Global Journal of Computer Science and Technology
Volume 11 Issue 7 Version 1.0 May 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
ISSN: 0975-4172 & Print ISSN: 0975-4350

REBEE – Reusability Based Effort Estimation Technique using Dynamic Neural Network

By Jyoti Mahajan, Devanand, Kashyap Dhruve
University of Jammu

Abstract- Software Effort Estimation has been researched for over 25 years but until today no real effective model could be designed that could efficiently gauge the effort required for heterogeneous project data. Reusability factors of software development have been used to design a new effort estimation model called REBEE. This encompasses the usage of Fuzzy Logic and Dynamic Neural Networks. The experimental evaluation of the model depicts efficient effort estimation over varied project types.

Keywords: Software Effort Estimation, Reusability, Dynamic Neural Networks, Fuzzy Logic, REBEE .

GJCST Classification: D.2.9, F.1.1

REBEE REUSABILITY BASED EFFORT ESTIMATION TECHNIQUE USING DYNAMIC NEURAL NETWORK

REBEE – Reusability Based Effort Estimation Technique using Dynamic Neural Network

Jyoti Mahajan[α], Devanand[Ω], Kashyap Dhruve[β]

Abstract- Software Effort Estimation has been researched for over 25 years but until today no real effective model could be designed that could efficiently gauge the effort required for heterogeneous project data. Reusability factors of software development have been used to design a new effort estimation model called REBEE. This encompasses the usage of Fuzzy Logic and Dynamic Neural Networks. The experimental evaluation of the model depicts efficient effort estimation over varied project types.

Keywords- Software Effort Estimation, Reusability, Dynamic Neural Networks, Fuzzy Logic, REBEE.

I. Introduction

For over two decades now researchers have developed varied methods to estimate the software effort required to complete a software development project but till date no conclusive method has evolved. Software Effort Estimation is vital to arrive at development effort required for Project Management. Effective software effort estimation techniques not only enable fruitful resource allocation, resource scheduling, risk assessment but also assist in project monitoring. Effective Effort Estimation techniques are useful for fiscal estimates and delivery timelines too.

Effort Estimation techniques could be broadly classified as

- Parametric Effort Estimation Techniques

  The Parametric effort estimation technique assumes the Software Development Cycle to be completely sequential and automated. The software effort required is calculated based on a set of parameters. Once the effort estimated is derived the stringent processes are followed to meet the timelines.

The parametric effort estimation technique neglects the variance in the learning, execution and programming capabilities of every individual involved in the project. Technology dynamics also cannot be analyzed using the parametric effort estimation model.

- Top down approach based effort estimation techniques.

The top down approach considers the entire project as a whole and defragments it into various smaller units. Effort Estimations are carried out for the fragments mainly based on expert judgment. The top down approach does not account for technological changes, future uncertainties and risk mitigation techniques.

- Bottom up approach based effort estimation techniques

The bottom up approach considers using smaller project modules for the construction of the entire project. The entire effort estimation is a summation of the efforts involved in the smaller modules. This approach has drawbacks similar to the top down approach discussed above.

- Analogy Based Approach for effort estimation

The analogy based approach for effort estimation could be considered effective as it is capable to handle dynamics of technological platform transformations, varied human behavior, risk mitigation techniques etc. Prior knowledge about similar projects leads to effective estimation in this approach.

The software industry has experienced tremendous growth over the past decade. Currently Software development contracts are awarded to organizations having a previous experience in handling similar project or related projects. This is done in order to assure quality, reliability, financial security and most importantly timely delivery. Surveys conducted have found that many of these projects fail [1]. Some of the projects encounter effort overruns or schedule overruns sometimes both, due to un-appropriate estimation technique used [2] though prior experience and knowledge is available.

This paper proposes a Reusability Based Effort Estimation Technique (REBEE) to address the issue. REBEE embodies a Neuro-Fuzzy engine for estimation. To handle the dynamics of all the parameters involved in Software Effort Estimation Dynamic Neural Networks is used.

