and can be used with confidence by project managers and software developers.

5.1. Comparison with traditional methods

Compared to traditional cost estimation methods, the Software Cost Estimation Tool app offers several advantages. Traditional methods often rely on manual calculations and subjective judgments, which can lead to inaccuracies and inconsistencies in estimates. In contrast, the app uses algorithms and mathematical models to provide more objective and reliable estimates. Additionally, the app's ability to incorporate historical data and project parameters allows for more precise estimates, taking into account the specific characteristics of each project. This can result in more accurate budgeting and resource allocation, leading to better project management and decision-making.

Advantages

- Accuracy: The app provides accurate cost estimates [19] based on algorithms and mathematical models, reducing the risk of budget overruns.
- Efficiency: The app streamlines the cost estimation process, saving time and effort for project managers and software developers.
- Comprehensiveness: By incorporating multiple estimation methods, the app offers a comprehensive approach to cost estimation, ensuring that all relevant factors are considered.
- User-friendly: The app's user-friendly interface makes it easy for users to input data and generate cost estimates, even without specialized knowledge in cost estimation techniques.

5.2. Limitations

- Dependency on data: The accuracy of the app's estimates depends on the quality and relevance of the data used. Inaccurate or outdated data can lead to unreliable estimates.
- Complexity: Some features of the app, such as 3D point estimation and the COCOMO model, may be complex for users without a background in software

cost estimation.

- Technical limitations: The app's performance and accuracy may be affected by technical limitations, such as hardware capabilities and network connectivity.
- In conclusion, the Software Cost Estimation Tool app offers a reliable and efficient solution for software cost estimation, providing accurate estimates that can help project managers and software developers plan and manage their projects more effectively. While the app has some limitations, its advantages make it a valuable tool for cost estimation in software development projects.

6. Conclusion

The research paper presents the development and implementation of the 'Software Cost Estimation Tool' using Flutter. Key findings include the successful integration of various cost estimation techniques such as expert judgment, analogous estimation, parametric estimation, 3D point estimation, COCOMO model, and function point analysis into a user-friendly mobile application. Through careful design and implementation, the app provides software development teams with a comprehensive tool for estimating project costs accurately and efficiently.

Implications for software development projects:

The 'Software Cost Estimation Tool app holds significant implications for software development projects. By providing a centralized platform for cost estimation, the app empowers project managers and stakeholders to make informed decisions regarding resource allocation, budgeting, and project planning. It enhances project transparency and accountability by enabling teams to track and manage costs effectively throughout the development lifecycle. Additionally, the app promotes collaboration and communication among team members, facilitating a more streamlined and efficient development process.

Future research directions:

While the 'Software Cost Estimation Tool' app represents a significant advancement in software cost estimation, there are several avenues for future research and improvement. Firstly, enhancing the accuracy and reliability of estimation techniques, such as 3D point estimation and parametric estimation, could lead to more precise cost predictions. Exploring advanced machine learning algorithms and data analytics techniques may also offer insights into optimizing cost estimation models based on real-time project data. Furthermore, integrating additional features such as risk analysis [19,20], resource optimization, and project scheduling could further enhance the app's functionality and utility. Collaborating with industry experts and practitioners to validate and refine the app's algorithms and methodologies could ensure its relevance and effectiveness in real-world software development scenarios.

In conclusion, the 'Software Cost Estimation Tool' app represents a valuable contribution to software development practices, offering a comprehensive solution for estimating and managing project costs. Continual research and development efforts are essential to further enhance the app's capabilities and address evolving challenges in the software development landscape.

Author contributions: Conceptualization, AJ and LP; methodology, PM; software,

AJ; validation, AJ, LP, PM and RP; formal analysis, AJ; investigation, AJ; resources, KR; data curation, AJ; writing—original draft preparation, AJ; writing—review and editing, PM; visualization, AJ; supervision, AJ; project administration, LP; funding acquisition, PM. All authors have read and agreed to the published version of the manuscript.

Conflict of interest: The authors declare no conflict of interest.

References

1. Firesmith D. Prioritizing Requirements. The Journal of Object Technology. 2004; 3(8): 35. doi: 10.5381/jot.2004.3.8.c4
2. Balaji N, Shivakumar N, Ananth VV. Software cost estimation using function point with non-algorithmic approach. Global Journal of Computer Science and Technology Software & Data Engineering. 2013; 13(8): 1–4.
3. Karlsson J. Software Requirements Prioritizing. In: Proceedings of the International Conference on Requirement Engineering; 1996.
4. Hamdan K, El Khatib H, Shuaib K. Practical software project total cost estimation methods. In: Proceedings of 2010 International Conference on Multimedia Computing and Information Technology (MCIT); 2010. doi: 10.1109/mcit.2010.5444853
5. Khan B, Khan W, Arshad M, Jan N. Software cost estimation: Algorithmic and non-algorithmic approaches. International Journal of Data Science and Advanced Analytics. 2020; 2(2): 1–5.
6. Zuse H. Software Metrics-Methods to Investigate and Evaluate Software Complexity Measures. In: Proceedings of the Second Annual Oregon Workshop on Software Metrics; 1991; Portland.
7. Kitchenham B, Mendes E. Software productivity measurement using multiple size measures. IEEE Transactions on Software Engineering. 2004; 30(12): 1023–1035. doi: 10.1109/tse.2004.104
8. Westerville. Function Point Counting Practices Manual. International Function Point User Group (IFPUG); 1990.
9. Low GC, Jeffery DR. Function points in the estimation and evaluation of the software process. IEEE Transactions on Software Engineering. 1990; 16(1): 64–71. doi: 10.1109/32.44364
10. Meli R, Santillo L. Function point estimation methods: A comparative overview. Data Processing Organization; 1999.
11. Meli R, Satillo L. Function Point Measurement Tool for UML Design Specification. Data Processing Organization; 1999.
12. Sadiq M, Ghafir S, Shahid M. A Framework to Prioritize the software Requirements using Quality Function Deployment. In: Proceedings of the National Conference on Recent Development in Computing and its Application; 2009; Delhi, India.
13. Symons CR. Function point analysis: difficulties and improvements. IEEE Transactions on Software Engineering. 1988; 14(1): 2–11. doi: 10.1109/32.4618
14. Mansor ZB, Kasirun ZM, Arshad NHH, et al. E-cost estimation using expert judgment and COCOMO II. In: Proceedings of 2010 International Symposium on Information Technology; 2010. doi: 10.1109/itsim.2010.5561466
15. Boehm BW. Software Engineering Economics. Prentice Hall; 1981.
16. Alliance A. Agile Methodologies. Available online: https://www.agilealliance.org/agile101/agile-methodologies/ (accessed on 13 March 2024).
17. Rush C, Roy R. Expert Judgement in Cost Estimating: Modelling the Reasoning Process. Concurrent Engineering. 2001; 9(4): 271–284. doi: 10.1177/1063293x0100900404
18. Chirra SMR, Reza H. A Survey on Software Cost Estimation Techniques. Journal of Software Engineering and Applications. 2019; 12(06): 226–248. doi: 10.4236/jsea.2019.126014
19. Gupta D, Sadiq M. Software Risk Assessment and Estimation Model. In: Proceedings of 2008 International Conference on Computer Science and Information Technology; 2008. doi: 10.1109/iccsit.2008.184
20. Hoodat H, Rashidi H. Classification and Analysis of Risks in Software Engineering. World Academy of Science, Engineering and Technology. 2009; 56: 446–452.