Information and Software Technology 51 (2009) 738–748

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Comparison of estimation methods of cost and duration in IT projects

Stanislav Berlin [a], Tzvi Raz [a,1], Chanan Glezer [b,*], Moshe Zviran [a]

[a] Faculty of Management, Tel Aviv University, Tel Aviv 69978, Israel
[b] Department of Industrial Engineering and Management, Ariel University Center of Samaria, Ariel 44837, Israel

ARTICLE INFO

Article history:
Received 28 January 2008
Received in revised form 29 July 2008
Accepted 22 September 2008
Available online 4 November 2008

Keywords:
Software engineering
Project cost estimation
Artificial neural networks
Regression

ABSTRACT

Producing accurate and reliable project cost estimations at an early stage of a project's life cycle remains a substantial challenge in the information technology field. This research benchmarks the performance of various approaches to estimating IT project effort and duration. Empirical data were gathered from various "real-world" organizations including several prominent Israeli high-tech companies as well as from the International Software Benchmarking Standards Group (ISBSG) IT project database. The study contrasts two types of models that have been employed to estimate project duration and effort separately: linear regression estimation models and models deriving from a more novel approach based on artificial neural networks (ANNs).

1. Introduction

Being first to market an IT product enables software providers and organizations to benefit from larger revenues and larger profits as well as to gain early feedback from the market for use in future development. At the same time, high-paced technological change leads to shorter product life cycles, which demand shorter research and development periods. These time-to-market pressures and environmental uncertainties make estimating the time and effort needed to develop an IT project one of the most challenging tasks that modern project managers face.

Estimates that are too low result in cost overruns, while estimates that are too high often mean missed financial opportunities. Considering that more than $300 billion are spent each year on approximately 250,000 projects, which equates to an average budget of $1.2 million, the financial implications are substantial [7], and bearing in mind Boehm's observation [6] that project estimates err in a range from 25% to 400%, starting early on in the project life cycle, we arrive at a financial variance of $300,000–$4.8 million. What makes estimation so difficult?

The difficulty in deriving accurate project cost estimations stems from a lack of detailed information about the project characteristics at an early stage in its life cycle. Most early life cycle estimation models use either the requirements' document or historical data for a size estimate as the foundation for formulating polynomial equation models. The list of reasons given by [19] for inaccurate estimation includes, among others: lack of data on completed previous projects; rapid changes in IT; difficulty in producing explicit requirements; lack of experience with similar projects and more.

The pressure to present very accurate estimates regarding anticipated completion rates, deadlines and other metrics as input to an ongoing project selection and activation process has led to the development of specific global practice methods for IT projects in recent years. PMBOK [1] divides all these tools into the following three categories: human expert judgment panels (e.g., the Delphi method); quantitative modeling (e.g., multivariate regression [10]) based on empirical data; and machine-learning (ML) techniques (e.g., artificial neural networks (ANNs)) based on training and testing some artificial intelligence differencing mechanism. However, none of these estimation techniques is fully satisfactory in terms of accuracy and applicability. Thus, with quantitative modeling, for example, the complexity of the equations, each subject to multiple interpretations, entails extensive human intervention [24]. With both quantitative modeling and ML there is a need for a detailed project database, which may or may not be available, depending on the maturity of the organization.

The objective of this study is therefore to benchmark the performance of the various approaches to estimating IT project effort and duration by applying several estimation techniques on data collected from a number of prominent Israeli high-tech companies as well as from a non-Israeli IT project database. To this end, the study examines the following techniques: linear regression and artificial neural networks (ANNs).

2. Research constructs

In this section we present the independent variables that were used to evaluate the effectiveness of the linear regression and the

* Corresponding author.
E-mail address: chanang@ariel.ac.il (C. Glezer).
[1] This manuscript is dedicated to the memory of the late Prof. Tzvi Raz.

0950-5849/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.09.007

artificial neural networks (ANNs) approaches to estimating project effort and duration, along with the way they were operationalized. The following measures were used in the operationalization: *historical productivity* – productivity in terms of size/persons month of recent projects; *size* – measured by the source line of code (SLOC); *complexity (functional size)* measured by the function points or number of files; *project duration* – actual elapsed duration of the project; *project effort* – actual effort in hours invested in the project.

2.1. Historical productivity

In economics, *productivity* is defined simply as a measure of productive efficiency calculated as the ratio of what is produced to what is required to produce it. The inputs taken as the denominator of the ratio are usually the traditional factors of production – land, labor, and capital – taken singly or in the aggregate. However, measuring the productivity of an engineer, or of an engineering process, is not at all intuitive: when the input is the experience, knowledge and time invested by a system developer and when the output is a computer or physical product, the relationship between input and output is not that straightforward. Difficulties also arise with reused codes or parts of projects. The reliability of the estimate depends on how closely the project correlates with past experience and on the ability of the expert to recall all the facets of the past project.

