International Journal of Advanced Research in Engineering Technology & Science
ISSN: 2349-2819
Email: editor@ijarets.org Volume-6, Issue-5 May- 2019 www.ijarets.org

DETERMINE THE EXTENT OF VALUE ADDITION DURING VARIOUS STAGES OF SOFTWARE DEVELOPMENT APPROACH FOR SOFTWARE COST ESTIMATION

Ms. Komal Sharma
Teacher, KV School, Sec - 62 Noida.

Mr. Gyanendra Kumar Shukla
Asst. Prof.,Comm-IT Career Academy
(GGSIP University Delhi)

ABSTRACT

The term "Global software development" (GSD) has become more popular in recent years. Asynchronous communication is being used by a large number of software companies to facilitate collaboration across geographically distributed teams working in various time zones. Estimating software costs for such projects can be challenging due to factors such as the effort expended in team-structure and awareness handoff in the process of creating a software invention construction that can be effortlessly dispersed and reduces cross-tie communiqué, as well as facilitating communication between remote teams working together on consistent parts of the structural design and their day-to-day operations. In this article, we discuss the factors that contribute to the extra costs associated with dispersed development and evaluate the significance of each of these variables in terms of the total cost of the software expansion plan. In order to cater for these new issues, we suggest numerous changes to COCOMO-II, which is the most often used software expansion cost estimation method.

Keywords: *Software, Cost Estimation.*

INTRODUCTION

Because of factors such as the growth of the global software market, the trend toward developing software in countries with lower labour costs, and the increasing complexity and scope of software programme structures, the proportion of developments that are carried out in multiple locations across the globe has consistently increased (Kodmelwar, 2019). Since GSD increases the necessity for growth processes, project-management techniques, construction, excellence, and other associated tools and so on, "Siemens-Corporate-Research", Inc. (S.C.R.), has been conducting research to acquire a better knowledge of the issues and impact of various GSD practises (GSD).

According to a research that was published in 1999 by the Standish Group (Gharehchopogh, 2014), the size of the project or team has a significant relationship with project performance. We think that the physical separation of teams, which happens more often as projects become bigger, is one of the key contributors to this connection. This physical separation necessitates additional activities and effort, such as the construction of teams, the transmission of knowledge for asynchronous associations, the development of a disseminated architecture that reduces the need for communication between sites, and the facilitation of communication between teams that are working on consistent components of the architecture (Sagar et al., 2018; Boehm,

1984; Chirra et al., 2019). Because of this additional work, monumental planning, synchronisation, and control are achieved that are above and beyond what you are responsible for in the day-to-day administration of GSD operations.

The vast majority of enterprises, on the other hand, are teaming up with software development companies located in "Eastern Europe, South America, and Asia" in order to reap the financial benefits of the lower cost of labour that these companies provide. On the other hand, taking into consideration the other elements that play a role in offshore sourcing, this might be deceptive. In this paper, we discuss a method that, when applied to COCOMO cost estimates for GSD, may improve the level of precision achieved. After that, we will provide some suggestions for enhancing COCOMO II. In conclusion, we provide some results, highlighting both limitations in our methodology and possible future directions for study.

After providing a brief introduction to COCOMO II, a framework that is widely utilised for estimating the costs of developing software, we proceed to investigate the factors that contribute to difficulties in distributed software development projects and the degree to which C.O.C.O.M.O. successfully represents those factors. The results of our research may be used to do a cost-benefit analysis when making a decision about whether or not to centralise or decentralise software development. The main goal of our investigation into these topics is to provide managers with decision-making frameworks that they may use when they are faced with issues of this kind.

The proliferation of transnational software projects has been helped forward by globalisation. Nevertheless, a variety of research predict that the number of offshore projects will increase throughout the course of time (Khan et al., 2021). It is anticipated that the number of global software development (GSD) projects carried out in countries such as India and China would increase by 20 to 30 percent. Due to the much reduced cost of labour in Asia and Eastern Europe, a number of Western software businesses have relocated their operations outside of the West (Chirra et al., 2019). GSD is being used by companies for a number of reasons, including the improvement of speed-to-market as a result of variances in time zones and the hiring of highly experienced virtual teams.

