Journal of Software Engineering and Applications, 2017, 10, 824-842
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

# Software Cost Estimation Approaches: A Survey

**Ismail M. Keshta**

Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
Email: ismailk@kfupm.edu.sa, dr.ismail.keshta@gmail.com

**How to cite this paper:** Keshta, I.M. (2017) Software Cost Estimation Approaches: A Survey. *Journal of Software Engineering and Applications*, **10**, 824-842.
https://doi.org/10.4236/jsea.2017.1010046

**Received:** June 26, 2017
**Accepted:** September 25, 2017
**Published:** September 28, 2017

## Abstract

The software cost estimation aims to predict the most realistic effort that is required to finish a software project and so it is critical to the success of a software project management. A Software Cost Estimation affects nearly all management activities, including project bidding, resource allocation and project planning. It is affected by a number of factors, such as implementation efficiency, as well as how much the various reviews and studies completed prior to the software development stage cost. Accurate cost estimation will help us to complete the project on time and within budget. Accurate estimation is important because it has led to extensive research into the methods of software cost estimation. Some important software cost estimation methods have been studied in this research work. In addition, we have set out own criteria, which has been used to compare all the different selected methods. We have also given a score for each evaluation criteria, so that we can compare the different methods numerically for cost estimation. Our observations have shown that it is best to use a number of different estimating techniques or cost models, and then compare the results before determining the reasons for any of the large variations. None of the methods are necessarily better or worse than the others. We found, in fact, that their strengths and weaknesses often complement each other. Therefore, the main conclusion is that there is no one single technique that is best for every situation, and the results of a number of different approaches need to be carefully considered to discover what is the most likely to produce estimates that are realistic.

## Keywords

Software Cost Estimation, COCOMO Model, Parametric Models, Putnam Model

## 1. Introduction

Estimating the costs of software projects is a critical activity that requires the use of both proper methods and techniques in order to achieve a good estimation of the results. This is a challenging task that poses many obstacles. The size of the software and its accuracy has a great effect on the estimation's accuracy. Project management also plays a vital role in the guidance of these estimation processes. Much research has been carried out that reflects the rising demands of high-quality software through effective cost estimation [1] [2] [3] [4] [5].

Software engineers have to apply the theories, tools and methods in a software project in order to solve a problem. However, they must also work within the financial constraints that were predefined. A vital issue which is closely related to a software project's financial aspects is the accurate estimation of the software cost involved. This helps to manage any software project as it means it will be within the set budget [6] [7] [8] [9].

Software cost estimation is a very challenging activity in the project management of software because predicting the cost is a difficult process at the early stage of the software's development [4] [10]. Moreover, the estimation of the software's cost is impacted by many factors, including the implementation's efficiency, and the number of reviews and studies done prior to the development stage cost. There is clearly a strong relationship that exists between the estimation of software effort and software cost, as it can be said that the effort is the primary driving factor for the software's cost [11] [12] [13] [14].

It is important to state that the estimation of the software's cost is a continuous activity that begins at the proposal stage and then carries on throughout the life of the project. When project cost management is calculated, it includes the processes that are required to ensure the project is finished within the approved budget. The main processes include [15] [16]:

- Estimating the costs (including top-down and bottom-up estimates, parametric modelling, etc.);
- Determining the budget (the cost baseline);
- Controlling the costs (Earned Value Management (EVM)).

Many approaches have been designed to address this software cost estimation process, which have been proposed by both scientists and researchers trying to create an accurate cost estimation technique that is accurate. The research work gives an extensive overview. It will address a total of five fundamental software cost estimation approaches, and a comparison will be made between the approaches based on evaluation criteria. This will then be thoroughly examined and used throughout the research work.

It is essential to point out that the novelties of this work include: studying important software cost estimation methods, setting out basic criteria (*i.e.*, ease of use, adaptability, accuracy, consistency, interpretable, automatable, tool supported, empirical validations, sensitivity, and handling imprecision and uncertainty), comparing selected cost estimation methods based on these evaluation criteria and giving a score for each evaluation criteria in order to compare the different methods numerically for cost estimation. Moreover, this work provides the implementation for one of the well known software cost estimation models that indicate both the time and the effort required to complete a software project of a specific size.

The paper will be structured as follows. Section 2 provides a list and summary of some of the existing approaches to software cost estimation. Section 3 lists the evaluation characteristics and Section 4 provides both discussions and comparison between the different approaches. Finally, Section 5 gives the conclusion and also future directions.

## 2. Software Cost Estimation Approaches

We will list and summarize some of the existing approaches to software cost estimation throughout this section. For each approach, we will also describe the mechanisms and features. We can divide Cost Estimation Techniques into two main broad categories. These are those that utilize the source lines of codes (SLOC) as their input and others that do not.

### 2.1. Approach 1: Constructive Cost Model (COCOMO)

It is considered that COCOMO is a very important model that can calculate a software cost estimate.This uses an algorithmic formula in order to estimate the software's cost [17]. Therefore, this model is based on both mathematics and a number of experimental equations. Barry Bohem proposed it in 1981 for software cost estimation. It is considered to be the most complete approach and is better documented than the other cost estimation model, as is indicated in the references [18] [19] [20] [21]. In addition, many of researchers in the software engineering field are now trying to increase the efficiency by keeping the COCOMO model's base. Furthermore, due to this model's simplicity, it is usually used for algorithmic cost estimation technique. COCOMO is made up of three models, which are Basic, Intermediate and Detailed.

