Estimating the Number of Test Workers Necessary for a Software Testing Process Using Artificial Neural Networks

Alaa F. Sheta
Faculty of Business and Technology
Princess Sumaya University for Technology
Amman, Jordan
asheta66@gmail.com

Sofian Kassaymeh
Information Technology Department
Taif University
Taif, Saudi Arabia
samsaak@gmail.com

David Rine
Emeritus of Computer Science
George Mason University
Fairfax, VA, USA
davidcrine@yahoo.com

***Abstract*—On time and within budget software project development represents a challenge for software project managers. Software management activities include but are not limited to: estimation of project cost, development of schedules and budgets, meeting user requirements and complying with standards. Recruiting development team members is a sophisticated problem for a software project manager. Since the utmost cost in software development effort is manpower, software project effort and is associated cost estimation models are used in estimating the effort required to complete a project. This effort estimate can then be converted into dollars based on the proper labor rates. An initial development team needs to be selected not only at the beginning of the project but also during the development process. It is important to allocate the necessary team to a project and efficiently distribute their effort during the development life cycle. In this paper, we provide our initial idea of developing a prediction model for defining the estimated required number of test workers of a software project during the software testing process. The developed models utilize the test instance and the number of observed faults as input to the proposed models. Artificial Neural Networks (ANNs) successfully build the dynamic relationships between the inputs and output and produce and accurate predication estimates.**

Keywords*—Staff Management; Neural Networks; Software Testing; Estimation

I. Introduction

”Software is a place where dreams are planted and nightmares harvested ... a world of were- wolves and silver bullets.” This quote from Brad Cox [1] defined the challenges for software project managers in the past as well as today. The software project manager needs to have the skills, techniques and monitoring and control tools to meet the goal of a software development project. The goal is to complete software development within the agreed upon cost, schedule and user expectations. The measure of meeting this goal includes: meeting a schedule and a cost through improving budget distribution, managing human resources and adapting to environment changes. Intelligent project management requires many talents and skills. In 1987, the IEEE standards provide the following definition of software project management: Software project management is the process of planning, organizing, staffing, monitoring, controlling, and leading a software project.

Software development has long been perceived as a risky business [2], [3]. A project manager can always try to predict the required resources and plan a schedule for a deliverable, but there is no guarantee that this is what will happen unless a careful monitoring and control plan is maintained. His/her ability to identify risks in advance could help planning for additional time to recover and reduce the consequence losses. According to Dr. Patricia Sanders, Director of Test Systems Engineering and Evaluation at OUSD, in her 1998 Software Technology Conference keynote address, 40% of the DoDs software development costs are spent on reworking the software, which in the year 2000 is equal to an actual loss of $18 billion. Furthermore, Sanders stated that only 16% of software development would finish on time and on budget. It was also stated in [4] that:

> Given that software-intensive projects are among the most expensive and risky undertakings of the 21st century, the investment in weapons from fiscal years 2003 through 2009 will exceed $1 trillion. Furthermore, many of the DoD’s most important technology projects will continue to deliver less than promised unless changes are made. Improving how we acquire software-intensive systems is both long overdue and an imperative.

In fact, the software development process is all about people, methodologies and tools. This can be seen from the software development process shown in Figure 1. People have to understand the project requirements, develop project plan and make a design, deployment of the project, test and validate the business requirements and finally fix bugs if any.

Software life cycle includes testing of the software system. The testing process requires significant effort and could cost over 50% of the project effort. This process requires a significant effort. It is defined as the process of executing a program with the intent of finding software bugs, errors or any defects [5]–[7]. It is also the process of validating and verifying that the developed software program will work and satisfies the needs of stakeholders. Software testing to be implemented needs a team of qualified personal. The team

Prof. A. Sheta is on leave from the Computers and Systems Department, Electronics Research Institute (ERI), Cairo, Egypt.