Estimating costs is one of the parts of project management that managers working on various projects find to be one of the most difficult. In the case of GSD, it takes on an even greater level of significance owing to the fact that task and forecast resource allocation is much more challenging as a result of the dispersion of the resources (Kodmelwar et al., 2019). In the present state of the art, additional cost drivers are not taken into consideration, which prevents the anticipated accuracy for GSD from being improved. The objective of this article is to categorise the cost drivers that have an impact on the method of cost approximation that GSD utilises in this respect. When these cost factors are evaluated, it may assist practitioners cope with them and enhance GSD's cost estimate approach.

This research is predicated on an investigation that sought to discover which method for predicting software expenses is the most accurate, as well as the issues that are associated with using different methods. Determine also what factors are responsible for the inaccurate cost estimating tactics that have been used. Why aren't they able to get a cost estimate that is correct using these methods? The identification of GSD cost factors that have an impact on the estimation process is our primary objective.

This article tackles the lack of knowledge and research on the cost factors that may effect a GSD project, which prevents project managers from correctly predicting the cost of any software project. This lack of

information and research is what this paper addresses. It is the purpose of both this paper and the overall project to identify the most effective methods for accurately estimating the costs of software projects and for reducing the amount of money spent on software projects during the development phase. This will be accomplished by using a hybrid approach to find a solution to the problem of estimating software costs in the context of global software development. With the help of COCOMO II, we will look at a number of different approaches to estimating how much the programme will cost. The COCOMO II model will also be used by our team in order to plan the design and implementation phase of a software cost estimate.

TYPES OF SOFTWARE COST ESTIMATION TECHNIQUES

The majority of cost estimate approaches may be broken down into two categories: algorithmic and non-algorithmic. To calculate software cost estimates, algorithmic approaches employ a formula. The formula is developed with the use of models that include many cost considerations. In addition to that, the statistical method is used whenever a model is being constructed. Techniques that aren't based on algorithms don't use any kind of equation to estimate software costs.

Several different software cost estimating methods were advocated in a number of research. There have been many models that have been projected and constructed in order to find alternatives, enhance current models, or support existing models. The algorithm-based technique was one of the initial ways that researchers for S.C.E. created and it was one of the methods that was used (Hasanluo et al., 2016). Such methods first appeared in the late 1970s and include the pattern-based COCOMO II model, the Checkpoint model developed by Jones in 1997, the SLIM model developed by Putnam and Myers in 1992, and the PRICE-S model developed by Park in 1989 (Hasanluo et al., 2016). However, these techniques do not take into account a variety of factors that affect cost estimation, such as the amount of time and effort that is required (Khan et al., 2021). Numerous researchers have spent a significant amount of time investigating the problem of estimating the number of resources required for software development over the years. As a result, the field of software engineering has witnessed the introduction of non-algorithm techniques, such as machine learning algorithms, in an effort to reduce the impact of this issue through the utilisation of machine learning algorithms (Hasanluo et al., 2016). Despite this, a large number of potential solutions have been proposed for the developing issue.

The goal point counts seem to be a more compatible and more often occuring previous quantity of software size than the foundation lines of code, as stated by Low and Jeffery in 1990 (Low et al., 1990). The proper implementation of function points requires careful consideration and consideration of all relevant factors in order to get optimal results. The most significant of the earlier work in software effort evaluation, as created by Mukhopadhyay and Kekre in 1992, has placed a tremendous amount of significance on lines of code (L.O.C.) or other design aspects as major factors for cost models. This is the conclusion that they came to. The estimated size will be used in conjunction with the models that are already in place to provide early evaluations of the amount of work required for development.

To find a solution to the issue posed by S.C.E., researchers such as Gharehchopogh et al. turned to the hybrid COCOMO II model that incorporates fuzzy logic. The work that has been done up to this point demonstrates that fuzzy systems can be merged with triangle, trapezoidal, Gaussian, and extensive Bell intermediate COCOMO II functions. For the purpose of evaluating hybrid techniques, assessment criteria based on these models, such as M.M.R.E., M.R.E., PRED, MARE, V.A.F., VARE, and B.R.E., are used. The findings that

were obtained using the hybrid COCOMO-II model have showed superior performance when compared to the results generated by the COCOMO II model.