Based on the function point model [2], the following measure of software productivity has been proposed [22]: productivity = function points implemented/person-months, where the function points are calculated according to [16] and the person-months are delivered from the project database.

In this research, we used two databases: one was the ISBSG (International Software Benchmarking Standards Group) project repository – for the non-Israeli companies [21], and the other included data on projects executed by Israeli companies. Project productivity for the non-Israeli database was defined as delivery rate in hours per function point, calculated from summary work effort divided by adjusted function point count and retrieved from the ISBSG.[2] The productivity for Israeli companies' projects was delivery rate in hours per KSLOC (kilo source line of code) and was retrieved from project documentation.

2.2. Product size

It is generally recognized that requirements determine a product's size. Product size, in turn, determines project effort, and, finally, project effort determines project duration. Regarding the size variable, we simply try to get a feeling of how big the product we are going to develop is. Relying on COCOMO's model definitions [6], the source instructions, which are a fine-grained size measure, include all program instructions created by project personnel and they are processed into machine code by some combination of preprocessor, compilers, and assemblers. They exclude comments and unmodified utility software.

2.3. Product complexity (functional size)

The importance of the coarse-grained *functional size* measure to the project management process is widely acknowledged, insofar as it affects the project's time, cost and quality objectives, for example. Broadly speaking, higher product functional size is associated with greater complexity and leads to greater duration and cost [11]. Baccarini [5] proposed defining *product complexity* as the "many varied interrelated parts" of differentiation and interdependency. The work of [23] found that product complexity has no association with the technological performance, cost, schedule, or overall results of a project. However, [24] claimed that technology complexity, indicated by the number of technologies in the development effort, is positively associated with absolute development time. In [18] it was found that product complexity has a greater effect on cycle development time than novelty. Defining the complexity of the product is therefore an important determinant of the expected development cycle time, and the difficulties in evaluating complexity indices have compelled researchers to search for new methods.

In [4] a common technique that involves measuring complexity through so-called *function points* (FP) is used. It was originally used for large transaction-based software with a lot of file handling, but variants that are better suited to handle real-time systems have since been developed (see, e.g., [16,12]). For non-Israeli companies, we used raw data after professional function point analysis. For Israeli companies, function point data were not available and so as a surrogate measure we counted the number of diverse files in the product. The count of files included input and output files, read-from and written-to external applications, as well as files used internally in creating the product. This approach is analogous to function point analysis in expressing the functional size of a product, but it is not burdened with the complex calculations of function point analysis.

2.4. Project duration

Project duration was calculated as the total elapsed time for the project in calendar months. It should be noted that project duration covers all activities, including unplanned or planned lack of activity. This point will be discussed later in connection with estimation quality.

2.5. Project effort

Creating a highly accurate and reliable model to estimate the effort of high-tech engineering projects to the satisfaction of engineering practitioners is a challenging endeavor. PMBOK [1] defines the *project effort metric* as "the number of labor units required to complete an activity or other project element. This is usually expressed as staff hours, staff days, or staff weeks and should not be confused with duration." Data on project effort were retrieved from the project's documentation: timetables, and other project documents. For evaluation purposes, we used effort in person-hours recorded for the project. In [10], project size and development platform were used as mediating variables when investigating the effect of project effort on project duration.

3. Methodology

3.1. Data preparation

The Israeli companies' dataset consists of various firms from different high-tech areas, mostly software companies (Fig. 1). Data were gathered from a variety of sources within each firm including engineering lab books, manufacturing logs, project files, and the formal documentation of the development process. Although written evidence supported much of the data, some were reconstructed from the recollections of project personnel.

The non-Israeli database was obtained from the Australian branch of the International Software Benchmarking Standards Group (ISBSG) [21], which collects, validates, and publishes a repository of historical productivity data on software projects. Available since 2004 and now in its ninth release, this repository

[2] http://www.isbsg.org.

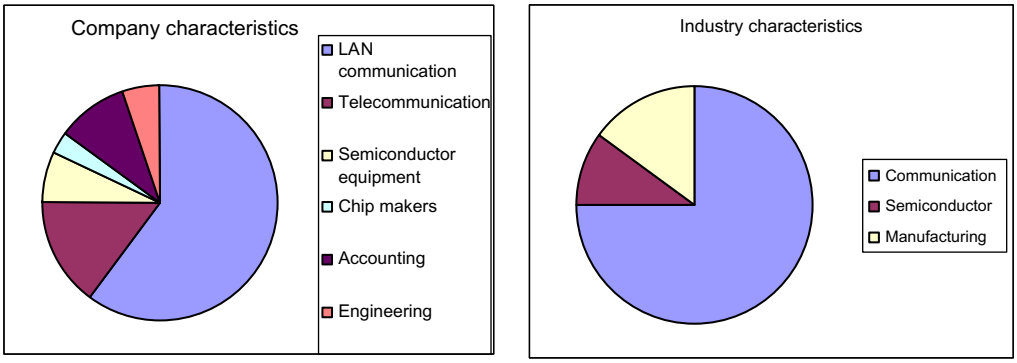


Fig. 1. Characteristics of Israeli companies.