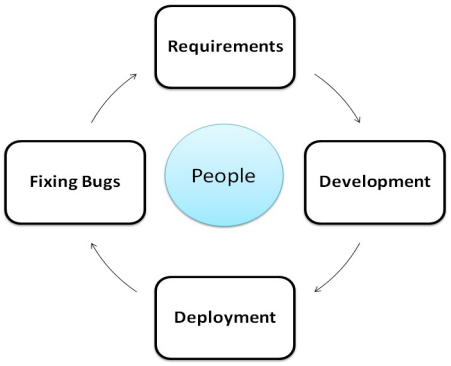

Fig. 1. Software development process

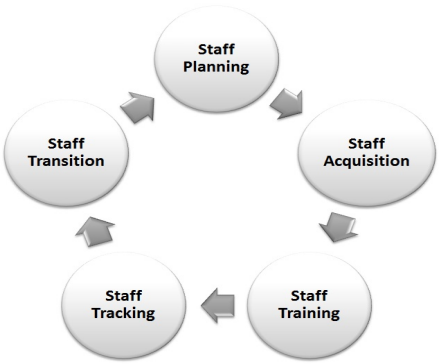

Fig. 2. Staff management process

size depends on many factors. These factors include the size and complexity of the developed software or program. Staff turnover means frequent replacement of the development personnel. This in fact is one of the significant problems a software project manager could deal with.

In this paper, we provide a non-parametric Artificial Neural Network (ANN) model for predicting the number of test workers required during the software testing process. The number of required test workers will depend upon the count of faults (defects) observed at certain test instances. The model should be capable of accurately defining the required team size for testing and also help project managers distribute the effort of his team on various tasks required for the project. In Section II, we present a definition to the staff management problem. Statistical Regression Analysis is presented in Section III. An overview of soft computing techniques and specifically Artificial Neural Networks is presented in Section IV. The evaluation criterion for measuring the goodness of the developed models are presented in Section VI. The two case studies considered in this article are presented in Sections VII and VIII. Finally, we present the conclusion and future work.

II. Staff Management

Time, cost, and number of staff estimations are essential duties for project managers in all business enterprises and especially for software projects. The manager needs to calculate an estimate for these main attributes in the early development process. This is not always an easy task for project managers. The role of a project manager is to manage, analyze and make decisions at all development phases according to accessible resources. Estimating time, cost, and staff helps sustain the monitoring and controlling of project activities and, in the end, produce quality. The field of software effort/cost estimation is concerned with providing an estimate of the expected cost, schedule, and manpower required to produce a software system. In fact there are common problems which could occur whenever we build a software system. The source of these problems could be one of the following:

- Insufficient requirements for the project
- Inadequate financial resources
- Loss of coordination because of many vendors
- Staff turnover

The staff management process for the project consists of the following five elements: Staff Planning, Staff Acquisition, Staff Training, Staff Tracking, and Staff Transition. This process is shown in Figure 2. Specific information related to staff need to be collected, organized and updated during the project development life cycle. The staff management process for the project consists of the following five elements: Staff Planning, Staff Acquisition, Staff Training, Staff Tracking, and Staff Transition. This process is shown in Figure 2. Specific information related to staff needs is to be collected, organized and updated during the project development life cycle. The staff management information collected should include:

- The adequate numbers of staff needed for each project phase.
- The contribution as a function of time staff member.
- The source of staff such as staff hiring, part timer hiring or consulting.
- The schedule for joining and leaving the project.

Staff management includes project cost. The manager needs to gather adequate information such that the estimated project cost can be computed. Many software effort/cost estimation models where proposed to help in providing a high quality estimate to assist a project manager in considering the best decisions for a project [8], [9]. Many software cost estimation models were reported in the literature [10]–[13]. These models were used to help project managers to estimate effort, time and cost.

Staff scarcities is considered as sources for either inefficient use of resources or delay in delivering the project. Computing staffing members needed for a project depends on correct predictions of the project demand and expected date of the product to be in the market. Any delay might cause business loss or damage to firm reputation. Numerous methods were used to compute the estimate and predict staffing needs, based on the firms past experience, project types and sales and manufacture statistics [14]–[17].

III. Statistical Regression Analysis

Statistical regression analysis associates relationships among a set of independent variables and one or more

dependent variables. The independent variables could be historical measurements about certain events in the past while we want to estimate or predict an independent variable at this instant of time or even in the future. Many techniques for carrying out regression analysis were evolved in the past. Linear regression and ordinary least squares regression are parametric methods that use Least Square Estimation (LSE) to estimate mathematical model parameters. COCOMO uses such regression methods.

