COMPONENT-BASED SOFTWARE ENGINEERING FRAMEWORK TO INCREASE THE ACCURACY OF SOFTWARE COST ESTIMATION

Shabir Ahmad[*1], **Muhammad Furqan**[2], **Bilal Ehsan**[3]

[*1,3]*Department of Computer Science, Govt. Graduate College of Commerce, Multan, Pakistan*
[2]*Department of Computer Science, NCBA&E, Multan Campus, Multan, Pakistan*

[*1]mianshabbir192@gmail.com, [2]furqankamboh13571@gmail.com, [3]bilal_636@hotmail.com

DOI: https://doi.org/10.5281/zenodo.19046399

Keywords
Algorithm Cost Models, Component-Based Development (CBD), Lines of Code (LOC), Software Cost Estimation, COCOMO, Factor Analysis.

Article History
Received: 15 January 2026
Accepted: 27 February 2026
Published: 16 March 2026

Copyright @Author
Corresponding Author: *
Shabir Ahmad

Abstract
This study explores the identification and assessment of reusable factors and variables that have a major impact on software cost estimation (SCE) by using the framework of Component-Based Software Engineering (CBSE). When used in component-driven settings, traditional estimation methods—in particular, object-oriented methods based on Lines of Code (LOC), function points and class metrics—display significant limitations. Although algorithmic models like COCOMO have been widely used, cost estimation that is specifically suited to CBSE has received little scholarly study. This framework treats time as the primary independent variable and suggests an organized process to identify and statistically validate essential parameters. The study determines the most important factors influencing software cost variation using survey-based data collecting and statistical analysis using SPSS, including factor and communalities analysis.

INTRODUCTION

An essential and crucial step before any software project's design and development is software cost estimation. Since it establishes the estimated cost and completion time, the estimating of a new software system is inextricably tied to its development life cycle [1]. Therefore, the accuracy and dependability of a project's cost estimation procedure are critical to its overall success. Software cost estimating has garnered a lot of scholarly interest in the last 10 years and has become a recurring problem in the software sector. A substantial amount of research over the past three decades has concentrated on machine learning-based algorithms to improve estimation accuracy.

A heuristic optimization method for adjusting the parameters of COCOMO-based models is the Firefly Algorithm. These consist of the IFML user interface model, the original COCOMO model, and additional modifications to the fundamental COCOMO framework that have been suggested in the literature [2]. Many performance measures can be used to assess these models. When compared to other meta-heuristic methods, such as Genetic Algorithms and Particle Swarm Optimization, experimental results show that the Firefly Algorithm achieves a high degree of prediction accuracy with a relatively low error rate.

By methodically reusing pre-existing software components, the development paradigm known as Component-Based Software Engineering (CBSE) makes system construction easier [3]. These elements can be broadly divided into two main categories: infrastructure components and business components.

Currently, there are many different approaches to cost estimating [4]. COCOMO-I and COCOMO-II are two of the most well-known Software Cost Estimation (SCE) methods. Mahmood and Lai [5] presented a more recent contribution in this area. The functional characteristics of freshly constructed systems are the main focus of traditional software cost estimation methodologies.

Strong estimating techniques designed especially for small and medium-sized software development companies are still desperately needed, and several researchers have worked toward this goal. Within this framework, the current study focuses on estimating software costs using Component-Based Software Engineering [6].

Time is considered an independent variable in this study since it is commonly known that the cost of software development is closely correlated with the amount of time needed to finish the project [6]. Since most current SCE approaches rely on algorithmic methodologies, the goal of this research is to achieve more accurate Software Cost Estimation via CBSE by utilizing proven Algorithmic Cost Models. Methodologies including surveys, case studies, and experimental investigations are frequently used to validate study findings.

In order to gather empirical data for the current study, a structured questionnaire was used as part of a research survey. This ensured that the evaluation procedure was in line with the particular requirements and nature of the research problem.

2. RELATED WORK

Diverse approaches have been put forth by different scholars to validate the work, which reflects the ongoing complexity and changing difficulties related to software cost assessment. Despite COCOMO-II's continued prominence and widespread use, a variety of methods were used to increase estimation accuracy and adaptability.

Comparing seven random-effects models for meta-analyses that estimate summary odds ratios has added more methodological rigor [7], bolstering statistical validation procedures in empirical research.