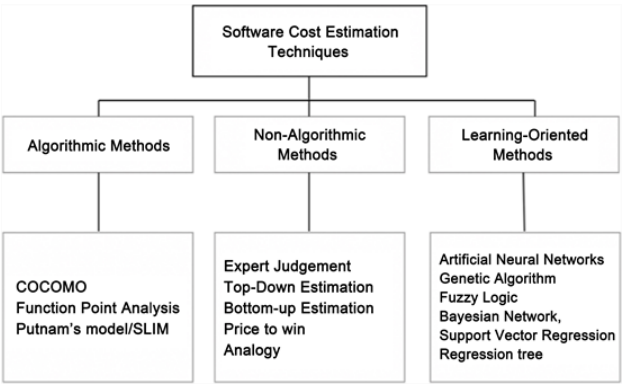


FIGURE 2

FURTHER TYPES OF SOFTWARE COST ESTIMATION TECHNIQUES

The fuzzy logic model was used in order to solve the challenge of estimating the cost of software. As part of this process, the parameters of the COCOMO II model were first converted to fuzzy numbers, and then the numbers were emitted from the fuzzy state. The suggested technique was successful in producing a comparison with both the COCOMO II and AlaaSheta models. The comparison suggests that the performance will be enhanced using the technique that has been presented. In addition to COCOMO II and AlaaSheta, the suggested approach had a greater marginal error of M.MR.E., PRED (N), and V.A.F. in its results. Several different PSOs were used in the research study that was conducted by Reddy et al. in 2011 (Reddy et al., 2011) in order to improve the COCOMO II model parameters. Independent trials of the suggested paradigm were conducted for both small and large-scale projects. As a result of the findings, the MARE percentages for the microprojects in the COCOMO II model and the model that was recommended are 16,1306.0% and 9,0143.0%, respectively. In addition, the MARE percentages for COCOMO II and the model that was recommended were, respectively, 18.1548 and 20.9717%. Based on the findings, it was determined that the model had superior performance when compared to the COCOMO II models.

Cost Drivers

These are the several kinds of cost drivers that have the potential to influence the S.C.E. The company's current economic climate is a critical factor in determining the structural cost drivers. It also relies on the magnitude and breadth of the organisation, both of which might have an effect on the S.C.E. Executioner or operational cost drivers are the elements that impact a company's cost position and are based on the company's ability to effectively "execute" its operations or activities. These cost drivers may also be referred to as cost drivers for operations. Using the results of the I.M.V.P. survey, we may do an analysis of the following three important structural cost factors: automation, plant size, and product mix complexity (Jain et al., 2018). The resource cost driver is what determines how many resources an activity uses up in the course of its execution. It is used in the process of allocating the cost of a resource to an activity or cost pool. The Activity Cost Driver is a metric that is used to assess the regularity and magnitude of the demand that is placed on activities by cost objects. It is used in the process of associating costs with cost items on the basis of activities. The activity cost driver has an impact on the fixed costs of personnel and maintenance, as well as other variable costs. ABC is a subfield of management accounting that assigns indirect costs, sometimes

known as overheads, to activities. ABC necessitates the use of cost drivers (Ziauddin et al., 2013). All of these different cost factors may have an effect on the S.C.E., which can then cause the cost to either go up or down. However, according to the GSD researchers who are looking for the most effective way to minimise costs while still serving the organisations' needs,

Review of Traditional Software Cost Estimation Methods

Estimating how much money should be spent on a software project is one of the most challenging components of project management. This amount should be tracked throughout the project's life cycle and iterated at specified milestones. Because it is more difficult to estimate software costs in the absence of any sample data at the outset, it is possible for us to obtain real data from previous projects and have a reliable estimate based on several and old projects in certain circumstances. This can be beneficial because estimating software costs would be more challenging. At the beginning of the project process, several of the criteria that are outlined in the first two tables, such as the size of the database and the complexity of the functions' source line of code, as well as the capability of the developers to address them, are not apparent and are not tangible.