The first model, which is the Basic one, is used as a function of the program size in the computing software effort and cost. It is primarily used for small to medium-sized software projects in order to perform a speedy, rough estimation. Basic COCOMO is, therefore, considered to be effective in circumstances where only a rough effort estimate is required. The equation for estimating the software effort for this basic model is:

$$\text{Effort} = \text{a} * (\text{SIZE})^{\text{b}} \qquad (1)$$

The SIZE is measured in this equation in a thousand delivered source instructions (KLOC, this is a thousand lines of code). Both of the coefficients, "a" and "b", are productivity coefficient, as well as the scale factor coefficient, respectively. It is vital to point out here that the value of the coefficients all depends on the modes of the project. Three different modes of the project proposed by Boehm are the Organic mode, the Semi-detached mode and the embedded mode. The first is organically utilized for small-sized projects of up to 2 - 50 KLOC. The second is a semi-detached mode which is for medium-sized projects of up to 50 - 300 KLOC. Thirdly, the embedded mode is for complex, large projects that are typically over 300 KLOC.

The intermediate model is utilized to compute the effort as a function of the program's size and the set of cost drivers. This model differs slightly from the Basic one as the Basic COCOMO fails to take into account the software's development environment, which the intermediate model does. The Intermediate COCOMO has 15 cost drivers that add a level of accuracy to the Basic COCOMO. There are four classes of these cost drivers, which are Computer attributes, Product attributes, Project attributes and Personnel attributes.

The equation that is used for estimating the software Effort for the intermediate model is [17]:

$$\text{Effort} = a\left(\text{SIZE}\right)^{b} \times m\left(X\right) \tag{2}$$

In the equation, m(X) presents the effort adjustment factor and this is the product of a total of 15 Effort Multipliers. The third one (the detailed model) has two more capabilities. These are phase-sensitive effort multipliers and 3 level product hierarchies. The 3 levels are the Module, Subsystem and system and these are used to derive an accurate estimate.

### 2.2. Approach 2: Feed-Forward Neural Network with Principal Component Analysis (PCA)

The authors in [22] propose the reduction technique, which is called the feed-forward neural network with Principal Component Analysis (PCA). The authors' main objective is to use this in order to measure the accuracy of the software cost estimation model. The proposed technique is based on both algorithmic and non-algorithmic methods. So, they used a combination of algorithmic method (COCOMO) and non-algorithmic (Artificial Neural networks) to estimate the software project's costs.

In this paragraph, the authors will briefly address the Architecture Design of their System for Software Cost Estimation they have proposed. Its main parameters, which will be used as inputs for the proposed methods, are the size, cost factors and the scale factors. These parameters are all from the Actual Dataset that has been collected, as per the Project Specification. The second step is to apply the PCA. This is done by calculating the correlation coefficient matrix, as well as the Eigen-value of correlation coefficient matrix. The amount of principal components can be determined after that. These components are fed as input into the neural network system in order to train the dataset. The output layer then sends the size, effort multiplier and the scale factor values to COCOMOII. From these inputs, which are sent from the neural network system COCOMOII, the software cost can be estimated. This result was based on the COCOMO sample dataset, which is widely used by researchers. It consists of over 161 historical projects collected from various countries all over the world. The results show that the Hybrid technique provides a more accurate cost estimation than those provided by the same type of algorithm when the PCA and neural network are not applied.

## 2.3. Approach 3: Putnam Model/SLIM

The Putnam/SLIM estimating method was developed in the late 1970s by Larry Putnam of Quantitative Software Management, as is highlighted in the references [23] [24] [25] [26] [27]. SLIM (Software Lifecycle Management) is the name Putnam gave to the proprietary suite of tools that his company QSM, Inc. had developed which were based on his model. This is an empirical software effort estimation model and it is also one of the very first algorithmic cost models. It describes both the time and the effort that is required to finish the software project of a specified size. Based on the Norden/Rayleigh function, it is generally known as the macro estimation model (as it is primarily used for large projects). The Putnam model's software equation is given as [23] [24] [25]:

$$\frac{B^{\frac{1}{3}} * \text{Size}}{\text{Productivity}} = \text{Effort}^{\frac{1}{3}} * \text{Time}^{\frac{4}{3}} \tag{3}$$

The software equation in practical use is solved for effort when making an estimate for a software task [23] [24] [25]:

$$\text{Effort} = \left[\frac{\text{Size}}{\left(\text{Productivity} * \text{Time}\right)^{\left(\frac{4}{3}\right)}}\right]^3 * \text{B} \tag{4}$$

The estimated size of the software when the project is completed and the productivity of the organisational process is used. The Time-Effort Curve is calculated by plotting the effort as a function of time and the estimated total effort that it takes to complete the project is represented by the points along the curve [27].

This method of estimating is quite sensitive to uncertainty in the size and productivity process. Putnam advocates getting this process productivity through calibration [23] [24] [25]:

$$\text{Process Productivity} = \left[\frac{\text{Size}}{\left[\frac{\text{Effort}}{B}\right]^{1/3} * \text{Time}^{\left(\frac{4}{3}\right)}}\right] \tag{5}$$

**SLIM's Advantages**

- It utilizes linear programming in order to consider the development constraints of both the cost and effort required.
- One of the Putnam model's distinguishing features is that the total effort decreases as the time taken to finish the project extends. This is usually represented by a schedule relaxation parameter in other parametric models.
- SLIM needs fewer parameters in order to generate an estimate over both COCOMO'81 and COCOMO'II.