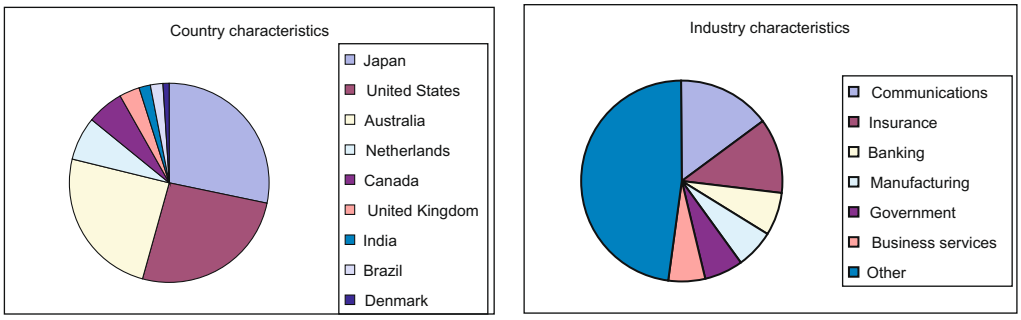


Fig. 2. Characteristics of non-Israeli companies.

contains historical data on 3024 software projects completed between 1989 and 2004 by a wide spectrum of companies worldwide (Fig. 2).

The first step in preprocessing the database was analyzing sampled data for the presence of outliers – observations that are unusual in comparison with the other data. In general, the presence of one or more outliers in a dataset tends to increase the value of the standard error of estimation. There are several methods to identify outliers. One is the boxplot graph, which is used as a visualization tool for outlier identification. Another is the leverage index, denoted h_i, which measures how far the values of the independent variables are from their mean values.

$$h_i = 1/N + \frac{(X_i - \bar{X})^2}{SSx}$$
$$SSx = \sum (X_i - \bar{X})^2$$

where

X_i – ith data sample.
$\bar{X}$ – simple average.
N – number of samples.

The rule of thumb is that if the leverage index does not exceed $6/N$, we do not detect influential observation in a dataset. One problem that may arise in using leverage to identify influential observations is that an observation can be identified as having a high leverage level but it may still not necessarily be influential in terms of the resulting estimated regression equation. Cook's distance measure uses both the leverage of observation i, h_i, and the residual for observation i, to determine whether the observation is influential.

$$D_i = \left(\frac{(Y_i - \hat{Y}_i)^2}{p - l}\right) * S^2 * \frac{h_i}{(1 - h_i)^2}$$

where

$(Y_i - \hat{Y}_i)^2$ the residual for observation i.
h_i – the leverage for observation i.
p – number of independent variables.
S – standard error of estimation.

In most cases, values of $D_i > 1$ indicate that the ith observation is influential and should either be studied further or removed from the database. After removing outliers, each database was randomly divided into two parts: the training part and the testing part. The number of samples in each set is presented in Table 1.

3.2. Analysis procedures

Fig. 3 depicts the computational procedures undertaken as part of this research. Two types of prediction models were used concurrently: linear regression, and artificial neural networks. These paths converge to a comparative analysis to derive recommendations for the most adequate estimation technique.

3.2.1. Linear regression

The first approach used for building a high-quality estimation model is that of traditional regression through a combination of

Table 1
Number of records in databases for duration and effort estimation

Database	Duration estimation	Effort estimation
Israeli database		
Training	89	96
Test	149	149
Non-Israeli database		
Training	535	797
Test	439	500

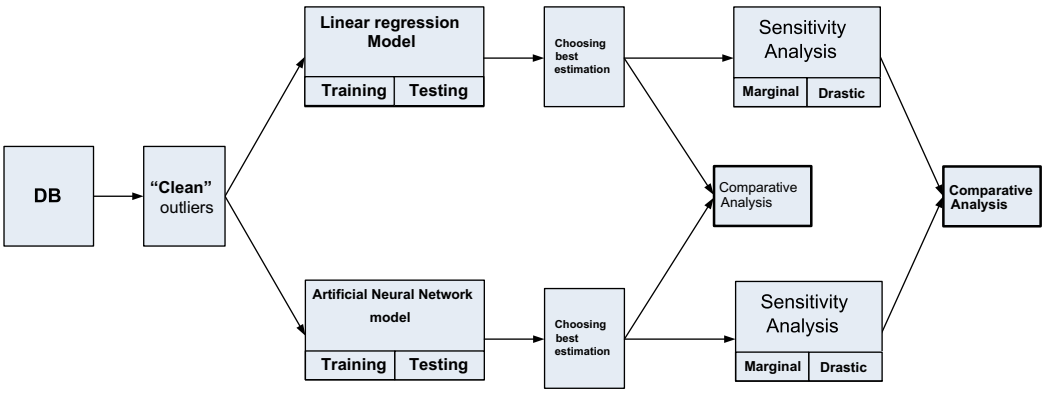


Fig. 3. Computational procedure.

technical methods and residual analysis. Note that transformation of one or more of the independent variables is often preferred over transformation of the dependent variable. Transforming *Y* not only changes the shape of the error distribution, but it also has an effect on the relationship between the *Y*s and the *X*s. The linear regression expression with logarithmic transformation provided the best fit among all tested models:

$$\widehat{Y} = B_0 * X_1^{B_1} * X_2^{B_2} * X_3^{B_3} * \cdots * X_n^{B_n} + \varepsilon$$

where

X is the independent variable.
$\widehat{Y}$ is the dependent variable.