Chidamber, Kemerer, and Li conducted a comparative analysis of twelve object-oriented programming (OOP) software metrics in the 1990s. According to their findings, one of the most important factors influencing project cost is software size, which makes thorough project classification necessary to improve estimation reliability.

To enhance component identification and selection procedures, RE-UML, an extension of the Unified Modeling Language (UML) for Requirements Engineering, was developed [8]. It is possible to identify precise candidate components that meet client criteria by using RE-UML. The authors created a Requirements Analysis and Assessment Process (RAAP) to implement this strategy and methodically support user-centered goals. The restricted integration of Component-Based System (CBS) development phases inside conventional UML procedures, especially during requirements analysis and component selection, was a major driving force behind this effort. The authors contended that by making it easier to identify and improve crucial system components, RE-UML successfully fills this need.

Modern software development techniques have been greatly improved by the Component-Based Software Engineering (CBSE) paradigm. The strategic significance of parameter reusability has been brought to light by comparative studies between traditional development methodology and CBSE approaches. These research presented novel viewpoints on software engineering methods and expounded on the structural and procedural differences between standard models and CBSE frameworks, which are frequently backed by diagrams and conceptual models.

Research on different phases of component-based software simulation was presented at the Intelligent Vehicle Symposium Workshop on Cooperative Interacting Vehicles. It addressed related issues and offered workable solutions. The study underlined how crucial it is to put organizational standards into place in order to guarantee efficient documentation and communication between model developers and software engineers in component-based modeling and simulation systems.

One of the fundamental principles of the Component-Based Development (CBD) approach is reusability. To optimize the advantages of reusable parameters and variables, researchers have examined several component kinds as well as the benefits and drawbacks of various development methodologies. A neural network–based model further reinforced the significance of component reuse in software development [9]. This study showed that characteristics inherent in component-based engineering have a direct correlation with the total cost of a software system. The standardization cost of components used during development was one of the two main cost-driving elements that were found. The authors contended that thorough cost-benefit assessments must account for these standardizing costs.

Further studies have looked at the reusability issues that arise during software development, emphasizing how systematic reuse improves project reliability while also lowering overall development costs. Aspect-Oriented Methodology was also used to examine cross-cutting cost issues [10], especially in relation to system integration involving micro services, Service-Oriented Architecture (SOA), and the Internet of Things (IoT). The study used real-world examples, such as the application of business rules that make use of reusable components, to support its conclusions.

Numerous academics have conducted thorough analyses of current Software Cost Estimation (SCE) methods, pointing out important research gaps and highlighting the need for more study in this area. These studies offered thorough discussions on the development of models generated from COCOMO (I & II), as well as their inherent limits, and methodically assessed the advantages and disadvantages of various estimation techniques. Specifically, several evaluation techniques, such as Mathematical Analysis of the Estimating Equation [11], have been used to examine the limitations of the COCOMO-II model.

To conceive component reusability, a graph-based model was also put out, emphasizing the optimization of crucial variables and parameters to cut down on development time and cost [12]. This approach's main goal was to increase system efficiency through parameter optimization and systematic reuse.

The accuracy of the Component-Based Method (CBM) for software cost estimation has also been assessed empirically. Through project-based analyses, researchers examined the relationship between Lines of Code (LOC) and Number of Components (NOC), employing case studies and statistical techniques to assess estimation reliability [13].

Collectively, these studies show that there isn't a single, broadly applicable approach for software cost estimation in CBSE settings. SCE is still an essential part of software development, nonetheless. Although COCOMO-I and COCOMO-II remain fundamental models in estimating research, their shortcomings further support the need for improved methods designed especially for component-based development scenarios.

3. FRAMEWORK FOR THE EXISTING PROBLEM

In the context of Component-Based Software Engineering (CBSE), the proposed framework seeks to determine and investigate the factors and variables that either directly or indirectly affect Software Cost Estimation (SCE). The authors methodically determined pertinent factors related to software cost estimation in object-oriented programming (OOP) environments that use CBSE principles in order to achieve this goal. An exhaustive review of peer-reviewed international academic articles about SCE in component-based and object-oriented contexts was used to extract these factors. The reference section contains documentation of every study that was consulted.