**SLIM's Drawbacks**

- This model is based on either knowing or being able to accurately estimate the size of the software (in the lines of code) to be developed. There is frequently a lot of uncertainty about the size of the software, which can result in the cost estimation being inaccurate. SLIM's error percentage is 772.87% [24], according to Kemerer's research,

- This model is extremely sensitive to development time, as decreasing this can greatly increase the number of people and months that are required for development.
- It is not suitable for small projects.

## 2.4. Approach 4: Function Point Analysis

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

Algorithmic models, such as COCOMO, Putnam, etc., need the number of SLOC (source line of codes) to be estimated so as to get both the man-months and the duration estimates. Function Point Analysis is another method that can be used to quantify both a software system's size and complexity, in terms of which functions it is able to deliver to the user. Allan Albrecht at IBM developed the Function Points Measurement method, which was first published in 1976 [28] [29]. Two steps are involved in counting the Function Points [24], which are:

- Counting the various user functions
- Making adjustments for processing the complexity

Currently, the five user function categories are: external output types, external input types, external interface file types, logical internal file types and external inquiry types. It was recognized by Albrecht that the effort that is needed to provide a given level of functionality could vary, and this depended on environmental factors. For example, it is harder to input transactions to a program if much emphasis has been placed on either the system throughput or on end-user convenience. Therefore, Albrecht listed 14 processing complexity characteristics in response to this. These are to be rated on a scale that goes from 0 (which signifies no influence) up to 5 (meaning a strong influence). All the processing complexity points that have assigned are then summed up in the next step. This number is multiplied by 0.01. It is then added to 0.65 in order to obtain the following weighting:

$$PCA = 0.65 + 0.01 * \left( \sum_{i=1}^{14} ci \right) \tag{6}$$

where $PCA$ = processing complexity adjustment and then $ci$ = complexity factors.

As a result, the various Function Points can vary ± 35 percent from the original Function Counts. Once they have been computed, these Function Points can be used in order to compare the size of the project that is proposed, compared with previous projects.

There are a number of advantages of a function point analysis based model, and these are [30] [31] [32]:

- The function points can be estimated from either the requirements specifications or the design specifications, which makes it possible to estimate the development costs in the development's early phases.
- These function points are independent of the language, tools or methodologies that have been used for implementation.
- Non-technical users are able to obtain a better understanding of what the function points are measuring, as the function points have been based on the system user's own external view of the system.

## 2.5. Approach 5: Wavelet Neural Network (WNN)

The authors employed Wavelet Neural Network (WNN) in [33] to make an estimation of the software development effort. Experiments were made on two variants of WNN, the Morlet wavelet function [34], the Gaussian function, and a threshold acceptance training algorithm for wavelet neural network (TAWNN) that had been proposed. The results of these were compared with other computational intelligent methods, such as the Multilayer Perceptron (MLP) [35], the Radial Basis Function Network (RBFN) [36], the Multiple Linear Regression (MLR) [37], the Dynamic Evolving Neuro-Fuzzy Inference System (DENFIS) [38] and the Support Vector Machine (SVM) [39]. All of these comparisons were computed in terms of the Mean Magnitude Relative Error "MMRE" applied on both the Canadian Financial "CF" dataset and the IBM Data Processing Services "IBMDPS" dataset.

The WNN that was used in this study is made up of a total of three interconnected layers. These are the input layer, the hidden layer and an output layer that has a single unit, as shown in **Figure 1**. There are also two variants of WNN, which are the Morlet function and also the Gaussian function. These were both applied as an activation function. It is crucial to point out that the authors are influenced by the Threshold Acceptance (TA) concept where a new solution's

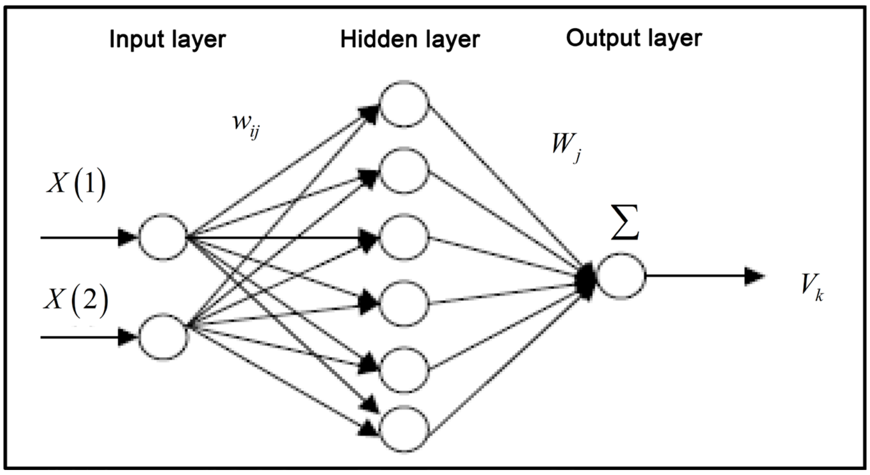

**Figure 1.** Wavelet Neural Network WNN.

acceptance is determined by deterministic criteria, instead of a probabilistic approach.

The idea is as follows: that the forward part of the back propagation remains undisturbed, while the back propagating of TA updates is done by making all of the weights a vector of decision variables. This TA concept was adopted by the authors in order to train the WNN, which is why it is called the Threshold Acceptance Wavelet Neural Network, or the TAWNN learning algorithm. This training algorithm's objective function is given as:

$$MSE = \sum_{K=1}^{np} \left(Vk - Va\right)^2 \tag{7}$$

The study's results demonstrate that the 4-models of WNN that are used in these experiments successfully produce better results compared to the other techniques. The mean magnitude of the relative error (MMRE) of both WNN-Morlet and WNN-Gaussian are successful, compared to both TAWNN-Morlet and TAWNN-Gaussian for the IBMDS and CF datasets.