Testing for the significance of the model is the next step. The first group of tests relates to the significance of the variables. A *t*-test explains whether a coefficient of a given independent variable differs from zero, the rule of thumb being that if the calculated *t*-value exceeds 2, then the coefficient is significant. Another test included in this group is the *F*-test, which tests for the significance of the model as a whole. Finally R^2 answers the question of the percentage of the whole dataset that is described by the given computed equations.

3.2.2. Artificial neural networks

An artificial neural network [3] has the ability to model complex non-linear relationships and is capable of approximating any measurable function. Starting from the explanation in [17] on how a simple neuron realizes the function of the inner product, the next step is to transform the weighted input vector into an output vector. Previous research indicates that it is vital to pre-process and normalize the input and output variables.

When the standardized values have disproportional spacing that is neither increasing nor decreasing monotonically across the interval, then we need a transformation that is non-monotonically non-linear and that approximately equalizes the number of component values in each sub-interval of equal length.

The performance of neural networks depends on the architecture of the network. Boetticher [8] limits the number of hidden layers and the number of hidden nodes per hidden layer to twice the number of inputs. For *N* inputs, the number of neural network architecture permutations equals: $1 + N + N^2$.

From [20], in which the question about how many layers were actually necessary and which activation functions could be used was discussed in detail, we have the following results: "A feed-forward artificial neural network with two layers of neuron and non-constant, non-decreasing activation function at each hidden neuron can approximate any piecewise continuous function from a closed bounded subset of Euclidean N-dimensional space..."

Hornik et al. [20] state that even architectures with one hidden layer can approximate any non-linear function. However, some researchers feel that although a single hidden layer is sufficient from a theoretical viewpoint, a large number of neurons may be required in the hidden layer. To keep the number of hidden neurons down, they sometimes use an extra hidden layer of neurons. On the basis of previous research, we decreased the raw number of architectures tested to two with one and two hidden layers. Table 2 presents all the permutations of the input variable tested for the two architectures.

Generally, it takes 6–7 trials to achieve the maximum results. After every trial (10,000 epochs), we interrupted the training process in order to check predicting quality indexes on a *testing set*. The process was finally concluded when quality index changes (margins) between two sequential series were insignificant. The main problem is that before learning, initial ANN weights are assigned randomly from a range [0;1]. Each trial may therefore lead to different results. Moreover, as ANNs attempt to minimize some goal function, starting from different points means that it is possible to reach different local minima and not always the optimum one. To diminish this possibility an optimization on the learning process was employed based on the PRED index for evaluating the predictive quality of the model, even though it rather optimistically focuses upon the best prediction. If the indicator did not change, the learning process was halted. In order to reduce variations between neural network models, several parameters were held constant. These included the learning rate and momentum, which were set to one.

3.2.3. Evaluation criteria

In addition to the basic mean square error index, we used several of its variations in this study to evaluate the predictive quality of the models, as shown in Table 3. Finally, we also used PRED(25), namely the percentage of predictions that fall within 25% of the actual value, though the 30% or 10% ranges are sometimes used.

It was important to be able to choose the most appropriate evaluation criteria, as well as to be able to compare them with previous

Table 2
Decreased number of neural network architecture permutations

Metric category	*N* inputs	*N* neural network architectures
SIZE	1	2
Productivity	1	2
Complexity	1	2
SiZE, Complexity	2	2
SiZE, Productivity	2	2
Complexity, Productivity	2	2
SiZE, Productivity, Complexity	3	2
Total network architectures		14

