Software Effort and Function Points Estimation Models Based Radial Basis Function and Feedforward Artificial Neural Networks

ALAA SHETA
Computers and Systems Department
Electronics Research Institute (ERI)
DAVID RINE
Computer Science Department
George Mason University
and
SOFIAN KASSAYMEH
Information Technology Department
Taif University

Correct cost estimates is an essential element to develop economic proposals and being competitive in the market. Companies have to estimate the effort, time and cost before bidding for a project. An inaccurate estimate will lead to money and market share loss. To do that the expected software development size has to be estimated early in the development process. Many software models were presented in the literature to handle this task such as expert judgment, analogy-based estimation, formal estimation models and combination-based estimation models. These models were found to be risky and created many problems in practice related to availability of expertise and inaccurate estimate. Soft Computing techniques were successfully used to solve a diversity of problems in software engineering project cost estimation management. Earlier investigation proved that techniques such as Artificial Neural Networks (ANNs) can solve many problems in the field of software engineering project cost estimation management with promising results. In this paper, we propose two new models for software effort and function point estimation using ANNs. Two types of ANNs will be explored; the Feedforward (FF) and the Radial Basic Function (RBF) neural networks. The Albrecht data set with a number of attributes was used to provide our results. Developed results shows that ANNs models can provide an accurate estimate for both the software effort and number of function points.

General Terms: Algorithms, Management, Design, Experimentation, Theory.

Keywords: Software effort estimation, function points, COCOMO, artificial neural networks

1. INTRODUCTION

Many governmental agencies, departments and private companies around the world spend billions of dollars yearly on software development and maintenance. Many projects failed to accomplish their requirements due to failure of existing software cost-estimation techniques in providing accurate estimate before the project initiation. This is a serious financial problem for these agencies. For example government departments and the US department of defense (DoD) spent approximately 30 billion dollars every year in developing and maintenance of software [Boehm 1987]. Also a study by the US government on IT projects revealed the extent of that challenge and stated that: 60% of projects were behind schedule, 50% of projects were over cost, and 45% of delivered projects were unusable [Garmus and Herron 2002].

Software effort estimation is defined as the process of estimating the most accurate volume of effort expressed in man-months that is required to develop or maintain software. This develop-

Author's address: A.Sheta, Electronics Research Institute, El-Tahrir Street, Dokky, Giza, Egypt
D. Rine, George Mason University, Fairfax, VA, 22030, USA
S. Kassaymeh, Taif University, Taif, Saudi Arabia.

ment in many cases depended on inaccurate or incomplete data. One reason is that each project has a unique characteristic to estimate and its probability of success in many cases depend on the team experience and the type of projects. Since late 1960s many researchers have been exploring the problems of software effort estimation and presenting variety of models that can solve this problem as given in [Nanus and Farr 1964].

Software effort estimation models are not only useful in computing effort or budgeting but also help in avoiding risks, project planning, controlling resources and improving the company stock investment. In the past, software engineers gave more attention to developing formal model structure to estimate effort. Before the 70s software used to be estimated based on rules of thumb or trial and error. In 1970 with the growing size of software systems, a number of tools of automated software were presented. Barry Boehm [Boehm 1981], presented one of the well-known models in the literature, the Constructive Cost Model (COCOMO). This model was developed using sixty-three projects' data during the 1960's and 1970's. More details about this model can be found in Boehm's book, "Software Engineering Economics" [Boehm 1984]. Other popular methods for estimation in software engineering include: Analysis Effort method, Evidence-based Scheduling [Harchol-Balter et al. 2003], Function Point Analysis [Rask et al. 1992], Parametric Estimating [Zeng and Rine 2004], PRICE Systems [Stewart 1991], Putnam [Putnam and Myers 2003], SEER-SEM [Fischman et al. 2005] and many others. In [Kemerer 1987], Kemerer develop a famous study reporting the results of the comparative accuracy for four software cost estimation models. They are the Function Points [Albrecht 1979], SLIM [Putnam 1978], COCOMO [Boehm 1981], and ESTIMACS.

Another trend of software estimation was presented in 1975 which is called Function Point Analysis (FPA) for estimating the size and development effort. FPA provides a uniform method to measure the functionality of a software system from the user point of view. The answer to the question: Why we should use function points? Can be found in in [Furey 1997]. Authors claim that FPA helps developers and users measure the size and complexity of software application functions in a way that is useful to software users. FPA were used to measure and estimate real-time and embedded software system in [Lavazza and Garavaglia 2009].

We can claim that even though there many articles written are many articles since 1960 providing numerous models to help in computing the effort for software projects, being able to provide accurate effort estimation is still a challenge for many reasons. They include: 1) the cost drivers to be considered along with the development environment might not be clearly specified; 2) the uncertainty in collected measurement; 3) the estimation methods used which might have many drawbacks and

In this paper, we are exploring the use of feedforward ANN (FF-ANN) and Radial Basis Function ANN (RBF-ANN) to build two non-parametric software effort estimation models. A data set with four attributes: Kilo Line Of Code (KLOC), Methodology, Complexity and Experience were used as input for the software effort estimation model provided by Bailey and Basili in [Bailey and Basili 1981] from NASA software projects; while the FP model is utilizing the Inputs, Outputs, Files, User Inquiries as input to estimate the number of FP for software project adopted the Albrecht [Albrecht 1979; Albrecht and Gaffney 1983a]. We provide a brief description of two well-known models for cost estimation in the literature for software effort estimation: the COCOMO in Section 2 and the FP models in Section 3. Soft-Computing and its usage for effort estimation is presented in Section 4. A brief description to the adopted types of ANN: the FF and RBF models are presented in Section 5. The adopted criterion to evaluate the developed models are presented in Section 6. In Section 7, we present the experimental results based ANN. Finally, we provide a conclusion and future work for this research.