A. Single Linear Regression

Regression analysis measures the degree of influence of the independent variables on a dependent variable. In the case of simple bivariate regression where there is a single independent variable, the dependent variable could be predicted from the independent variable by the simple equation:

$$y = a + bx + \epsilon \tag{1}$$

a is constant and b is the slope. This model is linear in the parameters a_i. y is called the independent variable and $x_i, i = 1, \ldots, n$ are called the independent variables. The goal is to find the relationship between the dependent and independent variables. To compute the regression coefficient for the single independent variable given in Equation 1, we use the formula:

$$b = \frac{\sum(x_i - \hat{x})(y_i - \hat{y})}{\sum(x_i - \hat{x})^2} \tag{2}$$

Where $\hat{x}$ is the mean (average) of the x values and $\hat{y}$ is the mean of the y values. The parameter a is computed by the formula:

$$a = y - bx \tag{3}$$

Equation 2 can be expanded to be:

$$b = \frac{(\sum y_i \sum x_i^2) - (\sum x_i \sum x_i y_i)}{n(\sum x_i^2) - (\sum x_i)^2} \tag{4}$$

B. Multiple Linear Regression

Equation 1 can be expanded to a multivariate concept as follows:

$$y = a_1 x_{i1} + a_2 x_{i2} + \cdots + a_n x_{ij} \tag{5}$$

Where x_{ij} is the i^{th} observation on the j^{th} independent variable. To show how the parameter estimation process works, we assume we have a system with four input variables x_1, x_2, x_3, x_4 and single output y. Thus, the model mathematical equation can be represented as:

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \tag{6}$$

To find the values of the model parameters a's we need to build what is called the regression matrix ϕ. This matrix is developed based on the experiment collected measurements. Thus, ϕ can be presented as follows given there is a set of measurements m:

$$X = \begin{pmatrix} 1 & x_1^1 & x_2^1 & x_3^1 & x_4^1 \\ 1 & x_1^2 & x_2^2 & x_3^2 & x_4^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^m & x_2^m & x_3^m & x_4^m \end{pmatrix}$$

The parameter vector θ and the output vector y can be presented as follows:

$$\theta = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} \mathrm{y} = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{pmatrix} \tag{7}$$

The least squares solution of yields the normal equation:

$$\phi^T \theta = \mathrm{y} \tag{8}$$

which has a solution:

$$\theta = \phi^{-1} \mathrm{y} \tag{9}$$

But since, the regression matrix ϕ is not a symmetric matrix, we have to reformulate the equation such that the solution for the parameter vector θ is as follows:

$$\theta = (\phi^T \phi)^{-1} \phi^T \mathrm{y} \tag{10}$$

IV. Soft-Computing Techniques

Soft Computing techniques were explored to build efficient effort estimation models structures [18], [19]. In the past, authors in [20] explored the use of Neural Networks (ANNss), Genetic Algorithms (GAs) and Genetic Programming (GP) to provide a methodology for software cost estimation. ANN were used for software engineering project management in [21]. Authors in [22], 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 [23], [24] were explored with promising results. A fuzzy COCOMO model was developed in [18].

Recently, In [12], author provided a pioneering set of models modified from the famous COCOMO model with interesting results. Later on, many authors explored the same idea with some modification [25]–[28] and provided a comparison to the work presented in [12]. Exploration of the advantages of the Takagi-Sugeno (TS) technique on building a set of linear models over the domain of possible software Kilo Line Of Code (KLOC) were investigated in [29]. Authors in [30] presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to utilize improved estimations of 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 model was evaluated using NASA software projects data set. A comparison between COCOMO-PSO, Artificial Neural Networks (ANNs), Halstead, Walston-Felix, Bailey-Basili

and Doty models were provided with excellent modeling results. In [31], a research work describes the Estimation of Projects in Contexts of Uncertainty (EPCU) model. The model is an estimation process based on fuzzy logic which has the objective of solving the project estimation problem taking the benefits of the Expert Judgment in a formal way, without using quantitative historic data.