Table 3
Evaluation indices

Index name	Basic formula	Mean	Deviation
Mean square error	$MSE = (\widehat{Y} - Y_i)^2$	$MMSE = \frac{\sum(\widehat{Y}_i - Y_i)^2}{N}$	$DMSE = \left(\frac{\sum(MMSE - Y_i)^2}{N}\right)^{1/2}$
Balanced relative error	$BRE = \frac{\lvert\widehat{Y}_i - Y_i\rvert}{\min(\widehat{Y}_i - Y_i)}$	$MBRE = \frac{\sum\frac{\lvert\widehat{Y}_i - Y_i\rvert}{\min(\widehat{Y}_i - Y_i)}}{N}$	$DBRE = \left(\frac{\sum(MBER - Y_i)^2}{N}\right)^{1/2}$
Magnitude of error relative to estimate	$MER = \frac{\lvert\widehat{Y}_i - Y_i\rvert}{\widehat{Y}_i}$	$MMER = \frac{\sum\frac{\widehat{Y}_i - Y_i}{\widehat{Y}_i}}{N}$	$DMER = \left(\frac{\sum(MMER - Y_i)^2}{N}\right)^{1/2}$
Magnitude of relative error	$MRE = \frac{\lvert\widehat{Y}_i - Y_i\rvert}{Y_i}$	$MMRE = \frac{\sum\frac{\widehat{Y}_i - Y_i}{Y_i}}{N}$	$DMRE = \left(\frac{\sum(MMRE - Y_i)^2}{N}\right)^{1/2}$

$\widehat{Y}_i$ – actual ith value; Y_i – estimated ith values; N – number of samples, $i = 1, N$.

studies, so as to make our findings accessible to practitioners. We therefore chose to use MMSE, MBRE, MMER and MMRE, modifications of the MSE, BRE, MER, MRE indexes, respectively, which, like PRED, are suggested for evaluating the quality of project duration estimation. In fact, these parameters are the means of original indexes from a number of samples. The DMRE, DMSE, DMER, DBRE indexes represent deviations of the original indexes.

3.2.4. Sensitivity analysis

Sensitivity analysis tests the effect of fluctuations in input variables on the behavior of the model. It can indicate the model's similarity to the process under study; the quality of the model's definition; the main factors that contribute to output variability; the region within the space of input factors where the variation of the model is at its maximum; optimality or instability regions within the space of factors for use in subsequent calibration studies; and interactions between factors.

There are two main approaches to sensitivity analysis. The first, known as the *drastic* approach, determines the effects of making drastic changes in the input variables, like changing the input from its minimum to its maximum value. This type of sensitivity analysis should be distinguished from the second type – the *marginal* approach: making infinitely small changes (perturbations) in the input variables. This research performed both types of sensitivity analysis.

- *Drastic sensitivity analysis*: What ranges of input values (vectors) provide optimal results from the prediction capability point of view? Based on a graphical representation of BER, drastic sensitivity analysis serves to define regions with optimal prediction capability for a given independent variable.
- *Marginal sensitivity analysis*: Adding artificial noise to a model for the purpose of understanding the effects of changing input values on efficiency. The marginal sensitivity analysis added to the independent variables noise distributed normally according to the parameters $\sigma = \frac{S_i}{\sigma_q}\mu = S_i$, where S_i is the standard deviation of the ith sample. For our calculations we assumed $\sigma_q = 3$. This means that 99% of all values are located within the $3^*\sigma$ range. Adding such normally distributed noise and running the prediction model 10 times allowed us to find the value of the boundaries for the prediction capability indices for a given model.

The research plan included the sensitivity analysis for an individual variable (e.g., adding noise to the product complexity variable only) and groups of variables (e.g., adding noise concurrently to both product complexity and product size). Both types of sensitivity analysis were performed for each of the approaches to cost estimation – linear regression (best model), and neural network (best model).

4. Results

Fig. 4 presents a histogram of the raw independent variables. Table 4 presents the results of the Shapiro-Wilk normality test of all constructs operationalized in this study in the case of the non-Israeli companies. When W is significantly smaller than 1, the assumption of normality is not met. In our experience, smaller projects are undertaken more frequently than larger ones in most organizations, and the sample reflects this. Therefore, such data distributions offer weak support for linear regression. It was thus deemed appropriate to work on a mathematical transformation of the data.

Using the Tukey circle visualization [26], we looked for an appropriate mathematical transformation. Fig. 5 depicts the Lowess curve, which approximated the relationship between the input and output variable. By mentally superimposing the Tukey circle on the Lowess curve plots, we were able to choose the best fitting mathematical transformation model. The Lowess curve skirts the left hemisphere for all three plots. Thus, LOG(X) transformation can be chosen. Frequently, practitioners unite LOG(X) and LOG(Y) transformation and as result non-linear LOG transformation leads to an exponential expression. Fig. 6 depicts the independent variables after the LOG transformation. Table 5 compares prediction quality indices of various methods of duration and effort estimation for Israeli companies. Clearly the effort estimation PRED indices are far better than the indices of duration.