After the identification of different parameters, a comprehensive survey was conducted to empirically validate their relevance and determine

the significance level of each variable in addressing the research problem. The survey data were subjected to rigorous statistical analysis.
Factor analysis and communality analysis were performed using the Statistical Package for the Social Sciences (SPSS) to evaluate the survey outcomes. These statistical techniques facilitated the extraction of principal factors and the measurement of variance explained by each parameter. The results of this analysis (see Tables 2 and 3 in Section V) provide the quantified significance levels of all identified variables, thereby supporting the proposed framework.

4. RESULTS

SPSS software was used for statistical analysis that produced the research findings.

4.1 Analysis of Factors

The importance and underlying structure of the identified characteristics and variables related to Software Cost Estimation (SCE) inside the CBSE framework were ascertained through the use of factor analysis. This method ensured statistical relevance by making it easier to extract principal factors with Eigen values larger than one. Additionally, by maximizing the variation of component loadings and creating a more understandable factor structure, varimax rotation was used to improve interpretability.

4.2 Communalities Analysis

Communalities analysis was performed to assess the extent to which each variable contributed to the overall factor solution. This method measures the proportion of variance in each parameter explained by the extracted factors, thereby indicating its relative importance and explanatory power. The results obtained from this analysis are presented in Table 2.
The following section provides definitions of all parameters and variables referenced in Table 2.

The Parameters/Variables Used In Table 2			
TRNC	Time required to find a new component	**TPSC**	Task performed for the standardization of components
TIC	Time required to incorporate the component in the project	**TPSCD**	Task performed for the standardization of components in a distributive environment
COC	Complexity of the component	**SLCI**	Standardization level regarding the component interface
LCI	Level of component integration with other components	**SLCF**	Standardization level regarding component functionality
PIC	Performance of component inherited	**CQ**	Component quality
PURC	Percentage of user requirement changes	**CCL**	Component customization level

Table 1: The limitations of Software cost estimation that affects the cost of project

LOC Leung *et al.*, 2004 [14]	Post software cost estimation Language dependent Substantial dead coding and blanks. It does not work for visual languages Counting logical statements is difficult Automation is not widely possible.
SLOC Jones, 2006 [15]	Does not define the relationship between SLOC and end result There is no proper coordination of programs, i.e on change of language, SLOC will also be changed. Not a favorite for developers The language disturbs the size of the project.
SLIM (Liu, Qin, and Robert C. Mintram [16])	Not suitable for small-scale project development The estimate of such a project is quite sensitive in the context of technology. Not Transparent
Function Point (Jones, 2006 [17])	It does not support scientific applications To manage the complex applications, it is impossible to handle It also has some non-standardized variations It does not hold historical data like SLOC Subjective counting (screens, reports, etc) Impossible to automate. poor output time- consuming
Object-oriented Metrics (Ouellet A 2024 [18])	Not support the object-oriented paradigm Impossible to handle the full life cycle related problems Itemize is impossible No rules for the conversions from LOC and function point It does not support the software estimating tools
COCOMO-I (Rankovic N, 2024 [19])	The estimate is difficult. Misclassification of the development mode is impossible Goal achievement depends upon model changing Project cost estimation is not accurate.
COCOMO-II (Bajusova, Det *al.* , 2024 [20])	It follows water fall model This model is in experimental mode and not used all its features till now. Time calculation of small projects is impossible

This research underscores the critical importance of identifying and understanding the independent variables that significantly influence the overall cost of software developed using the Component-Based Software Engineering (CBSE) approach.

Table 2: Communalities Analysis

Variables	Questions	Initial	Extract
TRNC	Time required to find a new component	1.000	**.768**
TIC	Time required to incorporate the component in the project	1.000	**.709**
COC	Complexity of the component	1.000	**.751**
LCI	Level of component integration with other components	1.000	**.738**
PIC	Performance of the component is inherited	1.000	**.721**
PURC	Percentage of user requirement changes	1.000	**.704**
TPSC	Task performed for the standardization of components	1.000	**.689**
TPSCD	Task performed for the standardization of components in a distributive environment	1.000	**.612**
SLCI	Standardization level regarding the component interface	1.000	**.667**
SLCF	Standardization level regarding component functionality	1.000	**.651**
CQ	Component quality	1.000	**.768**
CCL	Component customization level	1.000	**.709**
TPSC	Task performed for the standardization of components	1.000	**.751**
LCCP	Level of customization of Components in the software project.	1.000	**.738**

It should be noted that the related work section has already covered the most of the current algorithmic cost models. In this work, we concentrate on a novel method: use CBSE to identify the different factors involved in software cost estimating (SCE) and assess their significance levels. This shows that comparing the method suggested in this research with the current algorithmic cost estimating techniques covered in the Related Work section is not essential.