C.O.C.O.M.O.

The Constructive Cost Model was first conceived of by Barry Boehm in 1981 (Rahikkala et al., 2018). (C.O.C.O.M.O.). COCOMO II, a more advanced version, was released in the year (1985), and it has been developed and improved ever since (Ali et al., 2019). At the present, the technique known as COCOMO-II is the one that is used by the most people in order to forecast the expenses of software projects. It is used in businesses of varying sizes and types throughout the whole of the globe. In a manner similar to that of the Function Point approach, COCOMO II is a functionally explicit algorithmic estimating model. It establishes a connection between the free variables and the dependent variable, which is measured in person months. The overall costs of the project are determined by multiplying the number of person-months worked on the endeavour by the overall quantity of person-months worked. The formula in its most fundamental version is as follows:

$$Effort = A \cdot x \cdot (Size)^{B} \qquad (1)$$

K.D.S.I. is the measurement system that is used ("kilos. of delivered. Source. Instructions."). The coefficient A is given a value of 3.0 so that COCOMO-II may be standardised to the one and only C.O.C.O.M.O. scheme database. The scaling factor, which is denoted in the equivalency by the exponent B, reflects the economies and inefficiencies of scale that arise when the breadth of a software plan expands. When B is equal to one, the scale economies and scale diseconomies are balanced, and linear extrapolation is the outcome. This approach is used for determining the cost of less ambitious undertakings. However, in the vast majority of cases, it is anticipated that 0 B will be greater than 1, which indicates that the project is plagued by scale diseconomies. When the scope of a project increases, the amount of time and effort required for planning, communication, and integration also increases (which can be reduced to some extent by initial risk management, using methodically authenticated design stipulations, or steadying requirements). If B is equal to one, the project benefits from the financial wisdom of scale as a consequence of specialisation advantages. This means that increasing the size of the project by a factor of two requires less labour than increasing the amount of effort by a factor of two. When looked at from a closer perspective, B may be broken down into the following additive factors:

$$B = 1.01 + 0.01 \sum_i W_i \quad (2)$$

Where Wi is shorthand for the five criteria that came before it: progression flexibility, architecture and design resolution, team cohesiveness, and process maturity. The basic person-month count that was acquired It is possible to significantly enhance estimates by multiplying them by a variety of cost components that are together referred to as effort multipliers (E.M.). They are estimated independently, and then their totals are summed, which results in:

According to the information shown in the table, every extra GSD operation has to be accounted for by a separate project factor called Multi-site Expansion. This cost driver is determined by being centred on two factors, which are multi-site apposition (ranging from entirely collocated to international) and multi-site infrastructures (varieties from some mail to collaborating multimedia). Because of this, it's possible that we won't have enough information to get a more accurate cost estimate for GSD projects when we go on to the next part. For instance, the making selection has a significant influence on the amount of effort required and the cost of the project since it dictates the distribution of work between locations as well as its direction. One of the factors that contributes to multi-site expansion does not have this characteristic.

OBJECTIVES

1. To study the extent of value addition during various stages of software development
2. To study approach for software cost estimation

METHODOLOGY

One possible explanation for the reason for the creation of such a methodology is because the process of developing software does not now make use of any strategies that provide an estimate of the total cost. The majority of today's submissions employ strategies for approximate cost estimation that are easily available as stand-alone solutions. The vast majority of the ways, in addition to the perimeters that are used in those approaches, are organization-exact. There is a need for a strategy that is both more complete and integrated, and which integrates the cost estimating methods that are the most generally recognised and used. In this usage, it refers to the whole of the software life cycle, making it a more general term. To enhance project planning and scheduling, the approach advises using a succession of cost estimating tools in an integrated, step-by-step way. This is done in accordance with the technique.

