A Real Time Extreme Learning Machine for Software Development Effort Estimation

Kanakasabhapathi Pillai[1] and Muthayyan Jeyakumar[2]
[1]Department of Electrical and Electronics Engineering, Kalaivanar Nagercoil Sudalaimuthu Krishnan College of Engineering, India
[2]Department of Computer Applications, Noorul Islam University, India

Abstract: *Software development effort estimation always remains a challenging task for project managers in a software industry. New techniques are applied to estimate effort. Evaluation of accuracy is a major activity as many methods are proposed in the literature. Here, we have developed a new algorithm called Real Time Extreme Learning Machine (RT-ELM) based on online sequential learning algorithm. The online sequential learning algorithm is modified so that the extreme learning machine learns continuously as new projects are developed in a software development organization. Performance of the real time extreme learning machine is compared with training and testing methodology. Studies were also conducted using radial basis function and additive hidden node. The accuracy of the Real time Extreme Learning machine with continuous learning is better than the conventional training and testing method. The results also indicate that the performance of radial basis function and additive hidden nodes is data dependent. The results are validated using data from academic setting and industry.*

Keywords: *Software effort estimation, extreme learning machine, real time, radial basis function.*

Received October 5, 2014; accepted March 30, 2016

1. Introduction

One of the major activities in software project management is Software Development Effort Estimation (SDEE). Recently machine learning methods and data mining techniques are getting more attention [3, 10, 14]. Problems of comparing one method with another arise as there are many criteria for accuracy evaluation. Accuracy also depends on the input data used for evaluation as well as the criteria used. Generally one can classify SDEE into four groups:

- Analogy based methods.
- Expert estimation.
- Model based such as Constructive Cost Model (COCOMO), Software Life Cycle Model (SLIM), etc.
- Artificial Intelligence (AI) methods such as neural networks, fuzzy logic, genetic algorithms or combinations there of Past projects data are used directly or indirectly in all the methods.

Analogy based methods compare the current project with past projects which is close to it. In expert estimation, opinion of experts is sought for effort values. In model based methods, relationship between effort and project parameters are obtained using historical data. Among the AI methods neural networks are most commonly used [2]. Here we have developed a Real Time Extreme Learning Machine (RT-ELM) which is easy to implement with only one parameter, number of hidden nodes, to be empirically selected.

The next section 2 gives related work followed by estimation problem in section 3. Data sets used are explained in section 4. RT-ELM is explained in section 5. Experimental results are provided in section 6. The last section 7 provides conclusions followed by references at the end.

2. Related Work and Motivation

Software development effort estimation continues to be a hot topic in spite of many persons from both industry and academia across many countries doing research. The major problems are related to the input data, algorithm, and accuracy evaluation criteria. One needs to consider all these three factors to arrive at a conclusion. Boehm *et al*. [1] suggest that no one technique should be relied upon for SDEE. Instead, multiple methods should be compared for decision making. The SDEE is a function of input where the size of the software projects plays an important role. The Extreme Learning Machine (ELM) which has many advantages over conventional feed forward back propagation neural network is gaining acceptance in many areas [6]. In addition to being extremely fast, it has better generalization capability than conventional back propagation networks. The design of the ELM is easy and straight forward. These motivated the authors to apply ELM for software development effort estimation. The ELM is applied for maintainability prediction for object oriented systems [11].

In a software development organization, project bidding happens one by one or chunk by chunk. Projects are developed in parallel and they are completed one by one or chunk by chunk. This necessitates SDEE in the beginning of the project. This is mainlyan effort prediction activity. At the end of the project, we know the actual effort and the model used in the beginning can be updated. Online sequential estimation is ideal for this situation. Hence, we have used Online Sequential Extreme Learning Machine (OS-ELM) [7]. Standard OS-ELM has training and testing phases. In the application considered, the extreme learning machine continuously learns as projects are completed and learning never stops. We call it RT-ELM as the parameters of the network are updated as soon as the project is completed. Its performance is compared with the conventional training and testing method. We have also studied the effect of radial basis function and additive hidden node as both requires the same number of parameters for realization.