The manuscript is organized as follows. Section 2 describes the existing Effort Estimations Techniques used. Section 3 describes the prominence of reusability and its effects. Dynamic Neural Networks used in the REBEE is discussed in Section 4. Section 5 discusses the REBEE model. Section 6 provides a practical

About[α]- Computer Engineering Department, Govt. College of Engineering & Technology, Jammu, India
E-mail : jyoti_1972@sify.com
About[Ω]- Department of Computer Science & IT, University of Jammu, India
E-mail : padhadevanand@yahoo.co.in
About[β]- Planet-i-Technologies, Bangalore, India
E-mail : kashyap@vardhanatech.com

application example. The paper conclusions with summary and recommendations are provided in Section7.

It can be stated that

Delivery Time α Effort Required ………….. (1)

II. Background

Software Effort Estimation form it emergence has been achieved using various methodologies. COCOMO [3] and COCOMO 2.0[4], DELPHI [5], Function Point [6], Planning Poker[7], Use Case Point[8],Expert judgment [9], IBM – FSD [10] based estimation techniques are commonly used. All these models established have drawbacks leading to gross error of estimation. Researchers have used additional techniques along with these to achieve improved efficiency. COCOMO with effort adjustment factor [11] provided about 30% improvement in effort variance. COCOMO used with fuzzy logic, trapezoidal function and Gaussian functions showed improved performance [12].As the regularly used estimation techniques failed to provide consistency when tested against several cases. Multiple software effort estimation techniques were integrated and their combinations were linearly weighed providing better results [13].

A clustering approach bundled with Support Vector regression was found to provide good estimation accuracy [14]. The Mantel's correlation randomization test named Analogy-X greatly improved the estimation algorithm performance [15].The after effects of Schedule and Budget pressure on Effort Estimation and the development cycle time has been closely studied[16]. Chronological Splits have to be carefully assigned for effective training and testing purposes [17]. Judgment Based Systems[18] and recommendation based[19] effort estimation techniques provided satisfactory effort predictions. Global Software Developments being executed at diverse locations worldwide also encounter inaccurate estimation techniques [20].

With numerous efforts estimation techniques available and none providing homogenous results it becomes difficult to decide whether formal models like COCOMO etc or human judgment based systems could be considered ideal for developing effort estimation models [21]. A combination of judgment based and formal based models could be considered as a ideal solution. REBEE proposed in this paper is a combination of formal models developed using dynamic neural networks in addition to judgment based reusability matrices which is discussed in the next section.

III. Reusablity in Effort Estimation

Effort Estimation is a prominent feature of the software development cycle. As studied none of the existing models could effectively predict the software effort for varied types of software projects. The software industry is matured and experience in handling similar kind of projects has provided for additional software development projects of similar nature being offered to organizations. These organizations face a mammoth task of effort estimation. Conventional formal models do not predict the effort accurately as they have not incorporated the Reusability Factor within them. Most of the forms of software development would consist of reusable components like dynamic link libraries, functions, test cases ,web services , etc. Reusability of codes is analyzed seriously by software development organizations that are also considered for appraisals of programmers [22]. Reusable codes would allow software development houses to cut costs [23], reduce effort and maximize profits.

Reusability is a very important factor being analyzed by researchers. Reusability based cost estimation models have been analyzed and the incorporation of the reusable weights into the existing COSYSMO let to a new model called COSYSMO reuse extension[24]. The conventional taguchi model incorporated with reusability exhibited efficient effort estimation results [25]. Reusability incorporation with COCOMO81, COCOMO2 [26] and COCOMO [11] has been analyzed to understand the model performance. While developing REEBEE we considered the importance as well as the ill effects of reusability in the development of the model [27].