## 3. Evaluation Characteristics

After describing the previous section's software cost estimation approaches, we will list our evaluation characteristics that we are going to use to compare them (see Table 1).

### 3.1. Ease of Use

This implies how simple it is to use and how easy it is to utilise a certain technique or approach. One fact that needs to be understood here is that the effort

Table 1. Evaluation criteria.

| Characteristic | Brief Description |
|---|---|
| Ease of Use | The approach used should be simple enough to be implemented in a reasonable time frame. |
| Adaptability | A method or a model should be adaptive to the changes otherwise the model will have limited usability. |
| Accuracy | Better accuracy implies better reliability. |
| Consistency | Consistency in results should be an important feature for any estimation model. |
| Interpretable | The results of the modeling technique have to be interpretable. |
| Automatable | It is desired to have a technique that could be substantially automated. |
| Tool supported "Supportability" | It is better if the proposed approach has been supported by an existing tool. |
| Empirical Validations | The empirical validation of a model adds to its credibility. |
| Sensitivity | In effort/time Estimation, it is desirable to have low sensitivity. |
| Handling Imprecision and Uncertainty | A model which considers imprecision is better than a model which doesn't. |

needed to estimate the cost of software development should be minimal. The approach used should preferably be simple enough to be done in a reasonable amount of time. If a software estimation approach uses a complex formula and algorithm, then the software cost estimation approach is said to have higher complexity and so might be undesirable.

### 3.2. Adaptability

The model's or method's ability to adjust to the new environment and fit the development practices' incremental style is called the adaptability of the model [38]. It is important that a method or model can be adaptive to change. If it is not, its usability will be limited. An approach might be valid for a certain kind of project (like a small one) but not be applicable to other kinds, such as large projects.

### 3.3. Accuracy

The definition of accuracy is how close a result is to the correct value [41]. The two ways to compare the result to its correct value are the difference and the ratio. In evaluating how accurate the software cost estimation models are, both the difference and ratio measures were used [42]. Improved accuracy implies more reliability. Comparing the estimation accuracy of the various approaches is difficult for various reasons, like different datasets, divergent definitions of similar terms and differing goals of estimation accuracy [43].

### 3.4. Consistency

Models that have been developed in different environments require calibration to work well. To consistently overestimate or under estimate a model is not as difficult to calibrate as an inconsistent one. As well as accuracy, consistency is an important feature for estimation models [42]. To measure the consistency level, there are some researchers who have used the correlation coefficient, SDR, between the observed and estimated values [44].

### 3.5. Interpretable

The modelling technique results all have to be interpretable. For example, if a modelling technique which produces hard-to-interpret results is identified as being the best one, it would not be a useful recommendation. This is because project managers would, in practice, be unlikely to apply a model that could not be understood. This excludes techniques like Artificial Neural Networks [40].

### 3.6. Automatable

As many techniques need intensive computation for accuracy, a technique that could be substantially automated [40] is desired.

### 3.7. Tools Supported or "Supportability"

Software cost estimation tools are able to improve accuracy by carrying out an automated calculation for the project. Tool-supported characteristics are able to point out if the proposed approach has a tool that supports it or not. If it is supported, then the major characteristic of this tool will be highlighted, such as its usability or efficiency [40] [42] [43].

### 3.8. Empirical Validations

A model's evaluation and validation or a general approach is vital. If the model can be validated, the criterion for validation and the dataset that it is validated on are considered. The industry's datasets are considered to be more reliable than the student datasets or those from open sources [43]. The model's empirical validation also adds to its credibility.

### 3.9. Sensitivity

An input's receptiveness or responsiveness to an input stimulus is called sensitivity, and in software development, we call a sensitive model one where there is a change in an estimated effort with respect to a small change in the input values. It is desirable to have a low sensitivity in effort/time estimation.

### 3.10. Handling Both Imprecision and Uncertainty

It is common for all software development practices to take into account both the imprecisions and uncertainty that is associated with the processes. There is reasonable imprecision when estimating the software's size and much uncertainty in predicting the various factors that are associated with developing the software [43]. A model that considers the factors is better than a model that does not.

## 4. Comparing the Approaches

### 4.1. Ease of Use

Ease of Use is an important criterion for evaluating the various approaches. This determines the degree of simplicity of a given approach, as it will try and answer how easy it is to utilise this approach. It is easy to use the COCOMO Model on small projects. However, it can be difficult to utilise it in large projects due to how complex these projects are and how many unknown variables there are that exist in these situation. Thus, this approach's score in this characteristic is 12 out of 15. For the hybrid approach (both algorithmic and non-algorithmic methods) proposed by the authors of [22], it not easy to use such an approach. This is because it is a hybrid approach which uses both an algorithmic method (COCOMO) and anon-algorithmic (Artificial Neural networks) in order to estimate the software project's cost. The score will be 8 out of a total of 15; as such an approach clearly requires a further step after using the COCOMO Model (which is the Artificial Neural networks algorithm). One of the earliest types of models developed is the Putnam model. It is amongst the ones most widely used and so is closely related to models like COCOMO. One of the model's key advantages is the simplicity it is calibrated with. Most software organisations, regardless of maturity level, can easily collect the size, effort and duration (time) of past projects. It is, therefore, given 12 out of 15. A single unit's cost (in dollars or hours) from past projects is calculated by the Function Point. This method enables abstraction from a specific language, methodology or technology to take place and it is easier to understand and also to interpret for non-technical and external stakeholders, as well as users. The score for function point analysis is, therefore, 12 out of 15. Wavelet Neural network (WNN) has 5 out of 15 as it is very hard to use and contains many parameters that require a specialist person in order to calculate them. This score is clearly a result of the model's complexity. Its two main functions are the Morlet wavelet function and the Gaussian function that are used to perform this calculation. These functions require more sophisticated people to deal with them.