3. Estimation Problem

The SDEE generally consists of two stages, model building and model evaluation. These are also known as training and testing. A part of the measurements is used to build the model and the remaining data are used to test the model. Model may work well for training data but its generalization capability may be poor. In order to minimize this problem training data is split into two parts; one for training and the rest for validation where the parameters of the model are finalized. In the software industry, managers have to predict the development cost for project bidding. The cost mainly depends on the effort put by humans and so effort estimation plays an important role in the software houses. This is generally achieved by predicting the software size and using environmental parameters. In fact, the RT-ELM algorithm does not distinguish between the training and testing data. The estimation problem aims at finding a relationship between dependent and independent variables using the data. The model is continuously validated for its prediction capability. The model accuracy is evaluated by finding the statistical characteristics of the errors, (Actual effort– Predicted effort), for each data set.

4. Data Sets Used

The data sets used in this study consist of Desharnais [12] and Maxwell [13] publically available from Predictive Models in Software Engineering (PROMISE) repository, Lopez data [8], and International Software Benchmarking Standards Group (ISBSG) [9]. The statistical characteristics of the dependent variable effort, mean, standard deviation, minimum, median, maximum, in terquartile range, skewness and kurtosis are given in Table 1. Desharna is data consists of six attributes and dependent variable effort in hours. Size of the software is measured in Function Point. It has 77 data points out of which 60 are used for training and 17 for testing. In Maxwell data effort is measured in hours. Size is measured in Function Point. We have used one dependent (Effort) and one independent variable (Size). Out of 62 projects, 40 projects data are used for training and the rest, 22 are used for testing. Lopez data has 163 training data points and 68 testing data points totalling 231 data points. Two attributes of the project are new and changed code and reused code. Code size is measured in lines of code and effort in minutes. These projects are small in size and developed in an academic setting. In ISBSG data projects are developed in industrial environments at different countries. It has 532 high quality projects. Number of projects used for training is 350 and testing is 182. Nine attributes are used as independent variables and dependent variable effort is measured in hours. Size is measured in Function Point.

5. Real Time-Extreme Learning Machine (RT-ELM)

Artificial Neural Network has the ability to learn complex functions and is the most commonly used method among machine learning techniques for effort prediction [2]. However, the conventional back propagation neural network algorithm has the following disadvantages:

- Determining the number of neurons required in each layer.
- Fixing of learning rate parameter and momentum coefficient.
- Possibility of converging to a local minimum.
- Deciding stopping criteria.
- Iterative learning for determining the weights which takes a long time.
- Determining the number of layers to achieve the accuracy.

Among the above iterative learning is totally avoided in the recently developed ELM [4, 5, 6]. Unlike the popular belief, for a Single Layer Feed forward Network (SLFN), both input weights and hidden layer biases can be randomly chosen in ELM. This method is not only fast but also makes better generalization. This non-iterative method has revolutionized the usage of ELM in many applications. These characteristics motivated the authors to implement ELM for SDEE.

The ELM consists of a single hidden layer with L nodes. For N arbitrary distinct samples (x_i, t_i), having activation function g(x), random weights, w_i, random biases, b_i, output weights β_i, The output function is

$$\sum_i \beta_i g(w_i x_j + b_i) = O_j,\ j=1,...,N;\ i=1,..., L \quad (1)$$

Where $\beta = [\beta_1,\ldots\ldots, \beta_L]^T$is the vector of output weights between the hidden layer of L nodes and the output node.

$$\sum_j \| O_j - t_j\|=0,\ j= 1,..,N \quad (2)$$

i.e., there exist β_i, w_i and $b_i$such that

$$\sum_i \beta_i\ g(w_i x_j + b_i) = t_j,\ j=1,...,N;\ i=1,...,L \quad (3)$$

Let

$$H = \begin{bmatrix} g(w_i x_j + b_1) & ---- & g(wixj + b_L) \\ - & ---- & - \\ g(w_i x_N + b_1) & ---- & g(w_L x_N + b_L) \end{bmatrix},$$