From the results presented in Table 2, it is evident that the highest variation, 76%, is explained by the time required to discover components (TRNC). This indicates that TRNC is the most significant variable in the ranking of factors for SCE of projects using CBSE. Conversely, it is observed that the statistically least important variable is COC, as it accounts for only 37.8% of the variation among the variables analyzed.

EIGEN VALUE

Five (05) key parameters are taken from the ranking of parameters and variables for the project's software cost assessment using CBSE. In actuality, it consists of the following: standardization of components in either a distributive or independent environment (CS); user requirements changes in relation to component attributes quality (CURCA); time needed for detailed component analysis (TRCA); customization of inherited components with efficient interfacing (CICEI); and time spent for the component's optimal implementation (TOIC). With twelve items, the five factors explain 69.170 percent of the variation compared to 100 percent. Table 3 shows the percentage of variation explained by each variable together with the cumulative percentage of variation.

Table 3: Variance %age Explained By Each Factor

Component Factor	Initial Eigen Values			Extraction sum of Squares			Rotation Sum of Squares		
	Total	% of Variable	Cumulative % of variable	Total	% of Variable	Cumulative % of variable	Total	% of variable	Cumulative %
CS(F1)	2.284	19.032	19.032	2.284	19.032	19.032	1.956	16.300	16.300
CURCA(F2)	1.846	15.383	34.415	1.846	15.383	34.415	1.731	14.425	30.725
TRCA(F3)	1.612	13.436	47.851	1.612	13.436	47.851	1.598	13.318	44.043
CICEI(F4)	1.447	12.061	59.912	1.447	12.061	59.912	1.512	12.600	56.643
TOIC(F5)	1.204	10.033	69.945	1.204	10.033	69.945	1.503	12.527	69.170

Table 3 demonstrates that the presence of F1 (Factor 1) explained 16.300 percent of the total variation, making it the most highly contributing factor and continuous improvement for ranking for SCE using CBSE F5 (Factor 5) explained only 12.527 percent of the total variation, which is the least contributing factor among the five factors chosen.

Table 4: The logical construct for the total variance derived from twelve parameters/variables

Factor	Number of Variables	Cumulative Percentage
CS (F1)	TPSC,TPSCD	**19.032**
CURCA (F2)	LCI, PURC, CQ	**15.383**
TRCA (F3)	TRNC, COC	**13.436**
CICEI (F4)	PIC, SLIC, LCCP	**12.061**
TOIC (F5)	TIC, SLCF	**10.033**

5. VALIDITY OF RESEARCH

To validate this research, a survey methodology was adopted by visiting several domestic, national, and international software houses, including Soles Technologies, Dream Valley, Info-Tech Solutions, Extreme Logic, Matrix Solutions, Soft Infinity Technologies, BISE, PITB, and Netsol Tech. It is significance noting that during the software development process, the system analysts, IT managers, team leaders, and software developers participated in evaluating the questionnaire.

Several assumptions were made at the outset of the survey:

- IT managers and system analysts are considered the primary IT professionals responsible for evaluating the survey.
- Software developers participating in the survey must have more than three years of professional experience.
- Each software house must meet all hardware and software requirements.
- The software house must possess detailed knowledge regarding software quality.
- The software houses must serve both international clients and domestic or national clients.

6. CONCLUSION

The results indicate that 11 out of the 14 variables or parameters are more than 50% significant in contributing to software cost estimation (SCE) for projects using CBSE. This indicates the successful accomplishment of the main goal of the study. Additionally,

Table 3 shows that the five most important variables/factors account for 64.257% of the variation. Therefore, the statistical analysis validates a 64.257% success rate for this study.

This study's main contribution is the identification of characteristics or parameters that are important for software cost assessment using CBSE.

This research not only determines the level of significance of each variable but also identifies the least significant parameter and provides an overall measure of significance.

This research represents an initial step toward software cost estimation for complex systems using CBSE. The researcher working in the CBSE field are major/primary beneficiary of this framework/study and the professionals working in the software development industry can also be recipients.