IV. Use of Dynamic Neural Networks in Rebee

Static Neural Networks possess learning and adaptive capabilities only for static input output relationships. But when we consider non linear mapping functions that exist in the matrices or parameters used for software effort estimation static neural networks would not be capable of handling the dynamics efficiently. REBEE is developed using dynamic neural networks (DNN). DNN are capable of providing instantaneous outputs for linear or non linear mapping functions that are required to effectively estimate the software effort required. A dynamic neuron unit (DNU) is considered as a basic computing block of the DNN.

A simple DNN *i* is as shown in Fig. 1.

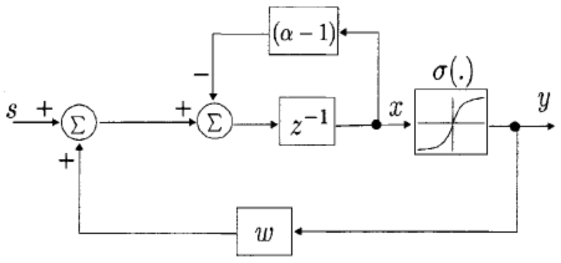

Fig.1. A simple i^{th} DNU

A simple structure of the DNN is shown in Fig. 2. given below.

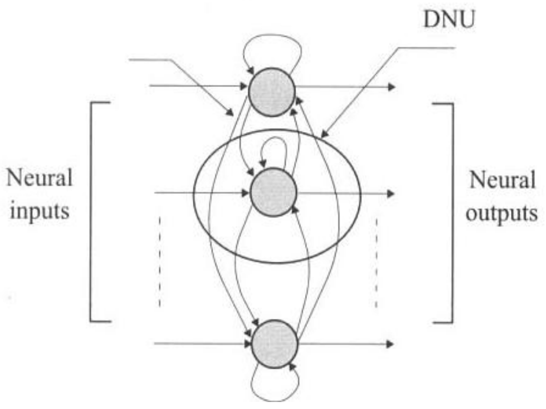

Fig.2. A Simple DNN Structure

Given a finite length discrete time sequence $x_d(k), k = 1,2,\ldots\ldots,N,$ we wish to design a discrete-time temporal learning algorithm such that the state of the following discrete-time dynamic neural unit (DT-DNU)

$$x(k+1) = -(a-1)x(k) + \sum_{i=1}^{n} a_i \sigma b_i x(k) + c_i) + s(k)$$

$$= -(a-1)x(k) + f(x(k),w) + (k)$$

$$= -(a-1)\,x\,(k) + a^T\sigma(bx(k) + c) + (k)$$

Will asymptotically track the sequence $x_{d(k)}$.

$$f(x,w) = \sum_{i=1}^{n} a_i \sigma(b_i x + c\,) = a^T \sigma\,(bx+c)$$

In this case, an error index with quadratic form is defined by

$$E\,(k) \quad = \tfrac{1}{2}(x_d\,(N)-x+(N))^2 + \tfrac{1}{2}\sum_{i=1}^{n-1}\,[x_d\,(k)-x(k)]^2$$

$$= \tfrac{1}{2}\,e^2\,(N) + \tfrac{1}{2}\sum_{i=1}^{n-1}\,e^2\,(k)$$

Where $e(k) = x_d\,(k) - x(k)$ and $e(N) = x_{d(N)} - x(N)$.

Using the discrete-time variational principle, a discrete-time lagrangian is defined by

$$\Phi \;= \tfrac{1}{2}\,(x_d\,(N)-x(N))^2 + \sum_{k=0}^{N-1}\{\tfrac{1}{2}x_d\,(k)-x(k))^2$$
$$-z(k+1)[x(k+1)+(a-1)x(k)-f(x(k),w)-s(k)]\}$$

$$= \tfrac{1}{2}e^2(N) + \sum_{k=0}^{N-1}\{\tfrac{1}{2}e^2(N) - z(k+1)[x(k+1)$$
$$+ (\alpha-1)x(k) - f(x(k),\boldsymbol{w}) - s(k)]\}$$

The reason that the disrete timev$(k+1)$ is associated with the Lagrange multiplier is due to the unfuziness of the final condition, as will be clear in the following discussion.