$T = [t_1^T \ldots\ldots t_L^T]^T$

$$\text{Then, } H\beta = T \quad (4)$$

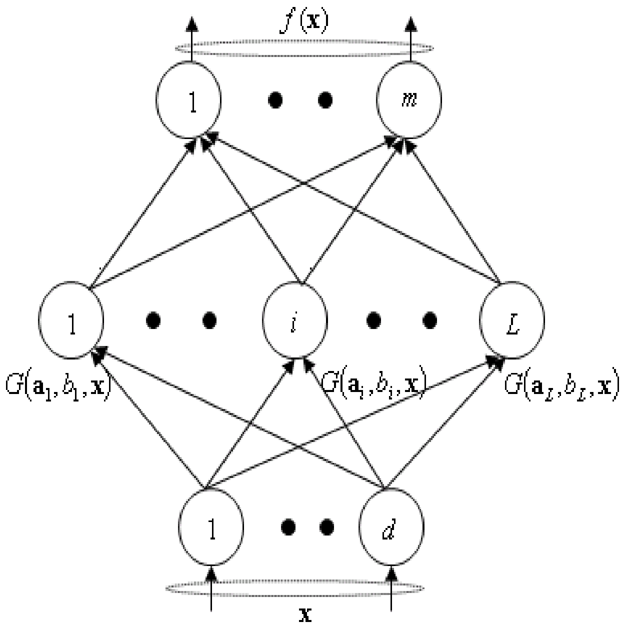
H is called the hidden layer output matrix of the neural network. If the activation function is infinitely differentiable it is proved in [6] that the required number of hidden neurons is $L \leq N$. Once the hidden node weights and biases are fixed, H remains fixed and one needs to estimate β. The minimum norm least squares solution for (4):

$$\beta = \min \| H\beta - T \| \quad (5)$$

If the number of hidden nodes is equal to the number of training samples, N, H is square and β can be determined by simply inverting H. However, in order to get better generalization, number of hidden nodes is adjusted and can be less than N. Then one needs to take the Moore-Penrose generalized inverse of H [6].

$$\hat{\beta} = H^{\dagger} T \quad (6)$$

Using $H^{\dagger}$, the Moore-Penrose generalized inverse of H; one can validate the algorithm for the total data. The architecture of ELM is shown in Figure 1.



$f(\mathbf{x})$
1
m
1
i
L
$G(\mathbf{a}_1, b_1, \mathbf{x})$
$G(\mathbf{a}_i, b_i, \mathbf{x})$
$G(\mathbf{a}_L, b_L, \mathbf{x})$
1
d
x

Figure 1. Extreme learning machine.

In a software industry, project bidding or project completion takes place either one by one or chunk by chunk. So the OS-ELM is suitable for this application.

OS-ELM involves initialization phase and sequential learning phase. Once the network learns, the model parameters are fixed and used for future prediction. However, the software development methodologies evolve over time. The development environment undergoes changes. The productivity of the programmers also changes. So the estimation model needs to be adapted to the changing situation. So, the learning process continues and there is no phase as testing. The detailed derivation of OS-ELM is available in [7] and a summary is provided here.

- *Step* 1: Initialization Phase: The ELM is initialized using a small chunk of data. Decide the number of hidden neurons (L). This is the only parameter we have to decide empirically. The initialization set (N_0) should be equal to or greater than (L).
- Assign random input weights, w_i, and bias, b_i, for additive (ADD) hidden nodes or centre, w_i, and impact factor, b_i, for Radial Basis Function (RBF) hidden nodes, $i = 1,\ldots, L$.
- Calculate the initial hidden layer output matrix H_0

$$H_0 = \begin{bmatrix} g(w_i x_j + b_1) & ---- & g(wixj + b_L) \\ - & ---- & - \\ g(w_i x_{N0} + b_1) & ---- & g(w_L x_{N0} + b_L) \end{bmatrix}, \quad (7)$$

H_0 is $N_0\ X\ L$ matrix.
Estimate the initial output weight

$$\beta^{(0)} = P_0 H_0^T P_0 T_o, \quad (8)$$

Where $P_0 = (H_0^T\ H_0)^{-1}$ and $To = [t_1,\ldots..,t_{N0}]$. Each t corresponds to actual effort of a project.
Set k = 0.

- *Step* 2: Sequential processing phase this is used for early phase of a project for prediction and for model update at the end of the project using actual effort data. Assuming one by one project data is processed, Calculate the partial hidden layer output matrix

$$h_{k+1} = [g(w_1,b_1,x_{k+1})\ldots\ldots g(w_L,b_L,x_{k+1})] \quad (9)$$

Where k is the processing instant.
Predict the effort required to complete the project

$$y_{k+1} = h^T_{k+1}\ \beta^{(k)} \quad (10)$$

Update the model parameters at the end of the project

$$P_{k+1} = P_k - P_k h_{k+1} h^T_{k+1} P_k (1 + h^T_{k+1} P_k\ h_{k+1}) \quad (11)$$

$$\beta^{(k+1)} = \beta^{(k)} + P_k h_{k+1}(t_{k+1} - y_{k+1}) \quad (12)$$

$$\text{Prediction error is} = t_{k+1} - y_{k+1} \quad (13)$$

Increment k and repeat step 2.