V. What is ANN?

According to the Defense Advanced Research Projects Agency (DARPA) Neural Network Study (1988, AFCEA International Press, p. 60):

> ... a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

According to Nigrin (1993), p. 11 Nigrin1993, ANN was defined as:

> A neural network is a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore each element operates asynchronously; thus there is no overall system clock.

ANN can exhibit many brain-like behaviors such as learning, association, generalization, feature extraction, optimization and noise immunity. The basic simple unit of any ANN is the perceptron which is presented in Figure 3.

Artificial neural networks (ANN) have been proposed in many articles as a tool which was successfully able to develop software cost estimates. In [32], author provided a novel artificial neural network (ANN) prediction model which incorporates COCOMO and ANN-COCOMO II, to provide more accurate software estimates at the early phase of software development. ANN was employed to regulate the software features considering historical project data. In [33], authors provided a survey on the cost estimation models using artificial neural networks. ANN has many advantages they include:

- A neural network can perform tasks that a linear program cannot.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- A neural network learns and does not need to be reprogrammed.
- It can be implemented in any application.

The learning process in ANN is the algorithm which is used to adjust the weights of the network in order to minimize the difference between the actual and predicted values by the network. Usually, the weights of the network are initialized randomly. There are four basic types of learning rule: Error Correlation Learning (ECL), Boltzmann

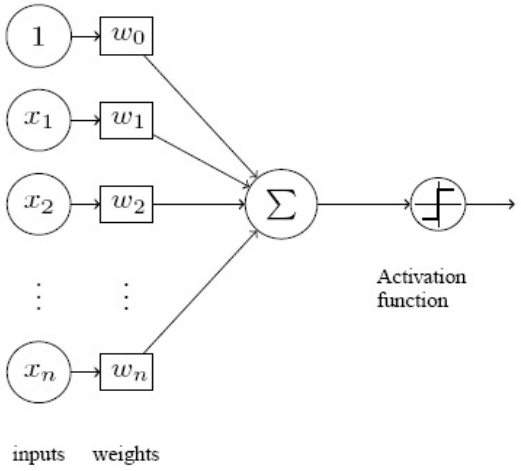

Fig. 3. The simple building block of ANN

learning (BL), Hebbian Learning (HL), and Competitive Learning (CL). The detailed descriptions of these learning rules are referred to the work of [34]. Among all the training algorithms, Back-Propagation (BP) which follows ECL rule is the most popular choice.

VI. Evaluation criteria

The performance of the developed two models; the Auto-Regression and the Artificial Neural Networks models will be evaluated using a number of evaluation criteria. They are:

- The Variance-Accounted-For (VAF) criteria was adopted by [35]:

$$VAF = \left[1 - \frac{var(y - \hat{y})}{var(y)}\right] \times 100\% \tag{11}$$

- The Mean square error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{12}$$

- The Euclidian distance (ED):

$$ED = \sqrt{(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{13}$$

- The Manhattan distance (MD):

$$MD = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{14}$$

- In [36], the authors provided an empirical study for data modeling in software engineering application and used radial basis function (RBF) to develop effort estimation model. They considered the mean magnitude of relative error (MMRE) as the main performance measure. We will evaluate the (MMRE) over the training and testing data as described in [36]. The mean magnitude of relative error (MMRE), defined as:

TABLE I. TEST/DEBUG DATA 1 x_1: TEST INSTANCES, x_2: REAL DETECTED FAULTS, y_k: NO. OF TEST WORKERS

x_1	x_2	y	x_1	x_2	y	x_1	x_2	y
1	5	4	38	15	8	75	0	4
2	5	4	39	7	8	76	0	4
3	5	4	40	15	8	77	1	4
4	5	4	41	21	8	78	2	2
5	6	4	42	8	8	79	0	2
6	8	5	43	6	8	80	1	2
7	2	5	44	20	8	81	0	2
8	7	5	45	10	8	82	0	2
9	4	5	46	3	8	83	0	2
10	2	5	47	3	8	84	0	2
11	31	5	48	8	4	85	0	2
12	4	5	49	5	4	86	0	2
13	24	5	50	1	4	87	2	2
14	49	5	51	2	4	88	0	2
15	14	5	52	2	4	89	0	2
16	12	5	53	2	4	90	0	2
17	8	5	54	7	4	91	0	2
18	9	5	55	2	4	92	0	2
19	4	5	56	0	4	93	0	2
20	7	5	57	2	4	94	0	2
21	6	5	58	3	4	95	0	2
22	9	5	59	2	4	96	1	2
23	4	5	60	7	4	97	0	2
24	4	5	61	3	4	98	0	2
25	2	5	62	0	4	99	0	2
26	4	5	63	1	4	100	1	2
27	3	5	64	0	4	101	0	1
28	9	6	65	1	4	102	0	1
29	2	6	66	0	3	103	1	1
30	5	6	67	0	3	104	0	1
31	4	6	68	1	3	105	0	1
32	1	6	69	1	3	106	1	1
33	4	6	70	0	3	107	0	1
34	3	6	71	0	3	108	0	1
35	6	6	72	1	3	109	1	1
36	13	6	73	1	4	110	0	1
37	19	8	74	0	4	111	1	1

TABLE II. EVALUATION CRITERIA OF THE ANN MODELS

Criteria	VAF	MSE	ED	MD	MMRE
MR	85.621%	0.559	7.881	0.526	0.148
NN	96.347%	0.143	3.994	0.243	0.076

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i} \quad (15)$$

Where y and $\hat{y}$ are the observed and predicted number of test workers the neural network model and n is the number of measurements used in the experiments, respectively.

VII. TEST/DEBUG DATA 1

Field report data was developed to measure system faults during testing in a real-time application [37]. The software system consists of 200 modules with each having one kilo line of code of FORTRAN. A Test/Debug dataset of 111 measurements is given in Table I. To develop a ANN test work estimate model, we used the data set to train the ANN. The observed and predicted number of workers was calculated based on the test instances and the real detected faults and shown in Figure 4. The convergence of the neural networks is shown in Figure 5 over 3000 epochs. The observed and predicted number of workers calculated based the test instances and the real detected faults is shown in Figure 5. The convergence of the neural networks is shown in Figure 6.

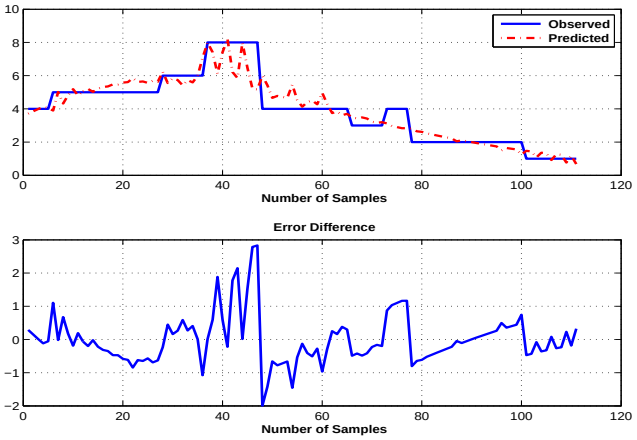

Fig. 4. Observed and predicted number of test workers using Multiple Regression Model: Test/Debug Data 1

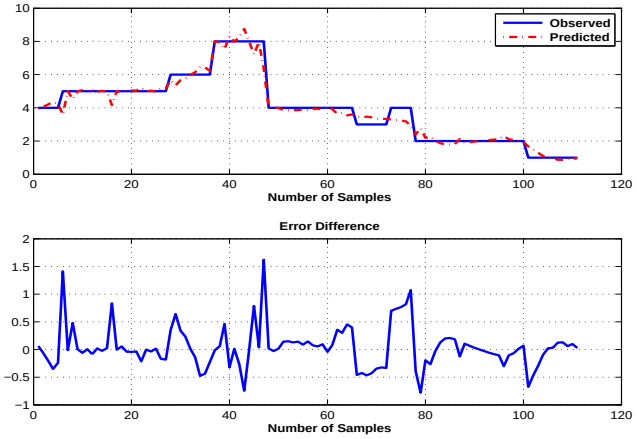

Fig. 5. Observed and predicted number of test workers using ANN: Test/Debug Data 1