Fig. 7 serves for testing the linear regression assumption about uniform variance. Recall that the ordinary least squares (OLS) regression model assumes that $V(\varepsilon_i) = \sigma_2$ for all j. That is, the variance of the error term is constant (homoscedasticity). If the error terms do not have constant variance, they are said to heteroscedastic. Based on the Goldfeld-Quant rule, we evaluated the significance of the claim about heteroscedasticity by comparing the

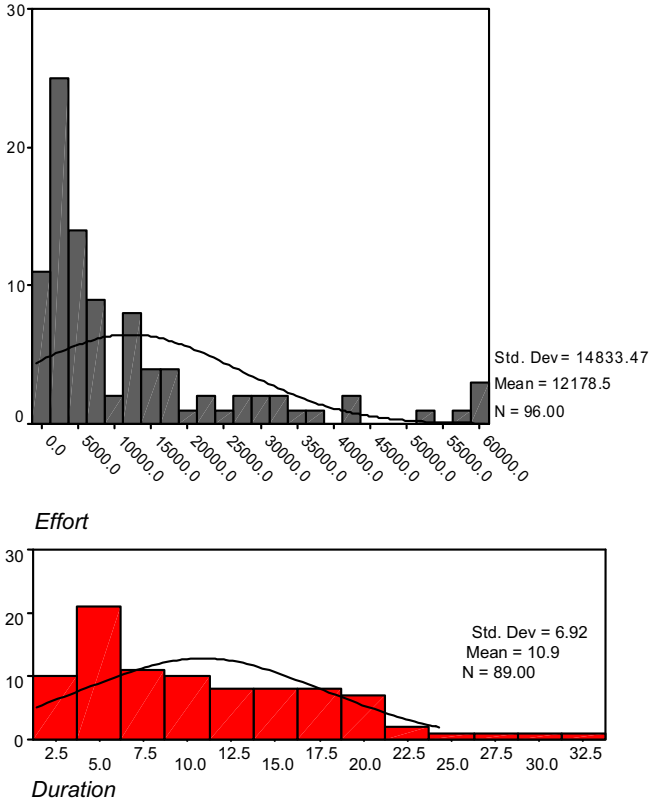


Fig. 4. Dependent variables – Israeli companies.

Table 4
Mathematical analysis of normality for input and output variables – Non-Israeli companies

Construct	Without transformation			LOG transformation		
	Statistics (W)	df	Significance	Statistics (W)	df	Significance
SIZE	0.722	637	0	0.989	637	0
Productivity	0.714	637	0	0.991	637	0
Complexity	0.669	637	0	0.970	637	0
Duration	0.711	637	0	0.985	637	0
Effort	0.493	797	0	0.991	797	0

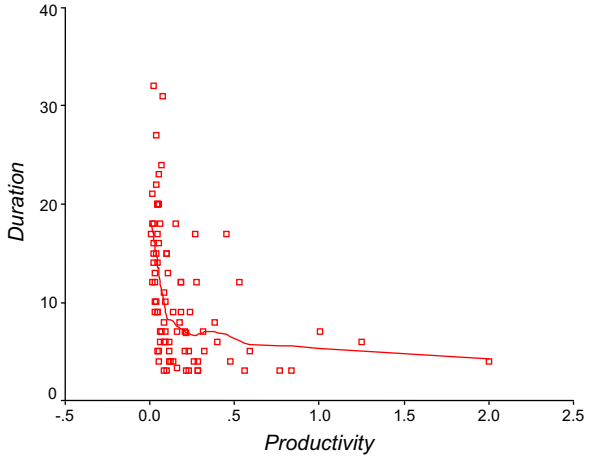


Fig. 5. The Lowess curve scatter plot for duration vs. productivity – Israeli companies.

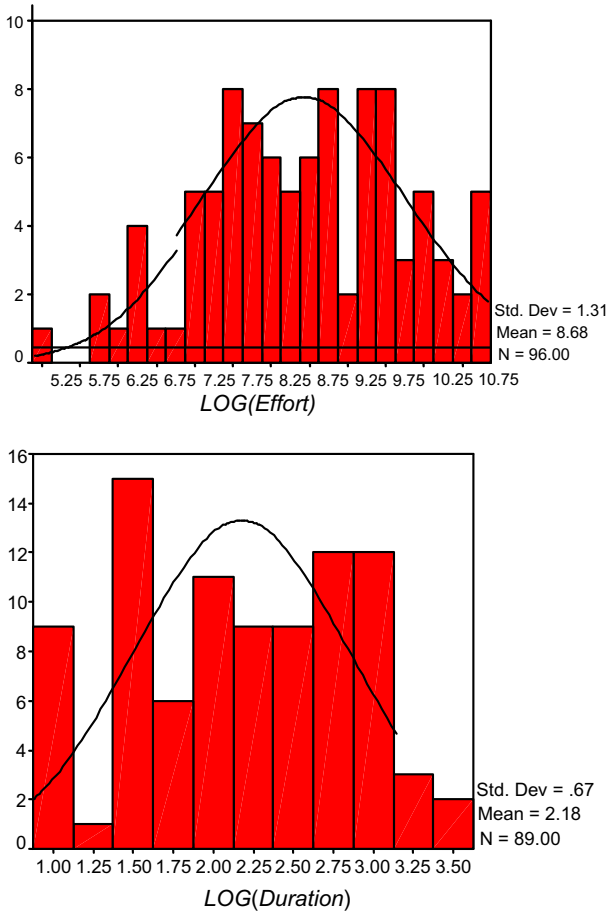


Fig. 6. Histogram of LOG-transformed dependent variables – Israeli companies.

difference of SSE in high and low values of the independent variables against a critical value. The results obtained for the non-Israeli companies and duration estimation indicate that productivity marginally violates the assumption of constant variance.