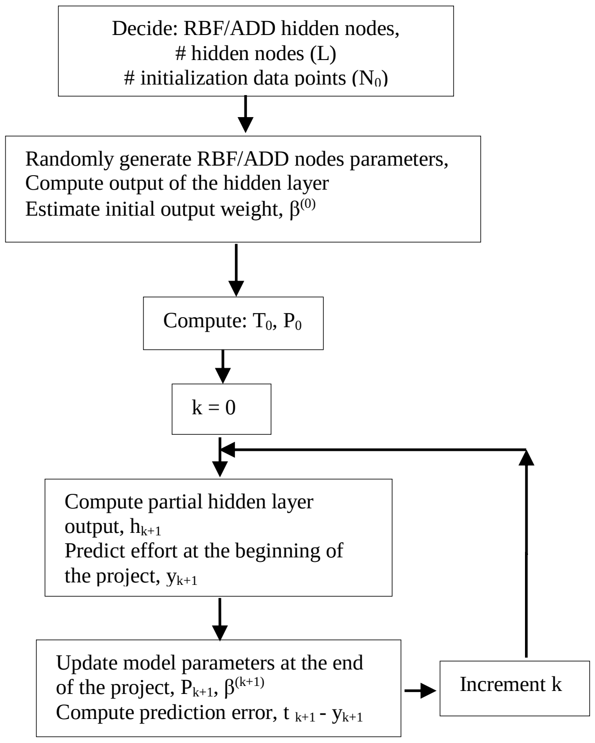

Decide: RBF/ADD hidden nodes,
# hidden nodes (L)
# initialization data points (N_0)
Randomly generate RBF/ADD nodes parameters,
Compute output of the hidden layer
Estimate initial output weight, $\beta^{(0)}$
Compute: T_0, P_0
k = 0
Compute partial hidden layer output, h_{k+1}
Predict effort at the beginning of the project, y_{k+1}
Update model parameters at the end of the project, P_{k+1}, $\beta^{(k+1)}$
Compute prediction error, $t_{k+1} - y_{k+1}$
Increment k

Figure 2. Flow chart of the proposed RT-ELM.

RT-ELM is same as OS-ELM with continuous learning. This method is best suited for software industry. The only parameter to be decided for RT-ELM is the number of hidden nodes. The network works for both additive and RBF hidden nodes. The number of parameters to be randomly selected is the same, but for impact width in RBF should be positive. The network performance can vary with respect to randomly selected parameters. These parameters are determined during the initialization phase. The RT-ELM flow chart is given in Figure 2.

6. Performance Evaluation of RT-ELM

The MINITAB® is used for studying the performance of RT-ELM. Initialization is done with five measurements. We have analysed the results for different number of samples for initialization. The RT-ELM performance is not affected by the number of samples beyond five. The input and output to RT-ELM are normalized between zero and one. Studies have been conducted with Radial Basis Function (RBF) and Additve (ADD) nodes. The initialization output of RT-ELM depends on the random input for centres and impact factors for the RBF nodes and weights and biases for the ADD nodes.

The study was repeated for 100 times and the random parameters for minimum mean square error were used for further studies. The minimum number of hidden nodes is fixed equal to the number of independent variables (attributes). Also, the initialization data points, N_0, should be greater than or equal to the number of hidden nodes, L. We have studied the standard way of fixing parameters using training and use them for testing. In RT-ELM, we have the flexibility of obtaining output vector, β, for each input. The statistical characteristics of the errors of both fixed and varying β for RBF and ADD nodes are provided in Tables 2, 3, 4, 5, 6, 7, 8 and 9 for the four different data sets. It can be observed from these tables that RMSE for RT-ELM, where the output weights, β, vary for all the inputs perform better than fixed weights for both additive and RBF nodes cases. This is true for the four different data sets studied here. Among additive and RBF nodes one can select based on RMSE of training data. But for Maxwell data RBF node performs better in terms of RMSE. Inter quartile ranges and standard deviations behave in a similar way. In all the cases correlation between observed and predicted data increases whenever RMSE decreases. For the Lopez data there is slight decrease in correlation although Root Mean Square Error (RMSE) has slightly decreased. The results clearly indicate the superior performance of RT-ELM compared to the conventional train and test method.