Step 1: Determine the Size of the Project

The primary objective of this stage is to formulate an estimate for the scale of the software product that is being considered (i.e., the number of non-commentary basis declarations). For the purpose of size estimate, the approach suggests using the Function Points and Conversion of Function Ideas into N.C.S.S. procedures, both of which were detailed in the preceding sections. There are a variety of other, more distinctive approaches to size estimation that may be used. The methods used and the stage of development both have a role. The principal outcome of this stage is the N.C.S.S. estimate for the software product that will be built in the next step.

Step 2: Calculate the Cost of Effort and Time

It has been suggested that COCOMO II and its derivatives be used to estimate the amount of work required and the amount of time necessary, expressed accordingly in terms of work months and months.

Step 3: Spread out the Exertion and Time Costs across the Life Cycle

For distribution, various approaches that are built on engagement in the environment of the so, the manufacturing process may be utilised, along with some strategies such as those presented in the COCOMO II methodology.

Step 4: Convert the Costs to Calendar Time

The approach need to include a few criteria for the use of additional growth employees and system forecasters. It is anticipated that the outputs would include monetary charges and calendar periods."

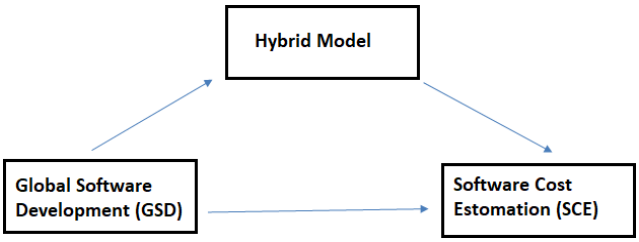
After Step 4 has been completed, one of the automated project administration applications may be used to organise and administer the project according to its timetable. An estimation method is considered to be effective when it is easy to understand, when it can be modified as the programme is being expanded, and when the early evaluation is within +/- 30 percentage points of the cost that will definitely be incurred. As a consequence of this, the process of cost estimating has to be carried out several times during the development stage. It is recommended that this be done at a number of different stages during the development process. Following each iteration, the estimated results should be improved by being associated with prior estimates and being brought up to date. The processes involved in spreading the cost estimate approach may be different based on a number of factors like the development model and/or methodology that was applied, the scope of the software innovation, and so on. The next part will discuss how the technique may be used in a broad sense throughout the process of developing software while adhering to Yourdon's Structured Design/Development approach.

The data set produced by the NASA software project serves as a standard for evaluating the strategy that has been presented. It is made up of three components: Developed Lines (DL), Methodology (M.E.), and a dependent component called Software Cost Estimation, which is going to be projected or estimated (S.C.E.). The L.O.C. (lines of code) value of the programme is used to compute the DL feature, which takes into consideration both lines of code that have been implemented and lines of code that have been reused. This allows for a more accurate estimation of the size of the software. It is important to note that the DL measurement takes into consideration comments in addition to counting re-used lines at a rate of 20%. (Litoriya et al., 2012). The ME functionality is supplied in a manner that is consistent with the approaches that are used in the creation of software projects. When determining the value of the ME characteristic, the following considerations are taken into account: Using official exam programmes, becoming certified, and being trained are all options (Litoriya et al., 2012).

In order to solve the issue of software cost estimation (S.C.E.), the morphological rank linear hybrid was developed. The technique of design, also known as M.R.L.H.D., is being planned. It makes use of an evolutionary search approach (Chen et al., 2005) to track down the components necessary for the building of an accurate system. I the fundamental data needed to approximation the cost (the dimensionality or number of structures required to label the data), and (ii) the model construction, parameters, and training procedure. I the fundamental data needed to approximation the cost (the dimensionality or number of structures required to label the data). Examining the data that has the fewest possible dimensions is necessary since the structure

of the model has to be as straightforward as is humanly possible.

According to Whigham et al. (2015), the technique that has been designed makes use of a modified genetic algorithm (MGA) to construct morphological-rank-linear (MRL) perceptrons (which employs optimal genetic operators to accelerate convergence). The M.G.A. is used to calculate the exact dimensionality of data and the correlating specific characteristics to show the information (first, a maximum dimensionality (MaxDim) is classified by the number of individual functionalities, and then the M.G.A. can select any significance in the span (Chalifelu et al., 2012)for each element in the community), as well as to search for the right initial MRL back propagation parameters. This was first described in Sarro



Framework

FIGURE 4

RESULTS

HYBRID MODEL OF SOFTWARE COST ESTIMATION

There are two hypotheses of this research:

H1 Global software development impact on software cost estimation positively.