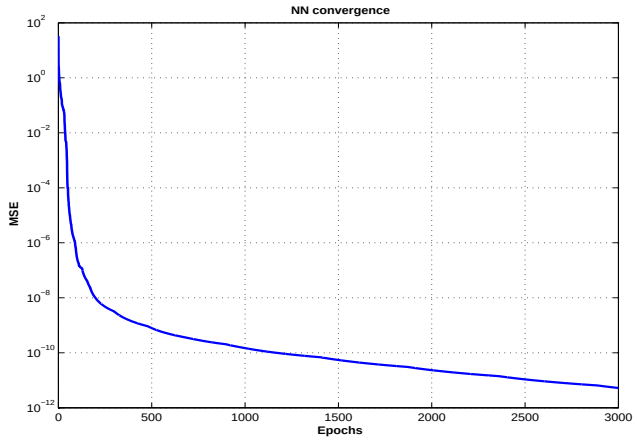

Fig. 6. NN Convergence using 50 neurons in the hidden layer: Test/Debug Data 1

TABLE III. TEST/DEBUG DATA 2 x_1: TEST INSTANCES, x_2: REAL DETECTED FAULTS, y_k: NO. OF TEST WORKERS

x_1	x_2	y	x_1	x_2	y
1	2	75	24	2	8
2	0	31	25	1	15
3	30	63	26	7	31
4	13	128	27	0	1
5	13	122	28	22	57
6	3	27	29	2	27
7	17	136	30	5	35
8	2	49	31	12	26
9	2	26	32	14	36
10	20	102	33	5	28
11	13	53	34	2	22
12	3	26	35	0	4
13	3	78	36	7	8
14	4	48	37	3	5
15	4	75	38	0	27
16	0	14	39	0	6
17	0	4	40	0	6
18	0	14	41	0	4
19	0	22	42	5	1
20	0	5	43	2	6
21	0	9	44	3	5
22	30	33	45	0	8
23	15	18	46	0	2

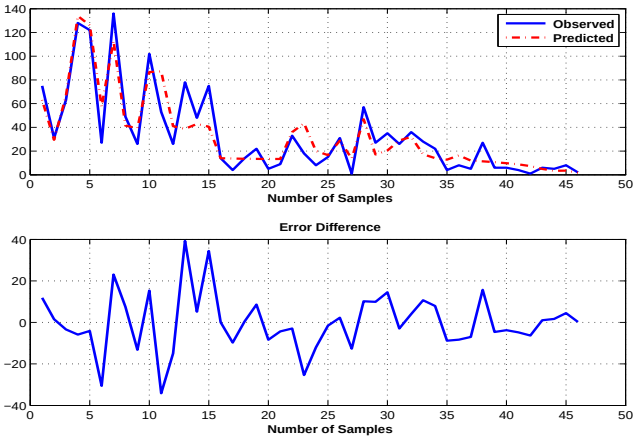

Fig. 7. Observed and predicted number of test workers using Multiple Regression Model: Test/Debug Data 2

VIII. TEST/DEBUG DATA 2

A Test/Debug data set has 46 measurements is given in Table III. The data set was presented in [37]. The number of measurements collected during the testing process is small. This represents a difficulty for traditional parameter estimation techniques. It is sometimes difficult to correctly estimate model parameters using a small number of measurements. To build a test work estimate model, we used the data set to build both the MR and ANN models. The observed and predicted number of workers calculated is based on the test instances and the real detected faults are shown in Figure 8. The convergence of the neural networks is shown in Figure 9 over 3000 epochs.

TABLE IV. EVALUATION CRITERIA OF THE ANN MODELS

Criteria	VAF	MSE	ED	MD	MMRE
MR	84.088%	188.73	93.175	9.992	0.931
NN	89.098%	129.99	77.328	7.252	0.707