REFRENCES

Haleem, M., Islam, M., Ahmed, M. N., Farooqui, N. A., Khan, A. N., Hussain, M. R., ... & Khan, I. (2025). A Critical Review for Software Maintenance Cost Estimation Models: A Data Driven. *IEEE Access*.

Bajusova, D., Silhavy, P., & Silhavy, R. (2024). Enhancing software effort Estimation with self-organizing migration algorithm: A comparative analysis of COCOMO models. *IEEE Access*, *12*, 67170-67188.

Eastman, C. M. (2018).Building product models: computer environments, supporting design and construction. CRC press.

Alatawi, M. N. (2024). Forecasting the software engineering model's effort estimation using constructive cost estimation models. *Iran Journal of Computer Science*, 7(4), 735-754.

Akbar, M. A., Khan, A. A., Mahmood, S., & Mishra, A. (2023). SRCMIMM: the software requirements change management and implementation maturity model in the domain of global software development industry. *Information Technology and Management*, *24*(3), 195-219.

Mansor, Z., Razali, R., & Nazri, M. Z. A. (2025). Advancing Agile software cost estimation through data synthesis: a comparative analysis of five generation techniques. *IEEE Access*.

Mandija, S., Sbrizzi, A., Katscher, U., Luijten, P. R., & van den Berg, C. A. (2018). Error analysis of Helmholtz based MR electrical properties tomography.Magnetic resonance in medicine, 80(1), 90-100.

Nadeem, M., Afzal, H., Idrees, M., Iqbal, S., & Asim, M. R. (2023). A Review Of Progress for Component Based Software Cost Estimation From 1965 to 2023. *arXiv preprint arXiv:2306.03971*.

Anasuri, S., Rusum, G. P., &Pappula, K. K. (2023). AI-Driven Software Design Patterns: Automation in System Architecture. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 4(1), 78-88.

Maidin, S. S., Yahya, N., bin Fauri Fauzi, M. A., & Bakar, N. S. A. A. (2025). Current and future trends for sustainable software development: Software security in agile and hybrid agile through bibliometric analysis. *Journal of Applied Data Sciences*, 6(1), 311-324.

Jackson, D., Law, M., Stijnen, T., Viechtbauer, W., & White, I. R. (2018). A comparison of seven random-effects models for meta-analyses that estimate the summary odds ratio. Statistics in medicine, 37(7), 1059-1085.

Hellwig, Alexander David, Stefan Kriebel, Evgeny Kusmenko, and Bernhard Rumpe. "Component-based Integration of Interconnected Vehicle Architectures." In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 153-158. IEEE, 2019.

Tashtoush, Y., Abu-El-Rub, N., Darwish, O., Al-Eidi, S., Darweesh, D., &Karajeh, O. (2023). A notional understanding of the relationship between code readability and software complexity. *Information*, *14*(2), 81.

Leung, Hareton, and Zhang Fan. "Software cost estimation." In *Handbook of Software Engineering and Knowledge Engineering: Volume II: Emerging Technologies*, pp. 307-324. 2002.

SIDDIQ, Shahida, et al. "Implementation Issues of Agile Methodologies in Pakistan Software Industry." (2019)..

Liu, Qin, and Robert C. Mintram. "Preliminary data analysis methods in software estimation." *Software quality journal* 13, no. 1 (2005): 91-115.

Nadeem, Rana Muhammad, Shabir Ahmad, and Rana Muhammad Saleem. "A COMPARISON OF BIG DATA AND CLOUD COMPUTING APPLICATIONS WITH PROSPECTIVE TO SOFTWARE ENGINEERING." *Science International* 29.3 (2017): 619-619..

Ouellet, A., & Badri, M. (2024). Combining object-oriented metrics and centrality measures to predict faults in object-oriented software: An empirical validation. *Journal of Software: Evolution and Process*, *36*(4), e2548.

Rankovic, N., Ranković, D., Ivanovic, M., & Lazić, L. (2024). AI in Software Effort Estimation. In *Recent Advances in Artificial Intelligence in Cost Estimation in Project Management* (pp. 157-195). Cham: Springer Nature Switzerland.

Bajusova, D., Silhavy, P., & Silhavy, R. (2024). Enhancing software effort Estimation with self-organizing migration algorithm: A comparative analysis of COCOMO models. *IEEE Access*, *12*, 67170-67188.