2. COCOMO MODEL

The Constructive Cost Model (COCOMO) is an algorithmic parametric software cost estimation model developed by Barry W. Boehm. COCOMO model was first presented in Boehm's 1981

book Software Engineering Economics [Boehm 1984] for estimating effort, cost, and schedule for software projects. model developed is based on 63 software projects at TRW Aerospace. TRW Inc. was an American corporation involved in a variety of businesses, mainly aerospace, automotive, and credit reporting. The model developed based on software projects with small to medium size from 2,000 to 100,000 lines of code with variety of programming languages such as assembly. The development methodology was based on the well-known waterfall model which was the dominant software development process in 1981.

Boehm presented three types of COCOMO models with various accuracy. The first type is known as Basic COCOMO. This model proves to be applicable in cases with quick and project estimation. The second type is known as the Intermediate COCOMO. This model takes in consideration number of attributes are called Cost Drivers to help improve the model estimation accuracy. Finally, the Detailed COCOMO moreover studies the effect various phases of development. Equation 1 shows the basic COCOMO model equation.

$$\begin{aligned} E &= a(KLOC)^b \quad [person - months] \\ D &= c(E)^d \quad [months] \\ P &= E/D \quad [count] \end{aligned} \tag{1}$$

E, D, P and $KLOC$ stands for software effort computed in man-months, development time, people required and Kilo Line Of Code, respectively.

The values of the parameters depend mainly on the class of software project. Software projects were classified based on the complexity of the project into three categories. They are: Organic, Semidetached and Embedded models [Benediktsson et al. 2003]. The values of the coefficients a, b, c and d are given in Table I. These models are expected to give different results according to the nature of software projects.

Table I: Basic COCOMO Models

Software project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

3. WHY FUNCTION POINT ANALYSIS?

Software size is an essential factor that help in developing an initial estimate for software effort/cost during software development life cycle. COCOMO model provided this estimate based on the KLOC. It was reported that KLOC produced many problems [DeMarco 1982; Jones 1986]. For example, in modern software programming, auto-generate tools produced large number of line of codes. KLOC also changes with the developer's experience, difference in programming languages, variation in the graphical user interface (GUI) code generation, and lack of functionality. The estimation of KLOC under this condition seems uncertain to measure.

This is why Albrecht proposed his idea of computing the software size based on the system functionality [Albrecht 1979; Albrecht and Gaffney 1983a]. He found that FPA:

(1) is an adequate tool to determine the size of application packages by counting all the functions included in the package;
(2) can help users determine the benefit of an application package knowing the exact match between the required functionality and provided functionality by the software package and finally
(3) an informal way to estimate cost and resources required for software development and maintenance.

3.1 Albrecht's Function Points (FP)

In 1979 Albrecht [Albrecht 1979], published his article on FP methodology while he was working at IBM. Albrecht's function point gained acceptance during the 1980's and 1990's because of the tempting benefits compared to the models based on the KLOC [Rask et al. 1992; Furey 1997]. Because FP is self-governing and independent of language type, platform, it can be used to identify many productivity benefits. FP is designed to estimate the time required for a software project development, and thereby the cost of the project and maintaining existing software systems.

The proposed FP is a non-parametric method. FP concerns about the logical view more than the physical. Thus, attributes such as like coding algorithms, database structure, screen-shots are not counted. FP is computed based on the analysis of project requirements. The requirements help in identifying the number of function to be developed along with the complexity of each function. Thus, there was no need to measure the size of Line of Code but only concern about project functionality. Once the number of FP measured, the average number of function points per month specified and the labor cost per month is estimated; the total budget can be computed. Albrecht originally proposed four function types [Albrecht 1979]: files, inputs, outputs and inquiries with one set of associated weights and ten General System Characteristics (GSC).

In 1983, the work developed in [Albrecht and Gaffney 1983a], proposed the expansion of the function type, a set of three weighting values (i.e. simple, average, complex) and fourteen General System Characteristics (GSCs) were proposed as given in Table II. In Albrecht FP, there are two parts in the model, which are *Unadjusted Function Point* (UFP) and *Adjusted Function Point* (AFP).

Table II: 1983 function types and weights

Function Type	Simple	Average	Complex
External Input	3	4	6
External Output	4	5	7
Internal Files	7	10	15
External Files	5	7	10
External Inquiry	3	4	6

(1) The UFP consists of five components. They are:
   (a) External Inputs (EI),
   (b) External Outputs (EO),
   (c) External Inquires (EQ),
   (d) Internal Logical Files (ILF) and
   (e) External Interface Files (EIF).

(2) There are 14 GSCs factors which affect the size of the project effort, and each is ranked from "0" means no effect to "5" means essential. GSCs consists of 14 factors known as $f_1, f_2, \ldots, f_{14}$. These factors are listed in listed in Table III. The sum of all factors is then multiplied given in Equation 2 which constitute the Adjustment Factor (AF) defined in the range [0.65, -1.35].

$$AF = 0.65 + 0.01 \sum_{i=1}^{14} f_i \tag{2}$$

(3) Then, the Unadjusted FP is then multiplied by the AF to create the Adjusted Function Point (AFP) count as given in Equation 3. The Adjusted FP value has values within 35% of the original UFP figure.