H2 Global software development impact on software cost estimation, mediation role of hybrid model.

The framework outlines how we will examine the flow of research as well as which elements influence which other ones. Using the structure that has been provided, we are going to develop a hypothesis, and then we are going to test both the framework and the hypothesis to see whether they are sound. In this conceptual framework, we provide the concepts of an independent variable and a dependent variable, as well as the concept of mediation between the two sets of variables. The first variable is called GSD, which stands for "Global Software Development." GSD is an independent variable, and we are going to determine the effect that GSD has on the other variables. S.C.E. (Software Cost Estimation) is the dependent variable that has the potential to influence GSD; alternatively, we might argue that GSD has the potential to alter the scenario. There is one mediation that can be performed, and it is possible that the influence on S.C.E. values will shift as a result of the mediation performed using the Hybrid model. As a result, we will validate the framework by developing a questionnaire and having the respondent answer the questions in the questionnaire. By providing us their ideas, respondents will be able to tell us the influence that these factors have.

With the help of this structure, we come up with two theories. The first hypothesis is that we will test whether or not the influence of global software development has a beneficial effect on the cost of software estimating. The question now is whether or not the GSD has resulted in a favourable adjustment in the calculation of the

cost of the programme. We are going to look at the beneficial ramifications of this development. In the second hypothesis, we will test whether or not GSD has an effect on S.C.E. by examining the function that the hybrid model plays as a mediator. Does the use of the hybrid model have any impact on the accuracy of the cost estimate, or does it not?

In the framework of GSD, overhead expenses have been singled down as the primary area of focus. Following that, we executed a pre-planned S.L.R. in order to create GSD-specific cost estimation models and cost drivers. The findings that were produced using SLR will then be corroborated by data that is accessible to the public (GSD-based). As a result of this, we have placed an emphasis on the critical factors that drive costs by carrying out a number of statistical studies. In the end, we presented a conceptual model based on the findings that we obtained. The ultimate goal of this study, as well as the path that it will go in the future, is to formalise the suggested model in order to provide practitioners with assistance in examining further cost assessment concerns.

Table 1 SURVEY OF CONDUCTED RESEARCH IN VARIOUS REPOSITORIES

Databases	Papers extracted throughSearch Terms	Papers extracted based on introduction and conclusion	Papers extracted based on title and abstract	Papers extracted based on full texts
Google Scholar	213	150	197	50
IEEE Xplore	189	112	98	70
Research Gate	300	198	223	90
ACM	178	134	101	39
Springer Link	183	126	112	57
Science Direct	90	97	76	88

In the study, the inclusion and exclusion criteria were utilised to choose the articles. This allowed the researchers to exclude any articles that did not meet the needed qualifications, such as those that presented material in an unclear manner. The suggestions that Kitchenham made in 2004 served as the basis for our decision-making over who to include and who to exclude in our study (Kitchenham, 2004). The study used Kitchenham's inclusion criteria (Kitchenham, 2004), which is shown in the table that follows. This criterion was utilised while selecting the feasible publications.

Table 2

INCLUSION CRITERIA

1.	The selected primary study should be a journal, conference or book chapter

2.	The studies that focused on cost estimation in the GSD context
3.	The studies that discuss the challenges or cost drivers affecting cost estimationin GSD
4.	The studies that present GSD specific cost estimation models or techniques
5.	The selected studies must be available full-text articles, specifically in theEnglish language
6.	Most of the studies published between 2010-2021

In addition, the exclusion criteria that were utilised to exclude the studies and literature that were used and adopted by the study are shown below in Table 3, which can be found further down the page.