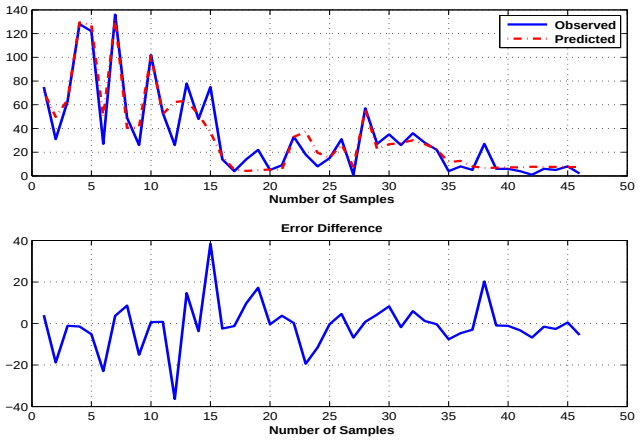

Fig. 8. Observed and predicted number of test workers using ANN: Test/Debug Data 2

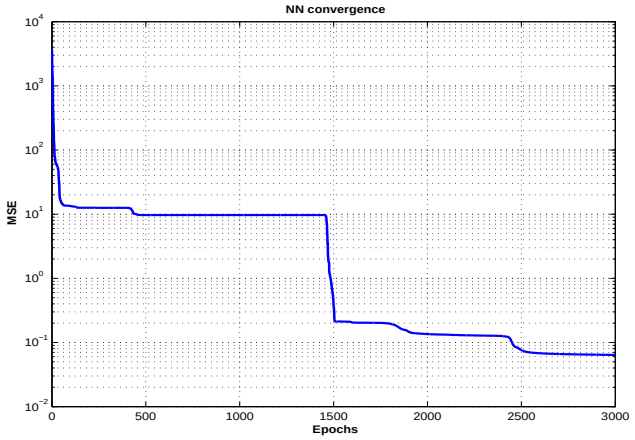

Fig. 9. NN convergence using 20 neurons in the hidden layer: Test/Debug Data 2

IX. CONCLUSIONS AND FUTURE WORK

Estimating the number of test workers during the software testing process became a challenge problem. Numerous methods were used to estimate and predict staffing needs, based on the firms past experience, project types and sales and manufacture statistics. Thus, tools and methods are required to fill the gap in this major area of software project life cycle development. In this paper, we propose our initial idea of developing predictive models for defining the estimated number of test workers of a software project during the software testing process using ANN. The developed models utilize the test instance and the number of observed faults as input to the proposed models. Two cases studies were presented and many evaluation criterions were used to validate the developed model performance. Artificial Neural Networks (ANNs) successfully build the dynamic relationships between the inputs and output and produce and accurate predication estimates. We plan to explore other soft computing techniques to handle this problem such as fuzzy logic to develop a mathematical relationship which can be easily explained in this case.

REFERENCES

[1] B. J. Cox, "Planning the software industrial revolution," vol. 7, no. 6. IEEE Software, 1990, pp. 25–33.

[2] W.-M. Han and S.-J. Huang, "An empirical analysis of risk components and performance on software projects," *J. Syst. Softw.*, vol. 80, no. 1, pp. 42–50, Jan. 2007.

[3] N. Ramasubbu and R. K. Balan, "Overcoming the challenges in cost estimation for distributed software projects," in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 91–101. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337235

[4] L. Pracchia, "Improving the dod: Software acquisition processes," *The Journal of Defense Software Engineering*, vol. 4, pp. 4–7, 2004.

[5] M. B. Cohen, M. B. Dwyer, and J. Shi, "Coverage and adequacy in software product line testing," in *Proceedings of the ISSTA 2006 Workshop on Role of Software Architecture for Testing and Analysis*, ser. ROSATEA '06. New York, NY, USA: ACM, 2006, pp. 53–63. [Online]. Available: http://doi.acm.org/10.1145/1147249.1147257

[6] I.-C. Yoon, A. Sussman, A. Memon, and A. Porter, "Effective and scalable software compatibility testing," in *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ser. ISSTA '08. New York, NY, USA: ACM, 2008, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/1390630.1390640

[7] T. Bultan and T. Xie, "Workshop on testing, analysis and verification of web software (tav-web 2008)," in *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ser. ISSTA '08. New York, NY, USA: ACM, 2008, pp. 311–312. [Online]. Available: http://doi.acm.org/10.1145/1390630.1390670

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

[9] ——, *Cost Models for Future Software Life Cycle Process: COCOMO2*. Annals of Software Engineering, 1995.

[10] K. Pillai and S. Nair, "A model for software development effort and cost estimation," *IEEE Trans. on Software Engineering*, vol. 23, pp. 485–497, 1997.