## 4.2. Adaptability

Adaptability is another important characteristic. This gives a degree of adaptability if the given approach can be adjusted, according to new changes and environments. The COCOMO Model is adaptive to the changes, particularly for little projects. In addition, we can just re-compute the values in the case of changes as a result of its mathematical model. It is also important to take the project's size into account as this can affect the COCOMO Model's adaptability. It, therefore, has a score of 8 out 10. Similarly, for [22], the model is given 8 out of 10 as it uses the COCOMO Model and takes into consideration the fact that Artificial Neural networks are fully adaptable. Software engineers look for a metric that should both be technology independent and support the need for estimating the project management, which measures the quality and gathering requirements. The measure that accomplishes all of these tasks is Function Point Analysis, and so was given 8 out of 10. The requirement specification was not included in the Putnam model. It is not expected that an estimation which uses SLIM will take place until the design and coding has occurred. This is why it gets a score of 6 out of 10. The WNN was given 8 out of 10 as it is similar in behaviour and performance to the Artificial Neural Networks, but the adaptation process has some complexity because of the complexity of the neuron's internal function.

## 4.3. Accuracy

As long as we are feeding this model with almost correct values in terms of its accuracy, the COCOMO Model is very accurate, as this is a mathematical model. But this model will produce the wrong result if we feed it with the wrong values for the variables. In the first case in our study, when we feed the COCOMO Model with almost the correct value for variables, the accuracy characteristic for the model is 12 out of 15. For [22], when the proposed approach uses the COCOMO Model, it proves to be accurate to a certain extent because the Artificial Neural networks are used. This might generate a wrong estimation if it is not done very well. We cannot guarantee 100% accuracy for ANN in many cases as it depends on the way that the construction phase, training phase, input value and numerous learning parameters are done. This approach, therefore, has a total of 8 out 15. The weakness of the Putnam is the same as the original COCOMO, in that SLOC has to be extrapolated in order for the software to be successfully implemented. This is a difficult thing to do, particularly at the start of the project. As a big section of the executable code is either reused or generated through standalone components or by middle ware in almost every modern system, it may be hard to get an acceptably accurate result using the method. Therefore, its score is 10 out of 15. Function Point Analysis contains some mathematical equations, as was previously mentioned in the literature review, and function points counts could either increase or decrease by 35%, so it might not be 100% accurate. Hence, the score is 12 out of 15. WNN is not 100% accurate as it depends on the kind of datasets used. Therefore, this model is given the same score as ANN as it depends on the method of the construction phase, training phase, input value, and a lot of learning parameters.

## 4.4. Consistency

An important feature for any estimation model should be the consistency of its result. We will focus during our evaluation on whether we are able to determine the consistency level or not. The COCOMO Model, Rina *et al.* [22] and WNN approaches have a 10 out of 10 score, as we can determine the consistency level by utilising the correlation coefficient between the observed and estimated values. The Putnam Model has two kinds of equations and it follows a typical Rayleigh curve. The results are expected to be consistent and, therefore, it is given a score of 10. The results of the function point analysis will be consistent if it is based on an assumption that every one of the inputs is correct, but if different steps are used, the output could differ sometimes. Thus, it is given a score of 8 out of 10.

## 4.5. Interpretable

We will assign a score for this based on the results of the approach, which can be easily interpreted. The COCOMO Model gets 10 out of 10 as it produces numbers that everyone can both understand and interpret. It is, therefore, a useful approach. It is recommended that project managers apply such a model as it means they will be able to understand the outputs. For [22] and WNN, however, a score of 5/10 is given as these approaches, using ANN and WNN, can produce understandable outputs. Even the ANN model has some parameters which need a sophisticated person to explain. These results can be easily interpreted by using the different tools for the Putnam/SLIM model and, therefore, gets a score of 10. The Function Point Analysis results are not as easy to interpret because here are various inputs and outputs to the system and so one user may interpret them differently from another, thus it is given a score of 6.

### 4.6. Automatable and Tool Supportability

When it comes to Automatable and "Supportability", we did not find any tool from the survey that is automated fully for the COCOMO Model. Therefore, we did not discover a tool that could produce the results from A-Z directly and automatically. We can utilise a tool at a specific stage to calculate a particular value or parameter.

### 4.7. Empirical Validations

Validation is another important characteristic of an approach or model. COCOMO is a validated model as this is a mathematical model, so it is therefore valid. It can be run on some datasets or an empirical study in order to validate it. It, therefore, has a score of 10/10 score for this characteristic. The approach for [22] is 5/10. This is because the approach uses ANN, which can perform better in some datasets, but poorly on another dataset. Therefore, we cannot validate it entirely, 100%. Similarly, this case will happen in WNN as they have an NN construction that is almost the same. The Putnam Model is based on mathematical equations as well. It follows the Rayleigh curve and can be easily validated, so it gets a score of 10. The Function Points rely on gathering a number of inputs and the function count may either increase or decrease by 35%, so it, therefore, gets a score of 8 out of 10.