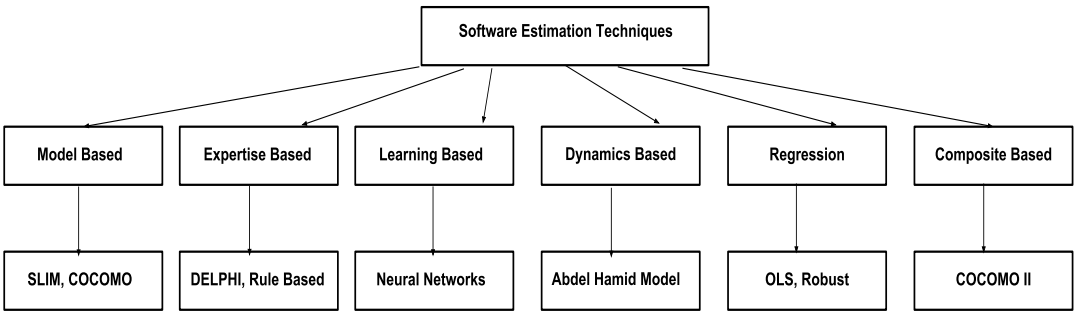
$$Adjusted\ FP = Unadjusted\ FP \times AF \tag{3}$$

Table III: General System Characteristics (GSCs)

1	Data Communications
2	Distributed Functions
3	Performance
4	Heavily Used Configuration
5	Transaction Rate
6	Online Data Entry
7	End User Efficiency
8	Online Update
9	Complex Processing
10	Reusability
11	Installation Ease
12	Operational Ease
13	Multiple Sites
14	Facilitate Change

4. SOFT-COMPUTING FOR EFFORT ESTIMATION

In the past, Soft Computing techniques were explored to build both parametric and non-parametric effort estimation models structures [Ryder 1995; Hodgkinson and Garratt 1999]. In [Kelly 1993], the author explored the use of Artificial Neural Networks (ANN), Genetic Algorithms (GAs) and Genetic Programming (GP) to provide a methodology for software cost estimation. Later authors in [Dolado and andez 1998], provided a detailed study on using Genetic Programming (GP), Neural Network (ANNs) and Linear Regression (LR) in solving the software project estimation. Many data sets provided in [Albrecht and Gaffney 1983b; Matson et al. 1994] were explored with promising results. In [Shepper and Schofield 1997], authors provided a survey on the cost estimation models using artificial neural networks. ANN were used for software engineering project management in [Kumar et al. 1994].



Examples of Software Estimation Techniques

Recently, many questions were introduced about the applicability of using Soft Computing and Machine Learning Techniques to solve the software effort and cost estimation problems. For example, can machine learning techniques used to tune the parameters of the COCOMO model? or is it possible to estimate the number of function points using Genetic Algorithms?. In [Sheta 2006b], the author used Genetic Algorithms to tune the parameters of the COCOMO model with number of variation in the model structure. Later on, many authors explored the same idea with some modification [Mittal and Bhatia 2007a; 2007b; Uysal 2008; Sandhu et al. 2009] and provided a comparison to the work presented in [Sheta 2006b]. Exploration of the advantages of

the Takagi-Sugeno (TS) fuzzy logic technique on building a set of linear models over the domain of possible software were investigated in [Sheta 2006a].

Authors in [Sheta et al. 2008] presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to estimation software effort for NASA software projects. On doing this, Particle Swarm Optimization (PSO) was used to tune the parameters of the COCOMO model. The performance of the developed models were evaluated using NASA software projects data set [Bailey and Basili 1981]. Also, a comparison between COCOMO-PSO, Artificial Neural Networks (ANNs), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided with excellent modeling results. Some of these models are parametric and some are non-parametric.

A research studies presented two new models for software effort estimation using both Multigene Symbolic Regression Genetic Programming (GP) [Aljahdali and Sheta 2013] and Fuzzy Logic [Sheta and Aljahdali 2013] were presented. One model utilizes the Source Line Of Code (SLOC) as input variable to estimate the Effort (E); while the second model utilize the Inputs, Outputs, Files, and User Inquiries to estimate the Function Point (FP). The proposed models show better estimation capabilities compared to other reported models in the literature. The validation results are accepted based Albrecht data set. The same data set is adopted in this study. In Figure 1, we provide a general hierarchy of number of estimation techniques as presented in [Suri and Ranjan 2012].

5. ARTIFICIAL NEURAL NETWORKS

A neural network is a massively parallel distributed processor non-parametric model. ANN has the ability to recognize functions or pattern as long as it was trained with enough knowledge. There are many similarities between ANN and the brain. They include:

(1) A learning process/algorithm is required.

(2) Synaptic weights are used to store the knowledge.

(3) The network is clever such that it can generalize.

ANN adopts a learning algorithm which is capable of updating the network weights such that the objective criterion is accomplished. The basic building block of the neural network system is the neuron which sends/receives information to/from various parts of the network architecture. Each neuron collects inputs from a single or multiple sources and produces a single output in accordance with a certain predetermined non-linear function.