Table 1. Input data, effort, characteristic.

Data set	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis
Desharnais	4834	4188	546	3542	23940	3542	2.04	5.30
Maxwell	8223	10500	583	5190	63694	7209	3.35	13.70
Lopez	77.68	34.81	11	71	195	48	0.77	0.19
ISBSG	4603	7901	31	2185	61891	3614	4.12	20.60

Table 2. Performance for desharnais data with RBF nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	152	2236	-3798	-34	7308	2372	1.0	4.3	0.70	2223
Testing, β fixed	1040	3666	-4802	-61	9533	3494	1.0	3.6	0.50	3705
Testing, β varying	774	2519	-2272	75	7196	3528	1.1	3.5	0.74	2563

Table 3. Performance for desharnais data with ADD nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	486	2123	-3464	82.6	7308	2304	1.0	4.3	0.77	2161
Testing, β fixed	769	2811	-1786	-139	8676	2447	1.6	4.9	0.72	2833
Testing, β varying	557	2219	-1645	-112	7204	1962	1.8	5.8	0.87	2223

Table 4. Performance for maxwell data with RBF nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	-83	5194	-1047	-553	20689	3636	1.49	8.09	0.72	5129
Testing, β fixed	-1355	9046	-32820	-559	8995	3717	-3.20	14.01	0.64	8941
Testing, β varying	-83	3687	-7286	-823	8202	3694	0.18	3.29	0.65	3603

Table 5. Performance for maxwell data with ADD nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	1145	6865	-14354	-214	21773	6408	0.96	5.11	0.50	6875
Testing, β fixed	4427	2293	-7868	214	10518	7042	4.10	18.61	0.19	22835
Testing, β varying	1353	8664	-7468	-1.76	35214	6465	2.80	11.80	0.34	8572

Table 6. Performance for lopez data with RBF nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	1.42	23.78	-44.31	-0.58	72.92	30.61	0.50	2.82	0.71	23.75
Testing, β fixed	-12.80	32.68	-85.48	-8.58	47.92	42.34	-0.21	2.29	0.29	34.87
Testing, β varying	-8.94	29.46	-82.21	-4.77	49.42	49.47	-0.26	2.11	0.26	30.58

Table 7. Performance for lopez data with ADD nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	0.83	23.80	-45.31	-1.77	70.55	29.79	0.49	2.80	0.71	23.74
Testing, β fixed	-13.40	31.65	-88.85	-10.28	45.94	48.00	-0.27	2.21	0.28	34.15
Testing, β varying	-9.53	28.94	-83.66	-5.43	47.99	41.12	-0.21	2.34	0.31	30.27

Table 8. Performance for ISBSG data with RBF nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	389	4279	-10382	-424	37354	3228	3.12	22.75	0.52	4291
Testing, β fixed	454	7268	-16830	-579	53926	3644	3.76	24.54	0.46	7263
Testing, β varying	397	6983	-18147	569	53751	3055	3.70	25.48	0.48	6974

Table 9. Performance for ISBSG data with ADD nodes.

Data	Mean	St. Dev.	Minimum	Median	Maximum	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	363	4373	-10555	-362	44342	3137	3.95	35.31	0.52	4381
Testing, β fixed	-163	9911	-10375	-717	40332	3440	-5.43	69.61	0.46	9885
Testing, β varying	151	5625	-18247	-874	35581	2699	2.69	16.01	0.54	5611

7. Conclusions

In any software industry, projects are done either sequentially or in parallel. Predicting of effort at the beginning of the project and updating model parameters at the end of the project is natural which can be achieved by using RT-ELM where the output weights are updated for each project. It can be concluded from all the eight cases studied here, RT-ELM offers better accuracy than fixing parameters and predicting effort. It is expected that this methodology will play a major role in future.

References

[1] Boehm B., Abts C., and Chulani S., “Software Development Cost Estimation Approaches: A Survey,” *Annals of Software Engineering*, vol. 10, no. 1-4, pp. 177-205, 2000.