### 4.8. Sensitivity

Sensitivity is also an extremely important characteristic. A model is sensitive when there is a big change in the estimated effort in the input values. The COCOMO Model is called a sensitive model as it is a mathematical one and if we alter a power/exponent in the function, the difference that exists between the old and new value will be a significant one. For example, the $100^1$ and $100^2$ result for the first one is 100 and for the second, the result is 10,000. This shows a very big difference for a very small change, which was from power 1 to power 2. It too is, therefore, a sensitive model. Similarly, the Rina *et al.* [22] approach has a score of 10/10 because it uses the COCOMO Model. As WNN does not utilise the COCOMO Model, it has a low sensitivity. Its score is therefore 5/10. The Putnam/SLIM Model's software equation includes a fourth power and so it has strong implications for the resource allocation of big projects. Some relatively small extensions in the delivery rate can result in a substantial reduction in the efforts. Therefore, the sensitivity is 10/10. The overall aim of the Function Point's sizing process was to determine an adjusted function point (AFP) count which represents the software system's functional size. Several steps are required to achieve this goal and they mainly involve a summary, as well as the terms' product. Therefore, the sensitivity for the Function Point gets a score of 8 out of 10.

### 4.9. Handling Imprecision and Uncertainty

This is a vital characteristic when the different software's cost estimation is compared. We will determine in our evaluation if a given model takes into consideration imprecision during the development of a software project. We can say that the COCOMO Model is a static model and has several variables that must be known before an estimation of the overall cost can be calculated. It also takes the software's size into consideration. However, the project might not correctly handle the issues of uncertainty if the model is large and complex because there will be unknown variables. Our score for both the COCOMO Model and Rina *et al.* [22] therefore approaches 2/5. The approach of Rina *et al.* [22] gets 2/5 as it uses the COCOMO Model. There was uncertainty in the Putnam/SLIM model's earlier versions as this was generated for the minimal time solution. Providing risk tables removed some of this and so it is given a score of 3/5. The uncertainty can be reduced if the various inputs to the function points are done with precision, but as this uncertainty exists, it gets a score of 3 out of 5. The score for WNN is 2/5 because it is a static model that primarily depends on the dataset types. It, therefore, cannot handle the issue of uncertainty issue during the different processes of software development. Shown in **Table 2** is a summary comparison of the five fundamental software cost estimation approaches based on our defined evaluation criteria.

## 5. Conclusions

Software cost estimation can be seen as essential activity that needs the utilization of both right methods and techniques in order to accomplish a good estimation of the results. This is why we studied several cost estimation approaches in this work and then evaluated and compared five of them. These five approaches

Table 2. Comparison between selected cost estimation approaches.

| Evaluation criteria | The five software cost estimation approaches | | | | |
|---|---|---|---|---|---|
| | CoCoMo Model | Hybrid approach | Putnam model | Function Point Analysis | Wavelet Neural network (WNN) |
| Ease of Use | 12 | 8 | 12 | 12 | 5 |
| Adaptability | 8 | 8 | 6 | 8 | 8 |
| Accuracy | 12 | 8 | 10 | 12 | 8 |
| Consistency | 10 | 10 | 10 | 8 | 10 |
| Interpretable | 10 | 5 | 10 | 6 | 5 |
| Automatable | 5 | 5 | 5 | 5 | 5 |
| Tool supported Supportability | 2 | 3 | 5 | 2 | 2 |
| Empirical Validations | 10 | 5 | 10 | 8 | 5 |
| Sensitivity | 10 | 10 | 10 | 8 | 5 |
| Handling Imprecision and Uncertainty | 2 | 2 | 3 | 3 | 2 |
| **Total** | **81** | **74** | **79** | **72** | **55** |

are Constructive Cost Model (CoCoMo), Feed-forward neural network with Principal Component Analysis (PCA), Putnam model/SLIM, Function Point Analysis and Wavelet Neural Network (WNN).

It is important to note here that we introduced different evaluation characteristics (*i.e.*, ease of use, adaptability, accuracy, consistency, interpretable, automatable, tool supported, empirical validations, sensitivity, and handling imprecision and uncertainty) in order to compare between these five software cost estimation approaches.

Our observations indicated that it is best to use a number of different estimating techniques or cost models for the project manager, and then compare the results, before determining the reasons for large variations and documenting any assumptions that were made while making the estimates.

## References

[1] Boehm, B., Abts, C. and Chulani, S. (2000) Software Development Cost Estimation Approaches—A Survey. *Annals of Software Engineering*, **10**, 177-205. https://doi.org/10.1023/A:1018991717352

[2] Ramesh, M.R. and Reddy, C.S. (2016) Difficulties in Software Cost Estimation: A Survey. *International Journal of Scientific Engineering and Technology*, **5**, 10-13.

[3] Ramacharan, S. and Rao, K.V.G. (2016) Scheduling Based Cost Estimation Model: An Effective Empirical Approach for GSD Project. *IEEE 13th International Conference on Wireless and Optical Communications Networks* (*WOCN*), Chicago, 21-23 July 2016, 1-5. https://doi.org/10.1109/WOCN.2016.7759881

[4] Marinho, M., Sampaio, S., Lima, T. and Moura, H. (2014) A Systematic Review of Uncertainties in Software Project Management. arXiv Preprint arXiv:1412.3690.

[5] Al-Qudah, S., Meridji, K. and Al-Sarayreh, K.T. (2015) A Comprehensive Survey of Software Development Cost Estimation Studies. *Proceedings of the International Conference on Intelligent Information Processing*, *Security and Advanced Communication*, ACM, 53. https://doi.org/10.1145/2816839.2816913

[6] Galorath, D.D. and Evans, M.W. (2006) Software Sizing, Estimation, and Risk Management: When Performance Is Measured Performance Improves. CRC Press. https://doi.org/10.1201/9781420013122

[7] Wu, L. (1997) The Comparison of the Software Cost Estimating Methods. University of Calgary, Calgary, 2-12.