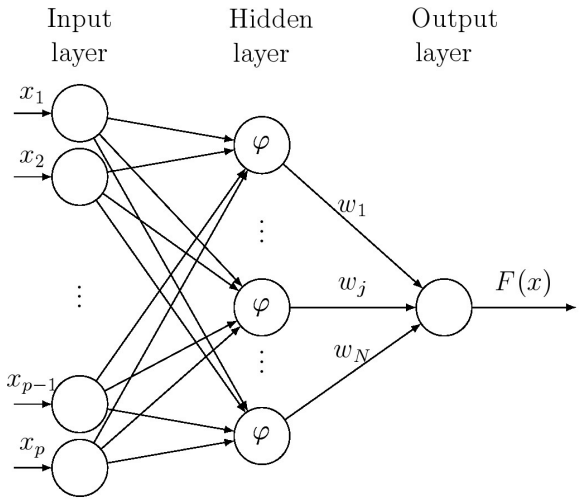
ANN is created by interconnecting many of these simple neuron models in a known configuration. The network weights are modified so as to minimize the difference between the network output and the desired output. After finishing the learning process, the network will be capable of generalization. It also can provide outputs based on specific inputs from the same domain of the problem.

5.1 Multi-Layer Perceptron

Multi-Layer perceptron (MLP) is one of the most popular types of ANN for supervised learning. It became dominant in 1986 with the development of the Backpropagation (BP) algorithm [Rumelhart and McCleland 1986]. The objective of the training process for MLP is to find the best set of weights which minimize the error difference between the system response and produced response by ANN. The MLP weights are updated by using gradient descent technique in the weight-space. Gradient descent can be viewed as a generalization of the popular least-mean-square (LMS) algorithm.

A MLP network consists of three layers: input, hidden and output layers. In some applications we could have more than one hidden layer. The number of neuron in the hidden layers depends on the complexity of the system function to be modeled. The input signal broadcasts over the network in a forward direction, layer by layer. The error backpropagation learning algorithm

consists of two phases. The first is usually referred to as the presentation phase (i.e. forward pass) and the second as the backpropagation phase (i.e. backward pass). In the presentation phase an input vector ϕ is presented to the network resulting in an output y at the output layer were the synaptic weights are all fixed. In the backpropagation phase the weights are adjusted based on the error between the actual and desired outputs. In Figure 2 we show a two layer MLP as presented in [Al-Hiary et al. 2008; Braik et al. 2008].



A Fully Connected Two Layer Feedforward Network

MLP is a fully connected network because all inputs/units in one layer are connected to all units in the following layer, the first layer is known as the hidden layer h and the second layer is the output layer. The depicted network consists of three inputs, two hidden units and two outputs. The MLP can be represented mathematically as given in Equation 4 [Norgaard et al. 2000; Al-Hiary et al. 2008].

$$\begin{aligned} \hat{y}_i &= F[\varphi, \theta, x] \\ &= \sum_{k=1}^{N} w_k \times \varphi_k \\ \varphi_k &= \sum_{l=1}^{p} f(W_l \times x_l) \end{aligned} \tag{4}$$

where W_l are the weights for the nodes at the input layer, $\hat{y}_i$ is the output signal, g_i is the function realized by the neural network and θ specifies the parameter vector, which contains all the adjustable parameters of the network (weights w_k, W_l and biases at each layer). $f(\dots)$ is the sigmoidal function used in the ANN. This function are more likely to be tan sigmoid at the hidden layer and linear at the output layer. MLP is trained by the backpropagation learning algorithm to adjust the network weights such that the objective criteria is achieved (i.e. the network output $\hat{y}$ matches the desired input y). Typically, to achieve this match many input/output pairs are used to train the network [Rebecca 1997].

5.2 Radial Basis Function Networks (RBF)

Radial Basis Function (RBF) Networks has lots of similarity such as FF neural network. They have been used successfully to solve variety of function approximation problems. They consists of three layer with only one hidden layer. The main difference is that their sigmoid function are in Gaussian format. RBF has many features makes it unique such as fast learning very good interpolation.

To see how RBF network works, let us consider a set of N data points in the input space R^d, together with their associated desired output values in R:

$$D = \{(x_i, y_i) \in R^d \times R, 1 \leq i \leq N \mid f(x_i) = y_i\} \tag{5}$$

Assuming we are considering only one dimensional output function. RBF network can be used to approximate a function f uses m functions ϕ. ϕ is the radial basis function defined as follows:

$$\phi_j(u) = \phi_j(|| \ x - c_j \ ||) \tag{6}$$

The c_j are the locations of the centroids i.e., the centers of the RBF, while || .. || denotes as the norm. x is the network input vector. The approximation of the function f may be expressed as a linear combination of the RBFs as $\hat{f}(x)$ given in Equation 7.

$$\hat{f}(x) = \sum_{j=1}^{M} w_j \phi_j(|| \ x - c_j \ ||) \tag{7}$$

The most common radial basis function, in practice, is a Gaussian kernel given by:

$$\phi_j(|| \ x - c_j \ ||) = e^{-(\frac{(||x - c_j||)}{r_j})^2} \tag{8}$$

where r_j is the width factor of the kernel j.

Once the general shape of the ϕ_j function is chosen, the purpose of the RBF algorithm is to find the parameters c_j, r_j and w_j to best fit the function f. By fitting, we mean that the global Mean-Square Error (MSE) between the desired outputs y_i for all data input points $x_i, 1 \leqslant i \leqslant N$ and the estimated outputs $\hat{y}(x)$ is minimized. The MSE is given in Equation 9.

$$\begin{aligned} MSE &= \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \end{aligned} \tag{9}$$

6. MODEL EVALUATION

The performance of the developed two models; the KLOC and the FP models based on ANNs shall be evaluated using number of evaluation criteria. They are:

(1) The Variance-Accounted-For (VAF):

$$VAF = [1 - \frac{var(y - \hat{y})}{var(y)}] \times 100\% \tag{10}$$

(2) The Mean Magnitude of Relative Error (MMRE), defined as:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i} \tag{11}$$

where y and $\hat{y}$ are the observed effort from previous projects and the estimated effort based on the proposed models and n is the number of measurements used in the experiments, respectively.