The expressions for prediction values after LOG transformation is

$$\hat{Y} = \text{Duration} = 0.08 * \text{Complexity}^{0.198} * \text{Productivity}^{-0.287} * \text{SIZE}^{0.254}$$

$$\hat{Y} = \text{Effort} = 0.049 * \text{Complexity}^{0.996} * \text{SIZE}^{0.002} * \text{Productivity}^{-0.996}$$

Physical validation of the equation leads to the following conclusions: product complexity and product size correlates positively with project duration (negligible for effort) and historical productivity correlates negatively. Tables 6 and 7 depict the results for Israeli companies' effort and duration estimation using neural networks (with both one and two hidden layers) across all possible input variable permutations.

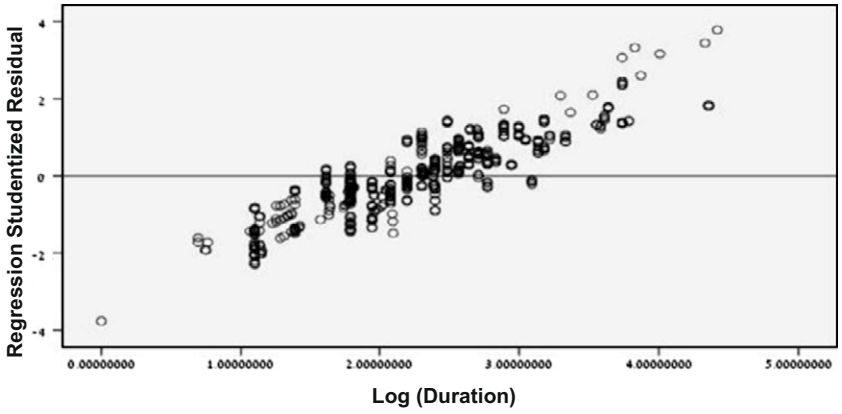
The best results, according to the PRED(25) index, are 56% obtained for estimating effort using an architecture with one hidden layer and three input variables. Table 8 depicts a comparative marginal sensitivity analysis for duration estimation (using the most effective sub-model). The models are not uniform in the influence of the normally-noised input variables on the overall prediction variation. For example, for the linear regression model for the Israeli dataset, product size contributes substantially to the instability of the whole model. However, for the ISBSG dataset, the influence of product size variable is minor.

Quite a different situation is apparent for effort estimation models. With minor assumptions, we can claim that regarding sensitivity as a function of input variables, analogous models are equal in every respect. Another finding is that for project effort estimation models, the effects of every set that included two noised input variable are larger than those of sets that included all three variables. Nothing similar was found for the duration estimation model. In practice, noise is an inherent part of data collection: input variables often suffer from additional noise. Therefore, the effort estimation model is expected to be more stable than the duration estimation model.

Table 5
Prediction quality indices duration – LOG–LOG regression estimation models

Category	MMSE	MMER	MMRE	MBRE	PRED(25) (%)
Israeli duration					
M	6.85	0.48	0.5	0.67	36.24
DEV	19.4	0.61	0.56	0.77	
Israeli effort					
M	7405.9384	0.0512248	0.051094	0.0541334	97.99
DEV	18938.537	0.0569993	0.0604135	0.0648108	
Non-Israeli effort					
M	37133840.36	0.23	0.22	0.29	80.90
DEV	265720998.6	0.54	0.42	0.66	
Non-Israeli duration					
M	144	0.5	0.88	1	27.40
DEV	549	0.57	1.28	1.3	

The drastic sensitivity analysis serves to define regions with optimal prediction capability. Table 9, which identifies the optimal region based on the BRE index, shows that the optimal ranges can be inferred based on the example MBRE limit $2 * \sigma \approx 2$, which covers about 95% of the total MBRE values. In truth, it should be noted



	SSE_{high}	SSE_{low}	F	$F_{25,25,0.01}$
LOG(Complexity)	80.401	108.26	1.34650067	~1.35
LOG(Productivity)	82.177	124.701	**1.51746839**	**~1.35**
LOG(SIZE)	88.906	113.748	1.27941871	~1.35

Fig. 7. Heteroscedasticity test: duration in non-Israeli companies.