Similar to the method used for the continuous-time case, the first variation $\Phi$of may be represented as

$$\begin{aligned}\delta\Phi \;&=\; e(N)\delta x(N) + \sum_{k=0}^{N-1}\{e(k)\delta x(k)\\ &\quad - z(k+1)[\delta x(k+1) + (\alpha-1)\delta x(k) + x(k)\delta\alpha\\ &\quad - f_x(x(k),\boldsymbol{w})\delta x(k) - \boldsymbol{f}_w(x(k),\boldsymbol{w})^T\delta\boldsymbol{w}]\}\\ &=\; e(N)\delta x(N) + \sum_{k=0}^{N-1}\{[e(k) - (\alpha-1)z(k+1)\\ &\quad + f_x(x(k),\boldsymbol{w})z(k+1)]\delta x(k) - z(k+1)\delta x(k+1)\\ &\quad - z(k+1)x(k)\delta\alpha + z(k+1)(\boldsymbol{f}_w(x(k),\boldsymbol{w}))^T\delta\boldsymbol{w}\}\end{aligned}$$

Let the Lagrange multiplier $z(k)$satisfy,

$$z(k) = e(k) + [f_x(x(k),\boldsymbol{w}) - (\alpha-1)]z(k+1)$$

Or

$$z(k+1) = \frac{z(k) - e(k)}{f_x(x(k),\boldsymbol{w}) - (\alpha-1)}$$

Then

$$\begin{aligned}\delta\Phi \;&=\; e(N)\delta x(N) + \sum_{k=0}^{N-1}[z(k)\delta x(k) - z(k+1)\delta x(k+1)\\ &\quad - z(k+1)x(k)\delta\alpha + z(k+1)(\boldsymbol{f}_w(x(k),\boldsymbol{w}))^T\delta\boldsymbol{w}]\\ &=\; z(0)\delta x(0) - [z(N) - e(N)]\delta x(N)\\ &\quad + \sum_{k=0}^{N-1}[-z(k+1)x(k)\delta\alpha + z(k+1)(\boldsymbol{f}_w(x(k),\boldsymbol{w}))^T\delta\boldsymbol{w}]\end{aligned}$$

Since the initial value $x(0)$ does not depend on the parameters $\delta x(0) = 0$. If we choose additionally the final condition of the Lagrange multiplier

Global Journal of Computer Science and Technology Volume XI Issue VII Version I

$$z(N) = e(N)$$

Then,

$$\begin{aligned}\delta\Phi &= \sum_{k=0}^{N-1}[-z(k+1)x(k)\delta\alpha + z(k+1)(\boldsymbol{f}_w(x(k),\boldsymbol{w}))^T\delta\boldsymbol{w}] \\ &= \left(\sum_{k=0}^{N-1} -z(k+1)x(k)\right)\delta\alpha + \left(\sum_{k=0}^{N-1} z(k+1)(\boldsymbol{f}_w(x(k),\boldsymbol{w}))^T\right)\delta\boldsymbol{w}\end{aligned}$$

Therefore, the partial derivatives of the error index with respect to the parameters are given by

$$\begin{aligned}\frac{\partial E}{\partial \alpha} &= -\sum_{k=0}^{N-1} z(k+1)x(k) \\ \frac{\partial E}{\partial \boldsymbol{w}} &= \sum_{k=0}^{N-1} z(k+1)\boldsymbol{f}_w(x(k),\boldsymbol{w})\end{aligned}$$

And the incremental terms of the parameters are

$$\begin{aligned}\Delta\alpha(k) &= -\eta_\alpha\frac{\partial E}{\partial \alpha} = \eta_\alpha\sum_{k=0}^{N-1} z(k+1)x(k) \\ \Delta\boldsymbol{w}(k) &= -\eta_w\frac{\partial E}{\partial \boldsymbol{w}} = -\eta_w\sum_{k=0}^{N-1} z(k+1)\boldsymbol{f}_w(x(k),\boldsymbol{w})\end{aligned}$$