7. EXPERIMENTAL RESULTS

To develop our effort estimation and FP estimation models, we used sets of data provided by [Bailey and Basili 1981] and [Albrecht 1979; Albrecht and Gaffney 1983a].

(1) **Effort Estimation Data Set 1**: A data set was provided by Bailey and Basili in [Bailey and Basili 1981] from NASA software projects is used to develop ANN model for software effort estimation. The data set consists of four attributes: KLOC, Methodology, Complexity, Experience and Effort for 18 software projects. In this second case, we are considering KLOC, Methodology, Complexity, Experience as inputs for the ANN model (See Equation 12). The data set was sorted for experimental use.

$$E = ANN(KLOC, Methodology, Complexity, Experience) \tag{12}$$

(2) **FP Estimation Data Set 2**: In the FP modeling process, we adopted the Albrecht data set [Albrecht 1979; Albrecht and Gaffney 1983a]. In this case, our goal is to build an ANN model that relates the main inputs: Inputs, Outputs, Files and Inquiries to the FP as output (See Equation 13).

$$E = ANN(Inputs, Outputs, Files, Inquiries) \tag{13}$$

7.1 Developed ANN Model for Effort Computation

We developed a neural network model to estimate the software effort taking into consideration four attributes. They are the KLOC, Methodology, Complexity and Experience. We ran the ANN MATLAB Toolbox to develop our results. The data was split to 50% for training and the other 50% for testing the model. We used the BackPropoagation learning algorithm to train the ANN. This algorithm involves two stages: Forward propagation and backward propagation. In Forward propagation, a training pattern's input is propagated through the network to generate the output pattern based on the initial selected weights. While in Backward propagation the difference between the input and output values of all output and hidden neurons are propagated back to adjust the ANN weights.

We proposed a FF-ANN with two layers; one hidden with tan sigmoid transfer function and one neuron in the output layer with linear sigmoid function. The hidden layer had three neurons. The maximum number of epochs was set to 300. For the RBF, we used a network with spread constant of three and number of hidden nodes equals 18. The RBF convergence after running ten epochs.

In Figure 3, we show the observed and estimated effort based both the feedforward and RBF networks. In Table I, we also show the computed values of the observed software effort in real project and the estimated effort based FF-ANN and RBF-ANN. The computed MMRE and VAF for the two developed models are shown in Table IV. It was found the FF-ANN model was able to provide a better results than the RBF model.

Table IV: Computed Criterion for the Effort Estimation Model

Criteria	Effort based FF-ANNs	Effort based RBF-ANNs
MMRE	0.026168	0.25555
VAF	99.978%	98.78%

Observed and Estimated Effort Using (a) Feedforward ANNs (b) RBF Network

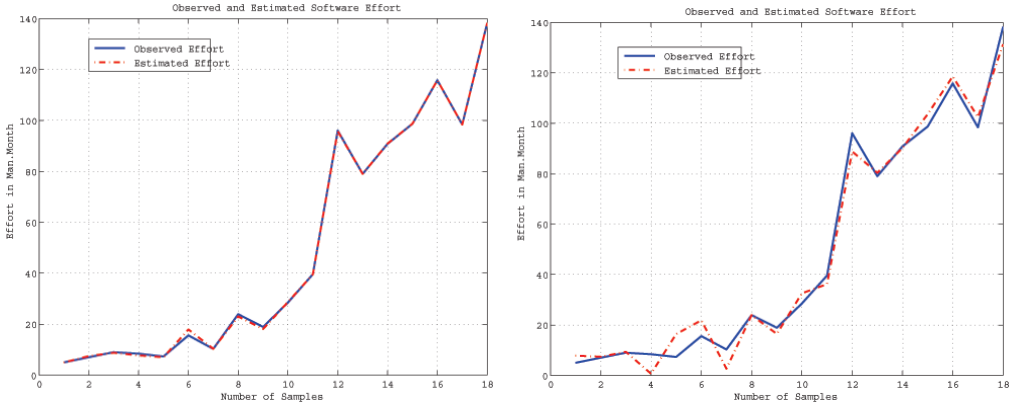


Table V: Observed and Estimated Effort Using ANN based on KLOC

KLOC	Methodology	Complexity	Experience	Effort	FF-ANN	RBF-ANN
2.1	28	19	20	5	4.9515	7.8479
3.1	26	18	6	7	7.4872	7.4044
4.2	19	23	12	9	8.7503	9.3551
5	29	21	14	8.4	7.7272	0.6594
7.8	31	18	16	7.3	6.9956	16.395
9.7	27	21	16	15.6	17.896	21.855
10.5	34	19	21	10.3	10.328	2.655
12.5	27	23	18	23.9	23.006	23.915
12.8	26	25	16	18.9	18.103	16.454
21.5	31	27	20	28.5	28.726	32.546
31.1	35	21	18	39.6	39.58	36.186
46.2	20	21	14	96	95.794	88.788
46.5	10	21	16	79	79.019	80.366
54.5	20	29	16	90.8	90.922	90.185
78.6	35	33	16	98.7	98.731	103.62
90.2	30	21	16	115.8	115.8	118.58
97.5	29	29	14	98.4	98.387	102.4
100.8	34	33	16	138.3	138.29	131.29

7.2 Developed ANN Model for FP Computation

We developed a neural network model for the effort taking into consideration the four main attributes of the FP. These attributes are: Inputs, Outputs, Files and Inquiries as presented in [Albrecht 1979]. The proposed ANNs were developed with a four-input single-output model. The ANN output is the number of function points for software projects. In this case, we also divided the data set to half-and-half; for both training and testing. A FF-ANN, with two layers, was used with three neurons in the hidden layer and one neuron in the output layer. FF-ANN convergences to the minimum error after small number of iteration.