[2] Dave V. and Dutta K., “Neural Network Based Models for Software Effort Estimation: A Review,” *Artificial Intelligence Review*, vol. 2,

no. 2, pp. 295-307, 2012.

[3] Dejaeger K., Verbeke W., Martens D., and Baesens B., "Data Mining Techniques for Software Effort Estimation: A Comparative Study," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375-397, 2012.

[4] Huang G., Wang D., and Lan Y., "Extreme Learning Machines: A Survey," *International Journal, Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107-122, 2011.

[5] Huang G., Zhou H., Ding X., and Zhang R., "Extreme Learning Machine for Regression and Multi-Class Classification," *IEEE Transactions on Systems, Man, and Cybernetics-Part b: Cybernetics*, vol. 42, no. 2, pp. 513-529, 2012.

[6] Huang G., Zhu Q., and Siew C., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.

[7] Liang N., Huang G., Saratchandran P., and Sundararajan N., "A Fast and Accurate Online Sequential Learning Algorithm for Feed Forward Networks," *IEEE Transactions on Software Engineering*, vol. 17, no. 6, pp. 141-1423, 2006.

[8] Lopez-Martin C., "A Fuzzy Logic Model for predicting the Development Effort of Short Scale Programs Based Upon Two Independent Variables," *Applied Soft Computing*, vol. 11, no. 1, pp. 724-732, 2011.

[9] Mohammed Y., *Analogy Based Software Project Effort Estimation*, Ph.D. Thesis, University of Bradford, 2010.

[10] Nagpall G., Uddin M., and Kaur A., "Grey Relational Effort Analysis Technique Using Regression Methods for Software Estimation," *The International Arab Journal of Information Technology*, vol. 11, no. 5, pp. 437-446, 2013.

[11] Olatunji S., Rasheed Z., Sattar K., Al-Mana A., Alshayeb M., and El-Sebakhy E., "Extreme Learning Machine as Maintainability Prediction Model for Object-Oriented Software Systems," *Journal of Computing*, vol. 2, no. 8, pp. 49-56, 2010.

[12] PROMISEhttp://promise.site.uottawa.ca/SERepository/datasets/desharnais.arff, Last Visited, 2012.

[13] PROMISEhttps://code.google.com/p/promisedata/source/browse/trunk/effort/maxwell/maxwell.arff, Last Visited, 2012.

[14] Wen J., Li S., Lin Z., Hu Y., and Huang C., "Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models," *Information and Software Technology*, vol. 54, no. 1, pp. 41-59, 2012.

Kanakasabhapathi Pillai received B.E. in Electrical Engineering in 1971 from Madurai University, Madurai, Tamilnadu, India. He obtained M.Tech. from IIT Madras, Chennai, Tamilnadu, India, in 1973. After his masters, he joined Indian Space Research Organization and worked for 22 years. Then he joined NeST, Trivandrum, India as President and worked for eight years. Afterwards, he was Vice-President at HCL Technologies for six years. He was a Black Belt from American Society for quality and Master Black Belt from Indian Statistical Institute. Currently, he is working as Professor in the Department of Electrical and Electronics Engineering at K N S K College of Engineering, Nagercoil, India. He is also perusing his Ph.D. in Computer Science and Engineering. He has published more than 30 papers in peer reviewed journals and conferences. He is a senior member of IEEE and senior member of ACM. He is also a life member of Computer Society of India. He is Fellow of Institution of Engineers, India. His interests include soft computing and software engineering,

Muthayyan Jeyakumar received his Post Graduation Degree in Master of Computer Applications from Bharathidasan University, Trichirappalli, Tamilnadu, India in 1993. He fetched his M.Tech degree in Computer Science and Engineering from Manonmaniam Sundarnar University, Tirunelveli, Tamilnadu, India in 2005. He completed his Ph.D degree in Computer Applications from Dr. M.G.R Educational and Research Institute University, Chennai, Tamilnadu, India in 2010. He is at present working as Professor in the Department of Computer Applications and Additional Controller of Examinations, Noorul Islam Centre for Higher Education, Kumaracoil, Tamilnadu, India and he has twenty years of teaching experience in this reputed institution. He has published Thirty Six research papers in International and National Journals. He has also presented more than twenty research papers in International and National Conferences conducted by esteemed organizations. His research interests are Mobile Computing, Software Engineering and Network Security.