That is, the updating equations are obtained as

$$\begin{aligned}\alpha(k+1) &= \alpha(k) + \eta_\alpha\sum_{k=0}^{N-1} z(k+1)x(k) \\ \boldsymbol{w}(k+1) &= \boldsymbol{w}(k) - \eta_w\sum_{k=0}^{N-1} z(k+1)\boldsymbol{f}_w(x(k),\boldsymbol{w})\end{aligned}$$

The Learning algorithm given above for such a sequence learning problem consists of a discrete-time two-point boundary-value problem that can be solved, in general, by reiterative technique. Here, the initial condition $x(0)$ of the state is known, and the final condition $z(N)$ of the Lagrange multiplier is a linear function of the unknown final condition $x(N)$ of the state.

From the above discussion we can analyze the behavior and the working of the DNN trained using the Back Propagation Algorithm. A sigmoid function is used as the activation function.

V. REBEE

In this section we would discuss about the functional properties of REEBEE. The working of REEBEE could be understood through the following steps. Let us consider the dataset provided to us represented by D matrice. D is a $m\times n$ matrix represented as

$$D_{(m\times n)} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn}\end{bmatrix}$$

Where p are the Parameters of the Data Set.

Using Fuzzy Rules we now need to derive the reusability matrix represented as R of the parameters presented in the dataset. The matrix R consists of the effort taken to achieve the reusable part of the parameters p. The reusability matrix is a i×j matrix represented as

$$R_{(i\times j)} = \begin{bmatrix} r_{11} & \cdots & r_{1j} \\ \vdots & \ddots & \vdots \\ r_{i1} & \cdots & r_{ij}\end{bmatrix}$$

Let fzp(x) represent the fuzzy rule such that

$$R_{(i\times j)} = f_{zp}\left(D_{(m\times n)}\right)$$

The fuzzy matrix is provided to the DNN of the REEBEE to obtain the estimated effort matrix E. E also is a m×n represented as. p(m×n) corresponds to the effort estimated in man hours / man months

$$E_{(m\times n)} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn}\end{bmatrix}$$

VI. A Practical Application Example

To evaluate the robustness of REBEE we conducted experimental evaluations on 2 data sets obtained from a \$5 billion per year international technology firm. The datasets contained data of a testing and an enterprise application development project. The data was presented in spreadsheet formats. To interface these data sets we incorporated a "Import Wizard" in REEBEE which also embodied the Fuzzy Rules to derive the reusability matrix.

On Importing the dataset into REEBEE the Import Wizard Provided the Reusability Matrix to the DNN. The DNN was trained using the reusability matrix. The learning Rate used was set to 0.01, Number of Iterations were 10,000. The Sigmoid Function was used as the activation function. The experimental results obtained are as shown graphically in the figure below.