[8] Thamarai, I. and Murugavalli, S. (2017) Competitive Advantage of Using Differential Evolution Algorithm for Software Effort Estimation. *Communication and Power Engineering*, 62.

[9] Kaushik, A., Chauhan, A., Mittal, D. and Gupta, S. (2012) COCOMO Estimates Using Neural Networks. *International Journal of Intelligent Systems and Applications*, **4**, 22. https://doi.org/10.5815/ijisa.2012.09.03

[10] Sommerville, I. (2006) Software Engineering. 8th Edition.

[11] Venkataraman, R.R. and Pinto, J.K. (2011) Cost and Value Management in Projects. John Wiley & Sons.

[12] Benediktsson, O. and Dalcher, D. (2005) Estimating Size in Incremental Software Development Projects. *IEE Proceedings-Software*, **152**, 253-259. https://doi.org/10.1049/ip-sen:20050019

[13] Boehm, B.W. (2017) Software Cost Estimation Meets Software Diversity. *Proceedings of the* 39*th International Conference on Software Engineering Companion*, 495-496. https://doi.org/10.1109/ICSE-C.2017.159

[14] Aljahdali, S. and Sheta, A.F. (2010) Software Effort Estimation by Tuning COOCMO Model Parameters using Differential Evolution. *IEEE/ACS International Conference on Computer Systems and Applications*, 1-6. https://doi.org/10.1109/AICCSA.2010.5586985

[15] Guide, P.M.B.O.K. (2004) A Guide to the Project Management Body of Knowledge. Project Management Institute, Vol. 3.

[16] William Dow, P.M.P. and Taylor, B. (2010) Project Management Communications Bible. Vol. 574, John Wiley & Sons.

[17] Boehm, B. (1981) Software Engineering Economics. Prentice Hall.

[18] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. and Selby, R. (1995) Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, **1**, 57-94. https://doi.org/10.1007/BF02249046

[19] Musilek, P., Pedrycz, W., Sun, N. and Succi, G. (2002) On the Sensitivity of COCOMO II Software Cost Estimation Model. *Proceedings* 8*th IEEE Symposium on Software Metrics*, 13-20. https://doi.org/10.1109/METRIC.2002.1011321

[20] Chen, Z., Menzies, T., Port, D. and Boehm, B. (2005) Feature Subset Selection Can Improve Software Cost Estimation Accuracy. *ACM SIGSOFT Software Engineering Notes*, **30**, 1-6. https://doi.org/10.1145/1083165.1083171

[21] Rastogi, H., Dhankhar, S. and Kakkar, M. (2014) A Survey on Software Effort Estimation Techniques. 5*th International Conference on Confluence the Next Generation Information Technology Summit*, 826-830. https://doi.org/10.1109/CONFLUENCE.2014.6949367

[22] Waghmode, R.M., Patil, L.V. and Joshi, S.D. (2013) A Collective Study of PCA and Neural Network based on COCOMO for Software Cost Estimation. *International Journal of Computer Applications*, **74**.

[23] Putnam, L.H. (1978) A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering*, **4**, 345-361. https://doi.org/10.1109/TSE.1978.231521

[24] Kemerer, C.F. (1987) An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, **30**, 416-429. https://doi.org/10.1145/22899.22906

[25] Leung, H. and Fan, Z. (2002) Software Cost Estimation. Handbook of Software Engineering, Hong Kong Polytechnic University, 1-14. https://doi.org/10.1142/9789812389701_0014

[26] Jeffery, D.R. and Basili, V.R. (1988) Validating the Tame Resource Data Model. *Proceedings of the* 10*th International Conference on Software Engineering*, 187-201. https://doi.org/10.1109/ICSE.1988.93700

[27] http://en.wikipedia.org/wiki/Putnam_model

[28] Albrecht, A.J. (1979) Measuring Application Development Productivity. *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, 83-92.

[29] Albrecht, A.J. (1984) AD/M Productivity Measurement and Estimate Validation. IBM Corporate Information Systems, IBM Corp., Purchase.

[30] http://www.computing.dcu.ie/~renaat/ca421/LWu1.html

[31] http://en.wikipedia.org/wiki/Function_point

[32] Sharma, N., Bajpai, A. and Litoriya, M.R. (2012) A Comparison of Software Cost Estimation Methods: A Survey. *The International Journal of Computer Science and Applications*, **1**.

[33] Kumar, K.V., Ravi, V., Carr, M. and Kiran, N.R. (2008) Software Development Cost Estimation using Wavelet Neural Networks. *Journal of Systems and Software*, **81**, 1853-1867.

[34] Rioul, O. and Vetterli, M. (1991) Wavelets and Signal Processing. *IEEE Signal Processing Magazine*, **8**, 14-38. https://doi.org/10.1109/79.91217

[35] http://en.wikipedia.org/wiki/Multilayer_perceptron

[36] Moody, J. and Darken, C.J. (1989) Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, **1**, 281-294. https://doi.org/10.1162/neco.1989.1.2.281

[37] http://en.wikipedia.org/wiki/Multiple_linear_regression

[38] Kasabov, N.K. and Song, Q. (2002) DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction. *IEEE Transactions on Fuzzy Systems*, **10**, 144-154. https://doi.org/10.1109/91.995117

[39] Vapnik, V.N. and Vapnik, V. (1998) Statistical Learning Theory. Vol. 1, Wiley, New York.