For the RBF, we used a network with 18 hidden node, which is the total number of the data set. In Figure 4, we show the observed and estimated function points based on both the feedforward and RBF networks. The computed MMRE and VAF for the two developed models are shown in Table VI. In Table VII, we also show the computed values of the observed FP in real project and the estimated FP based FF-ANN and RBF-ANN. In this case also, it was found the FF-ANN estimation results was better results than the RBF results.

Observed and Estimated Function Points Using (a) Feedforward ANNs (b) RBF Network

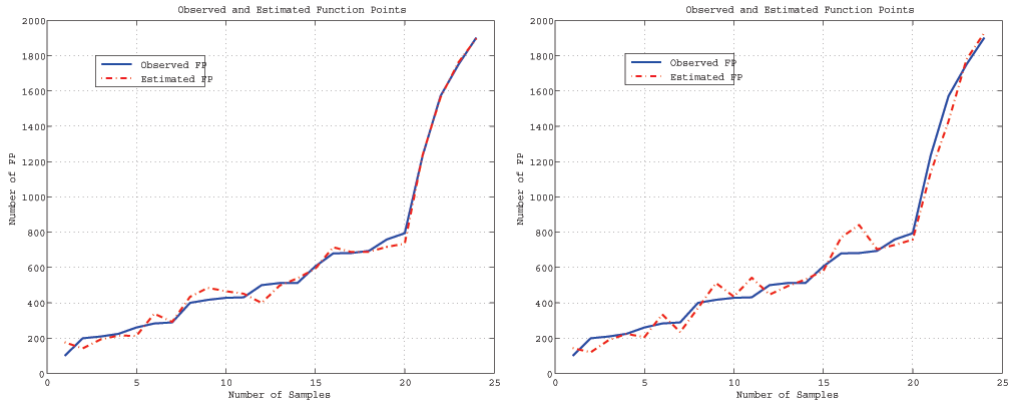


Table VI: Computed Criterion for the FP Estimation Model

Criteria	Effort based FF-ANNs	Effort based RBF-ANNs
MMRE	0.13647	0.16541
VAF	99.276%	98.011%

Table VII: Observed and Estimated Effort Using ANN based on Function Point

Inputs	Outputs	Files	Inquiries	FP	FF-ANN FP	RBF-ANN FP
34	14	5	0	100	175.73	143.19
15	15	3	6	199	141.82	119.43
7	12	8	13	209	192.45	188.28
33	17	5	8	224	214.39	223.97
12	15	15	0	260	212.71	205.2
13	19	23	0	283	338.36	334.4
17	17	5	15	289	291.72	234.17
27	20	6	24	400	433.89	370.34
28	41	11	16	417	485.13	511.87
70	27	12	0	428	465.46	434.73
10	69	9	1	431	451.5	542.33
25	28	22	4	500	399.12	446.59
41	27	5	29	512	497.09	493.97
28	38	9	24	512	538.82	532.1
42	57	5	12	606	593.28	581.12
45	64	16	14	680	715.25	770.58
43	40	35	20	682	688.06	840.83
61	68	11	0	694	689.2	703.56
40	60	12	20	759	717.02	727.95
40	60	15	20	794	734.8	758.2
48	66	50	13	1235	1239.2	1137.9
69	112	39	21	1572	1563.8	1429.1
25	150	60	75	1750	1762	1778.3
193	98	36	70	1902	1897.2	1929.9

8. CONCLUSIONS AND FUTURE WORK

Estimating the cost of development for software projects is an essential element for the success of any project manager. Many software parametric models were introduced in the literature. In this

paper, we provided two neural network models to estimate effort and number of function points for software projects. The effort model utilized the KLOC, Methodology, Complexity and Experience as input variables to estimate the Effort; while the second model utilized the Inputs, Outputs, Files, User Inquiries to estimate the Function Point of software project. Developed results showed that ANNs models can provide competitive results with high accuracy with respect to the VAF and MMRE. We plan to explore other soft computing techniques to handle the effort estimation problem.

REFERENCES

Al-Hiary, H., Sheta, A., and Ayesh, A. 2008. Identification of a chemical process reactor using soft computing techniques. In *Proceedings of the 2008 International Conference on Fuzzy Systems (FUZZ2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI2008), Hong Kong, 1-6 June*. 845–653.

Albrecht, A. J. 1979. Measuring application development productivity. In *Proceedings of the Joint SHARE, GUIDE, and IBM Application Developments Symposium*. 83–92.

Albrecht, A. J. and Gaffney, J. E. 1983a. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering 9,* 6, 639–648.

Albrecht, A. J. and Gaffney, J. R. 1983b. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Trans. Software Engineering 9,* 6, 630–648.