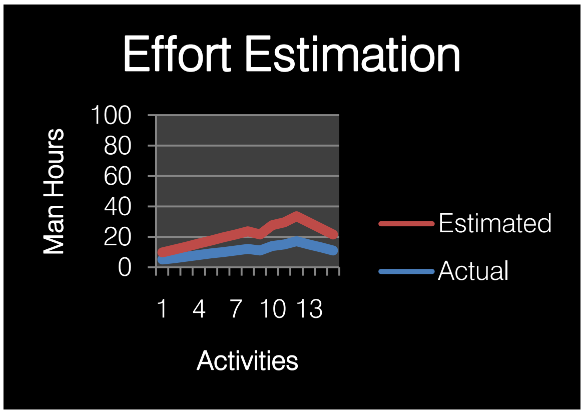

Fig.3. Effort Estimated and Actual Effort Time for Software Testing Project

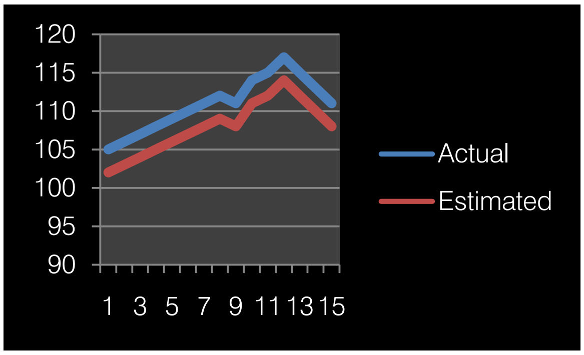

Fig.4. Effort Estimated and Actual Effort Time for Software Testing Project

From Fig. 3. and Fig. 4. we could conclude that the REBEE estimation is very close to the actual effort.

VII. Conclusion and Future Work

Effort Estimation is a critical operation of the Software Development Cycle. The existing estimation techniques do not provide similar estimation results for varied projects. Reusability factor is predominant in the current Software Development Cycle. The research work presented here introduced REEBEE, a reusability based effort estimation technique. REBEE is a neuro-fuzzy system which embodies fuzzy rules to derive reusability matrices and dynamic neural networks for effort estimation based on the reusability matrix. The experimental study conducted on data sets of 2 different projects showed impressive differences concluding that the REEBEE is capable of Effective Software Effort Estimation Techniques.

In the future we would like to further investigate the performance of REBEE over different project types and study its responses.

References Références Referencias

1. C. E. L. Peixoto, J. L. N. Audy, R. Prikladnicki, "Effort Estimation in Global Software Development Projects: Preliminary Results from a Survey," in Proc. 2010 *5th IEEE International Conference on Global Software Engineering, pp.123-127.*
2. K. Molkken, M. Jorgensen, "A Review of Surveys on Software Effort Estimation," Proc. 2003 *International Symposium on Empirical Software Engineering (ISESE'03), pp. 223.*
3. B.W. Boehm, W.W. Royce, Le COCOMO Ada, *Genie logiciel & Systemes experts*, 1989
4. B.W. Boehm, et al., "Cost Models for Future Software Life Cycle Processes: COCOMO2.0", *Annals of Software Engineering on Software Process and Product Measurement, Amsterdam,* 1995.
5. Website http://www.stellman-greene.com/aspm/images/ch03.pdf.
6. Meli, R., L. Santillo, "Function point estimation methods: a comparative overview", in Proc. 1999 *The European Software Measurement Conference – Amsterdam,* October 6-8.
7. http://www.mountaingoatsoftware.com/system/presentation/file/51/bayXP_070320_PlanningAgileProjects.pdf
8. S. Nageswaran "Test effort estimation using use case points" in *14th International Internet Software Quality Week 2001*, *San Francisco, California, USA,* June 2001.
9. M. Jørgensen, "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation," *IEEE Software*, vol. 22, no. 3, pp. 57-63, May/June 2005.
10. C.E. Walston, A.P. Felix, "A method of programming measurement and estimation," *IBM Systems Journal*, vol. 16, no.1, 1997.
11. M.J. Basavaraj, K.C Shet, "Empirical validation of Software development effort multipliers of Intermediate COCOMO Model" *Journal of Software*, vol. 3, vo. 5, pp 65 MAY 2008.
12. C.S. Reddy, KVSVN Raju, "An Improved Fuzzy Approach for COCOMO's Effort Estimation using Gaussian Membership Function". *Journal of Software*, vol. 4, no. 5, pp. 452-459, 2009.
13. C.J. Hsu, N.U. Rodas, C.Y. Huang, K.L. Peng, "A Study of Improving the Accuracy of Software Effort estimation Using Linearly Weighted Combinations," in Proc. 2010 *IEEE 34th Annual Computer Software and Applications Conference Workshops*, pp.98-103.
14. E. Kocaguneli, A. Tosun, A. Bener. "AI-Based Models for Software Effort Estimation" in Proc. *36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp.323-326.
15. J.W. Keung, B.A. Kitchenham, D.R. Jeffery, "Analogy-X: Providing statistical Inference to Analogy-Based Software Cost Estimation," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 471-484, July/Aug. 2008.
16. N. Nan, D.E. Harter "Impact of Budget and Schedule Pressure on Software Development Cycle Time and Effort" *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp 624-637, 2009.
17. C. Lokan, E. Mendes "Investigating the Use of Chronological Split for Software Effort Estimation" *IET*-Software, vol. 3, no. 5, pp 422-434, October 2009.
18. S. Grimstad, M. Jorgensen "Preliminary study of sequence effects in judgment-based software development work-effort estimation", *IET - Special Issue (EASE)* vol. 3 no. 5, pp 435-441, 2009.