[40] Briand, L.C., El Emam, K., Surmann, D., Wieczorek, I. and Maxwell, K.D. (1999) An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. *Proceedings of the* 1999 *International Conference on Software Engineering*, 313-323.

[41] Deeson, E. (1991) Collins Dictionary of Information Technology. HarperCollins.

[42] Lo, B. and Gao, X. (1997) Assessing Software Cost Estimation Models: Criteria for Accuracy, Consistency and Regression. *Australasian Journal of Information Systems*, **5**, 30-44. https://doi.org/10.3127/ajis.v5i1.348

[43] Kamal, M.W., Ahmed, M.A. and El-Attar, M. (2011) Use Case-Based Effort Estimation Approaches: A Comparison Criteria. Software *Engineering and Computer Systems*, 735-754. https://doi.org/10.1007/978-3-642-22203-0_62

[44] Mukhopadhyay, T., Vicinanza, S.S. and Prietula, M.J. (1992) Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation. *MIS Quarterly*, **16**, 155-171. https://doi.org/10.2307/249573

## Appendix: Case Study—Implementation of Putnam Model

As one of the first algorithmic cost models, the Putnam estimating method [24] [25] [26] [27] is an empirical software cost estimation model that describes both the time and the effort required to complete a software project of a specific size. The model was based on the Norden/Rayleigh function and it is generally known as a macro estimation model, as it is generally used for big projects. The model's software equation is given by:

$$\text{Effort} = \left[\frac{\text{Size}}{(\text{Productivity} * \text{Time})^{\left(\frac{4}{3}\right)}}\right]^3 * \text{B} \tag{8}$$

By plotting effort as a function of time, the Time-Effort Curve is established. The various points along this curve represent the estimated total effort that is made in order to complete the project. A distinguishing feature of the model is that the total effort decreases as the time that is taken to complete the project gets extended. Other parametric models usually represent this with a schedule relaxation parameter.

The model is used as a case study in order to see the effect of a factor on the Time-Effort curve. The size in the first case was assumed to be 50,000 source lines of code, while factor B was 0.000005. The curves were plotted by utilising the productivity values of 15, 13 and 11. As productivity decreases, the time that is taken to finish the project increases.

In another case, which involved the same source lines of codes, as well as the same three productivity factors, factor B was increased. This was done in order to see the effect of it on the curve. The three sets of curves (from left to right) are for the values of B, as 0.000005, 0.00005, and 0.0005 respectively. As factor B increases, the time taken to finish the project also increases.

As can be seen from the two figures above (Figure 2 and Figure 3), the Put-

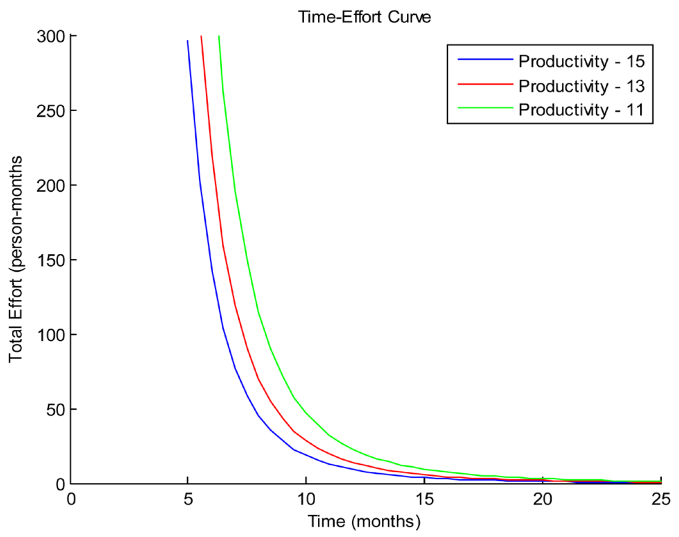

Figure 2. Time-Effort curve (Size = 50,000 source lines of codes, B = 0.000005, productivity = 15, 13, and 11).

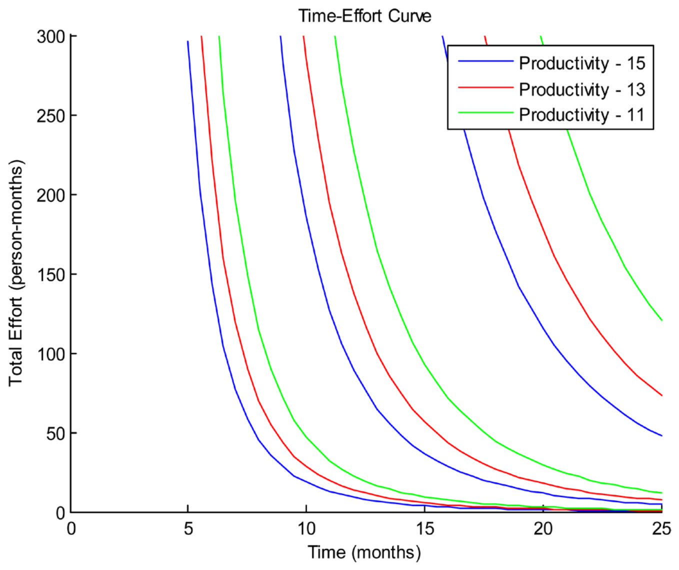

**Figure 3.** Time-Effort curve (Size = 50,000 source lines of codes, B = 0.000005, 0.00005, and 0.0005, and productivity = 15, 13, and 11).

nam model is based on the knowledge of—or being able to accurately estimate—the size (in the lines of code) of the software that is to be developed. There is frequently much uncertainty about the software's size, which can result in the cost estimation being inaccurate. In addition, this model is unsuitable for projects that are very small.