[11] C. Schofield, "An empirical investigation into software effort estimation by analogy," Ph.D. dissertation, Bournemouth University, 1998.

[12] A. F. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects," *Journal of Computer Science*, vol. 2, no. 2, pp. 118–123, 2006.

[13] A. F. Sheta, A. Ayesh, and D. Rine, "Evaluating software cost estimation models using particle swarm optimisation and fuzzy logic for nasa projects: a comparative study," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 6, pp. 365–373, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1504/IJBIC.2010.037016

[14] T. P. Bechet, *Strategic Staffing: A Comprehensive System for Effective Workforce Planning*. AMACOM Div American Mgmt Assn, 2008.

[15] H. Zeng and D. Rine, "A neural network approach for software defects fix effort estimation." in *IASTED Conf. on Software Engineering and Applications*, M. H. Hamza, Ed. IASTED/ACTA Press, 2004, pp. 513–517.

[16] U. Saxena and S. P. Singh, "Software effort estimation using neuro-fuzzy approach," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, 2012, pp. 1–6.

[17] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: An empirical study of commercial software projects," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 1042–1051. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486931

[18] J. Ryder, "Fuzzy COCOMO: Software cost estimation," Ph.D. dissertation, Binghamton University, 1995.

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

[20] M. A. Kelly, "A methodlolgy for software cost estimation using machine learning techniques," Master's thesis, Naval Postgratuate School, Monterey, California, 1993.

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

[22] J. J. Dolado and L. F. andez, "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, 1998, pp. 157–171.

[23] A. J. Albrecht and J. R. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Software Engineering*, vol. 9, no. 6, pp. 630–648, 1983.

[24] J. E. Matson, B. E. Barret, and J. M. Mellinchamp, "Software developemnet cost estimation using function points," *IEEE Trans. Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.

[25] H. Mittal and P. Bhatia, "A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers," *International Journal of Computer Science and Security*, vol. 1, no. 4, pp. 36–47, 2007.

[26] ——, "Optimization criteria for effort estimation using fuzzy technique," *CLEI Electronic Journal*, vol. 10, no. 1, pp. 1–11, 2007.

[27] M. Uysal, "Estimation of the effort component of the software projects using simulated annealing algorithm," in *World Academy of Science, Engineering and Technology*, vol. 41, 2008, pp. 258–261.

[28] P. S. Sandhu, M. Prashar, P. Bassi, and A. Bisht, "A model for estimation of efforts in development of software systems," in *World Academy of Science, Engineering and Technology*, vol. 56, 2009, pp. 148–152.

[29] A. Sheta, "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*, 2006, pp. 579–586.

[30] A. Sheta, D. Rine, and A. Ayesh, "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*, 2008, pp. 1283–1289.

[31] F. V. Souto, *Design Of A Fuzzy Logic Estimation Process For Software Projects: Estimation of Projects in a Context of Uncertainty EPCU Model*. Germany: LAP Lambert Academic Publishing, 2012.

[32] I. Attarzadeh, A. Mehranzadeh, and A. Barati, "Proposing an enhanced artificial neural network prediction model to improve the accuracy in software effort estimation," in *Proceedings of the 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks*, ser. CICSYN '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 167–172.

[33] M. Shepper and C. Schofield, "Estimating software project effort using analogies," *IEEE Tran. Software Engineering*, vol. 23, pp. 736–743, 1997.

[34] A. K. Jain, J. Mao, and K. K. Mohiuddin, "Artificial neural networks: A tutorial," *IEEE Computer Special Issue on Neural Computing*, pp. 31–44, 1996.

[35] R. Babuška, *Fuzzy Modeling and Identification Toolbox*. Delft University of Technology, The Netherland, http://lcewww.et.tudelft.nl/bubuska, 1998.

[36] M. Shin and A. L. Goel, "Emprirical data modeling in software engineering using radial basis functions," *IEEE Trans. on Software Engineering*, pp. 567–576, 2000.

[37] Y. Tohman, K. Tokunaga, S. Nagase, and M. Y., "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric districution model," *IEEE Trans. on Software Engineering*, pp. 345–355, 1989.