Table 3 INCLUSION CRITERIA	
1.	The studies that do not answer the research questions
2.	The studies that do not discuss the cost estimation process in the GSD context
3.	The studies that do not highlight the challenges or cost drivers of costestimation in GSD
4.	The studies that do not highlight the various cost estimation models
5.	The studies that were not written in English

An online questionnaire was developed in order to acquire the industry perspective on the cost factors that were extracted from the literature. The poll was aimed primarily at the project managers employed by multi-national software corporations as its target respondents. There were a total of 175 project managers involved, and they were all employed by global corporations. In addition to that, the survey for determining the validity of the results was sent out to over 20 different countries. For the questionnaire, each identified cost driver employed a five-point Likert scale, ranging from "Strongly agree" to "Neutral" to "Disagree" to "Strongly disagree." The findings were first expressed as percentages, and after that, they were improved with the use of several data analysis techniques.

TABLE 4

DEMOGRAPHIC CHARACTERISTIC OF THE SAMPLE GENDER

	Frequency	Percent	Valid Percent	Cumulative Percent

	Male	50	100	100	100
	Female	0	0	0	00
	Total	50	100.0	100.0	

CONCLUSION

Global Software Development (GSD), which consists of teams that are dispersed around the globe and work in collaboration with other businesses, is gaining popularity in spite of the time and space differences that exist between these teams. The most important advantage of GSD is that it enables more human resources to be made accessible at more affordable prices. On the other hand, the distance that exists between different development teams is the root of various problems. When software development teams are dispersed around the country, it is more difficult to coordinate their efforts and communicate effectively, which might result in hidden expenses. As a direct consequence of this, the work estimating models that GSD uses for collocated software are inadequate. Because of this, determining how much work goes into GSD becomes a significant area of study. During the course of the last ten years, the estimation of labour in GSD has been the primary focus of a number of studies. This paper offers the results of a complete literature analysis, which includes an investigation of unsolved research concerns in GSD effort assessment as well as a summary of the expenditures concealed in GSD with and without a hybrid model.

REFERENCES

[1]. Ahmad, J., & Khan, A.W. (2020). Software outsourcing quality evaluation management model (S.O.Q.E.M.M.). 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET).

[2]. Ahmad, J., Khan, A.W., & Qasim, I. (2018). Software Outsourcing Cost Estimation Model (S.O.C.E.M.). ASystematic Literature Review Protocol, 2(1).

[3]. Ali, A., & Gravino, C. (2019). A systematic literature review of software effort prediction using machine learningmethods. Journal of Software: Evolution and Process, 31(10).

[4]. Arslan, F. (2019). A review of machine learning models for software cost estimation. Review of ComputerEngineering Research, 6(2), 64-75.

[5]. Attarzadeh, I., & Ow, S.H. (2011). Improving estimation accuracy of the COCOMO II using an adaptive fuzzy logic model. 2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011).

[6]. BaniMustafa, A. (2018). Predicting software effort estimation using machine learning techniques. 2018 8th International Conference on Computer Science and Information Technology (C.S.I.T.).

[7]. Baskeles, B., Turhan, B., & Bener, A. (2007). Software effort estimation using machine learning

methods. 200722nd international symposium on computer and information sciences.

[8]. Betz, S., & Mäkiö,, J. (2007). Amplification of the COCOMO II regarding offshore software projects,. Offshoring Softw. Dev. Methods Tools Risk Manag.

[9]. Blaifi, S., Moulahoum, S., Benkercha, R., Taghezouit, B., & Saim, A. (2018). M5P model tree based fast fuzzymaximum power point tracker. Solar Energy, 163, 405-424.

[10]. Boehm, B.W. (1984). Software engineering economics. IEEE Transactions on Software Engineering, SE-10(1), 4-21.

[11]. Budgen, D., & Brereton, P. (2006). Performing systematic literature reviews in software engineering. Proceedings ofthe 28th international conference on Software engineering.

[12]. Chen, Z., Menzies, T., Port, D., & Boehm, B. (2005). Feature subset selection can improve software cost estimation accuracy. Proceedings of the 2005 workshop on Predictor models in software engineering - PROMISE '05. Chirra, S. M., & Reza, H. (2019). A survey on software cost estimation techniques. Journal of Software Engineering