Aljahdali, S. and Sheta, A. 2013. Evolving software effort estimation models using multigene symbolic regression genetic programming. *2,* 12, 52–57.

Bailey, J. W. and Basili, V. R. 1981. A meta model for software development resource expenditure. In *Proceedings of the International Conference on Software Engineering*. 107–115.

Benediktsson, O., Dalcher, D., Reed, K., and Woodman, M. 2003. COCOMO based effort estimation for iterative and incremental software development. *Software Quality Journal 11*, 265–281.

Boehm, B. 1981. *Software Engineering Economics*. Englewood Cliffs, NJ, Prentice-Hall.

Boehm, B. W. 1984. Software engineering economics. *IEEE Transactions on Software Engineering 1*, 4–21.

Boehm, B. W. 1987. Survey and tutorial series-improving software productivity. *Computer*, 43–57.

Braik, M., Sheta, A., and Arieqat, A. 2008. A comparison between GAs and PSO in training ANN to model the TE chemical process reactor. In *Proceedings of the AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications, Aberdeen, Scotland*. Vol. 11. 24–30.

DeMarco, T. 1982. *Controlling Software Projects*. Yourdon Press/Prentice Hall, Englewood Cliffs, New York, USA.

Dolado, J. J. and andez, L. F. 1998. Genetic programming, neural network and linear regression in software project estimation. In *Proceedings of the INSPIRE III, Process Improvement through training and education*. British Company Society, 157–171.

Fischman, L., McRitchie, K., and Galorath, D. D. 2005. Inside SEER-SEM. *CROSSTALK The Journal of Defense Software Engineering*, 26–28.

Furey, S. 1997. Why we should use function points [software metrics]. *IEEE Software 14,* 2 (Mar.), 28–30.

Garmus, D. and Herron, D. 2002. Estimating software earlier and more accurately. *The Journal of Defense Software Engineering*, 18–21.

Harchol-Balter, M., Schroeder, B., Bansal, N., and Agrawal, M. 2003. Size-based scheduling to improve web performance. *ACM Trans. Comput. Syst. 21,* 2 (May), 207–233.

Hodgkinson, A. C. and Garratt, P. W. 1999. A neuro-fuzzy cost estimator. In *Proceedings of the Third Conference on Software Engineering and Applications*. 401–406.

Jones, C. 1986. *Programming Productivity*. McGraw-Hill, New York, USA.

Kelly, M. A. 1993. A methodology for software cost estimation using machine learning techniques. M.S. thesis, Naval Postgraduate School, Monterey, California.

Kemerer, C. F. 1987. An empirical validation of software cost estimation models. *Commun. ACM 30,* 5 (May), 416–429.

Kumar, S., Krishna, B. A., and Satsangi, P. 1994. Fuzzy systems and neural networks in software engineering project management. *Journal of Applied Intelligence 4*, 31–52.

Lavazza, L. and Garavaglia, C. 2009. Using function points to measure and estimate real-time and embedded software: Experiences and guidelines. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. ESEM '09. IEEE Computer Society, Washington, DC, USA, 100–110.

Matson, J. E., Barret, B. E., and Mellinchamp, J. M. 1994. Software development cost estimation using function points. *IEEE Trans. Software Engineering 20,* 4, 275–287.

Mittal, H. and Bhatia, P. 2007a. A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers. *International Journal of Computer Science and Security 1,* 4, 36–47.

Mittal, H. and Bhatia, P. 2007b. Optimization criteria for effort estimation using fuzzy technique. *Clei Electronic Journal 10,* 1, 1–11.

Nanus, B. and Farr, L. 1964. Some cost contributors to large-scale programs. In *Proceedings of the April 21-23, 1964, Spring Joint Computer Conference*. AFIPS '64 (Spring). ACM, New York, NY, USA, 239–248.

Norgaard, M., Ravn, O., Poulsen, and Hansen, L. K. 2000. *Neural Networks for Modelling and Control of Dynamic Systems*. Springer, London.

Putnam, L. 1978. A general empirical solution to the macro software sizing and estimation problem. *IEEE Transaction Software Engineering 4,* 4, 345–381.

Putnam, L. H. and Myers, W. 2003. *Five core metrics : the intelligence behind successful software management*. Dorset House Publishing.

Rask, R., Laamanen, P., and Lyytinen, K. 1992. A comparison of albrecht's function point and symons' mark ii metrics. In *Proceedings of the thirteenth international conference on Information systems*. ICIS '92. University of Minnesota, Minneapolis, MN, USA, 207–221.

Rebecca, C. W. 1997. Neural network models: Foundations and applications to an audit decision problem. *75*, 291–301.

Rumelhart, D. E. and McCleland, J. L. 1986. *Parallel Distributed Processing*. M.I.T. Pressn.

Ryder, J. 1995. Fuzzy COCOMO: Software cost estimation. Ph.D. thesis, Binghamton University.

Sandhu, P. S., Prashar, M., Bassi, P., and Bisht, A. 2009. A model for estimation of efforts in development of software systems. In *World Academy of Science, Engineering and Technology*. Vol. 56. 148–152.

Shepper, M. and Schofield, C. 1997. Estimating software project effort using analogies. *IEEE Tran. Software Engineering 23*, 736–743.

Sheta, A. 2006a. Software effort estimation and stock market prediction using takagi-sugeno fuzzy models. In *Proceedings of the 2006 IEEE Fuzzy Logic Conference, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21*. 579–586.