Global Journal of Computer Science and Technology Volume XI Issue VII Version I

19. B. Peischl, M. Nica, M. Zanker "Recommending effort estimation methods for software project management", in Proc. IEEE*/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 03, pp 77-80, 2009.
20. C.E.L. Peixoto, J.L.N. Audy, R. Prikladnicki, "Effort Estimation in Global Software Development Projects: Preliminary Results from a Survey," in Proc. 2010 *5th IEEE International Conference on Global Software Engineering, ICGSE*, pp.123-127.
21. M. Jorgensen, B.W. Boehm, "Software Development Effort Estimation: Formal Models or Expert Judgment" *IEEE Software,* vol. 26, no. 2, pp. 14-19, Mar./Apr. 2009.
22. P.S. Sandhu, H. Singh, "Automatic Reusability Appraisal of Software Components using Neuro-Fuzzy Approach", *International Journal of Information Technology*, vol. 3, no. 3, pp. 209-214, 2006.
23. P.S. Sandhu, H. Kaur, A. Singh , "Modeling of Reusability of Object Oriented Software
1. Velmurugan T. and Santhanam T. (2010), "Clustering Mixed Data Points Using Fuzzy C-Means Clustering" retrieved from www.enggjournals.com/ijcse/doc/IJCSE10-02-09-112.pdf.
2. Worobey M, Gemmel M, Teuwen DE (2008) "Direct evidence of extensive diversity of HIV-1 in Kinshasa by 1960" retrieved volution.berkeley.edu/evolibrary/news/081101_ hivorigins - Cached - Similar.
3. Yang X. and Wand W. (2001) GIS Based Fuzzy C-Means Clustering Analysis of Urban Transit Network Service. The Nanjing City Case Study. Road and Transport Research China.
4. Zadeh L.A. (1965), "Fuzzy sets. Information and Control", Vol.8, pp.338-353.
5. Zain, M.F.M., Islam, M.N. and Basri, H. (2005), "An expert system for mix design of high performance concrete. Advances in engineering software", 36(5): 325 – 337.

System", *Journal of World Academy of Science, Engineering and Technology,* no. 56, pp 162 August 2009.

24. G. Wang, R. Valerdi, J. Fortune, "Reuse in Systems Engineering", *IEEE Systems Journal*, vol. 4, no. 3, pp 376-384, September 2010.
25. P. S. Sandhu, P. Blecharz, H. Singh, "A Taguchi Approach to Investigate Impact of Factors for Reusability of Software Components", *Journal of World Academy of Science, Engineering And Technology*, vol 25, pp 135-140, 2007.
26. CH.V.M.K.Hari, P.V.G.D.P Reddy, J.N.V.R.S Kumar, G.SriRamGanesh. CH.V.M.K. Hari., "Identifying the Importance of Software Reuse in COCOMO81, COCOMOII", *International Journal of Computer Science and Engineering JCSE,* vol. 1 no. 3, pp 142-147, 2009.
27. N. Ozarin, "Lessons Learned on Five Large-Scale System Developments" *IEEE Instrumentation & Measurement Magazine*, nol. 11, Issue- 1, pp 18-23, February 2008.