Sheta, A., Rine, D., and Ayesh, A. 2008. Development of software effort and schedule estimation models using soft computing techniques. In *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE CEC 2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong, 1-6 June*. 1283–1289.

Sheta, A. F. 2006b. Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. *Journal of Computer Science 2,* 2, 118–123.

Sheta, A. F. and Aljahdali, S. 2013. Software effort estimation inspired by COCOMO and FP models: A fuzzy logic approach. *4,* 11, 192–197.

Stewart, R. D. 1991. *Cost Estimating*. John Wiley and Sons.

Suri, P. and Ranjan, P. 2012. Comparative analysis of software effort estimation techniques. *International Journal of Computer Applications 48,* 21 (June), 12–19. Full text available.

Uysal, M. 2008. Estimation of the effort component of the software projects using simulated annealing algorithm. In *World Academy of Science, Engineering and Technology*. Vol. 41. 258–261.

Zeng, H. and Rine, D. 2004. Estimation of software defects fix effort using neural networks. In *Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - Volume 02*. COMPSAC '04. IEEE Computer Society, Washington, DC, USA, 20–21.

Alaa F. Sheta received his B.E., M.Sc. degrees in Electronics and Communication Engineering from the Faculty of Engineering, Cairo University in 1988 and 1994, respectively. He received his Ph.D. from the Computer Science Department, School of Information Technology and Engineering, George Mason University, Fairfax, VA, USA in 1997. He published more than 100 papers and book chapters. He published two books in the area of Landmine Detection and Classification and Image Reconstruction of a Manufacturing Process by LAP LAMBERT Academic Publishing. He is also the co-editor of the book entitled," Business Intelligence and Performance Management - Theory, Systems and Industrial Applications" by Springer Verlag, United Kingdom, published in March 2013. He received the Best Poster Award from the SGAI International Conference on Artificial Intelligence, Cambridge, UK on December 2011 for his publication on Quality Management of Manufacturing Processes. He is the founder and Chairman of the Advanced Computation for Engineering Applications (ACEA) workshop. This workshop was organized five times in Egypt, Jordan and Saudi Arabia. Recently, he was nominated as the Program Chair of the Science and Information Conference 2013 sponsored by the SAI Organization and Co-Sponsor by the IEEE Computational Intelligence Society, London, UK on October 2013. He has been an invited speaker in number of national and international conferences. He worked as a consultant for the Ministry of Communication and Information Technology, Egypt on the years 2002-2004. During then, he was the Vision Group Leader for the Information Infrastructure Program and project manager of the Economic Activities Project developed in collaboration with the Egyptian Ministry of Agriculture and both the Cancer Registry Network and the Telemedicine Projects with the Egyptian Ministry of Health. He was also hired as a part time consultant by the UNDP for the Smart School Pilot Project on the year 2003. Prof. Sheta is a member of the IEEE Evolutionary Computations and ACM societies. He was also the Vice President of the Arab Computer Society (ACS). He held number of management position during the years 2006-2009. He was the Vice Dean of Prince Abdullah Bin Ghazi Faculty of Science and Information Technology, Al-Balqa Applied University (BAU), Jordan (Fall 2008 - Spring 2009) and the Dean Assistant for Planning and Development with BAU, Jordan (Fall 2006-Summer 2008). His scientific research interests include Evolutionary Computation, Software Reliability Modeling, Software Cost Estimation, Modeling and Simulation of Dynamical Nonlinear Systems, Image Processing, Biotechnology, Business Intelligence, Robotics, Swarm Intelligence, Automatic Control, Fuzzy Logic and Neural Networks.

Dr. David C. Rine is Professor Emeritus of Computer Science, Volgenau School of Information Technology and Engineering at the George Mason University, USA. He received his Ph.D. in 1970 from The University of Iowa in the Mathematical Sciences Division with a Dissertation in Computer Science. Dr. Rine has received numerous awards from computer science and engineering societies and associations, including the IEEE Centennial Award, the IEEE Pioneer Award, the IEEE Computer Society Meritorious Service Award, the IEEE Computer Society Special Award, and the IEEE Computer Society 50th anniversary Golden Core Award. He has also been a multiple-time recipient of the IEEE Computer Society Honor Roll (and Distinguished Technical Service) Award and the IEEE Computer Society Certificate of Appreciation Award. His core areas of research are software systems engineering, computational science, information systems, computer science and science and engineering education. He has produced over 300 research papers and graduated over 40 PhD students during his career. The Army, Air Force, Navy, NATO, National Science Foundation, NASA, IEEE, ACM, IBM, AFIPS and many industrial organizations have funded Dr. Rine's scientific research and development work. IEEE, AFIPS, The College Board, Air Force, Educational Testing Service, George Mason University have funded his educational research and development. He has directed research and development projects that have integrated combinations of science, engineering and educational models. He directed the first model curriculum in software engineering as well as the first Advanced Placement programs in computer science.

Sofian Kassaymeh received his B.E. in Computer Information Systems from Arab Academy for Banking and Financial Sciences in 2007. Also he received his M.Sc. degrees in Management Information Systems/Software Engineering from Hasselt University in 2009. He was a Lecturer in Information Systems & Technology Faculty, University of Banking & Financial Sciences, Jordan (Oct. 2009 - Sept. 2011). Now he is a Research/Teaching assistance in Information Technology Dept./ College of Computers and Information System - Taif University, Saudi (Oct. 2011 - present). His scientific research interests include Evolutionary Computation, Software Cost Estimation, and Neural Networks.