Table 6
Israeli companies' dataset – neural network project duration

Input	*N* hidden layers		MMSE	MMER	MMRE	MBRE	PRED(25) (%)
Complexity	1	*M*	10.2	0.5	1.1	1.15	16.78
		DEV	17.5	0.3	1.12	1.11	
	2	*M*	0.8	0.5	1.08	1.14	16.11
		DEV	16.8	0.3	1.12	1.1	
Productivity	1	*M*	29	0.64	1.83	1.94	9.40
		DEV	36.5	0.57	1.9	1.93	
	2	*M*	30.8	0.62	1.86	1.96	11.00
		DEV	39.4	0.53	2	2	
SIZE	1	*M*	10.9	0.84	0.49	0.95	32.21
		DEV	28	1.071	0.46	1.094	
	2	*M*	10.8	0.83	0.49	0.95	32.89
		DEV	28.1	1.07	0.47	1.09	
SIZE, Complexity	1	*M*	9.55	0.69	0.5	0.84	33.56
		DEV	25.6	0.9	0.51	0.96	
	2	*M*	9.7	0.7	0.52	0.86	32.89
		DEV	24.7	0.88	0.56	0.97	
SIZE, Productivity	1	*M*	25.6	0.55	2.04	2.06	5.40
		DEV	22.2	0.25	2.13	2.12	
	2	*M*	101.3	0.58	2.6	2.62	7.38
		DEV	475	0.24	3.59	3.58	
Complexity, Productivity	1	*M*	7.34	0.46	0.8	0.88	23.50
		DEV	15.2	0.37	0.84	0.86	
	2	*M*	60	0.52	1.77	1.8	12.08
		DEV	523.7	0.27	3.5	3.51	
Complexity, SIZE, Productivity	1	*M*	8.3	0.67	0.44	0.77	38.00
		DEV	23.02	0.88	0.38	0.9	
	2	*M*	8.1	0.6	0.51	0.76	36.24
		DEV	21.3	0.7	0.52	0.8	

that these regions are not explicit. They contain a number of outliers, but most of the MBRE values are located inside them. Therefore, these intervals represent optimal ranges of an input variable for a given model.

Finally, the Wilcoxon signed-rank test was used to assess whether differences in prediction quality indexes were statistically significant. As a non-parametric equivalent of the paired Student's *t*-test for the case of two related samples, the Wilcoxon signed-rank test should be used whenever the distributional assumptions that underlie the *t*-test cannot be satisfied, especially the assumption of normal distribution for both samples.

Only the best sub-models were included in the comparative analysis. For the linear regression model, this was usually an exponential transformation. The sub-model with two hidden layers and three input variables was the best ANN model. In Table 10, which presents a comparative analysis of the dataset, the *Z* value represents the critical value of the test, while *P* represents the level of significance. For a two-level comparative test, 0.05 significance lev-

Table 7
Israeli companies – neural network project effort

Input	*N* hidden layers		MMSE	MMER	MMRE	MBRE	PRED(25) (%)
Complexity	1	*M*	9,330,444	0.66	2.95	3.02	12.00
		DEV	29,726,140	0.36	3.81	3.77	
	2	*M*	9,929,682	0.67	2.35	2.45	12.08
		DEV	35,708,135	0.51	2.68	2.65	
Productivity	1	*M*	86,666,848	0.71	5.72	5.72	6.04
		DEV	186,850,711	0.24	6.81	6.80	
	2	*M*	50,850,171	0.72	5.15	5.18	4.70
		DEV	90,892,073	0.28	6.28	6.26	
SIZE	1	*M*	13,146,185	1.90	0.72	2.15	22.15
		DEV	87,898,748	5.28	0.92	5.28	
	2	*M*	13,080,143	1.83	0.74	2.10	20.81
		DEV	87,711,133	5.03	0.97	5.04	
SIZE, Complexity	1	*M*	11,753,220	1.11	0.85	1.50	20.81
		DEV	80,980,901	2.04	1.14	2.22	
	2	*M*	11,201,287	1.00	0.94	1.48	20.13
		DEV	76,253,487	1.59	1.34	1.94	
SIZE, Productivity	1	*M*	10,470,184	0.59	1.56	1.67	21.48
		DEV	70,390,563	0.75	1.88	1.95	
	2	*M*	11,610,722	0.62	1.59	1.73	17.45
		DEV	74,471,923	0.95	2.00	2.13	
Complexity, Productivity	1	*M*	1,909,514	0.36	0.95	0.96	36.24
		DEV	5,046,072	0.24	1.37	1.37	
	2	*M*	1,656,438	0.28	0.70	0.71	51.01
		DEV	7,575,117	0.23	1.16	1.16	
Complexity, SIZE, Productivity	1	*M*	1,676,241	0.42	1.04	1.07	28.68
		DEV	3,414,173	0.24	1.49	1.48	
	2	*M*	5,444,148.6	0.27	0.52	0.53	56.38
		DEV	47,038,136	0.21	0.81	0.81	

Table 8
Comparing duration marginal sensitivity analysis result *PRED* (*25*) across models

Model	Israeli companies set	Non-Israeli companies set
Linear regression	SIZE (8.05%) > Complexity (4.03%) ~ Productivity (4.7%) (LOG Transformation)	Complexity (4.79%) ~ Productivity (4.57%) > SIZE (1.14%) (1/X Transformation)
	(Percentage of instability contributed by dependent variables)	
ANN	Productivity (6.04%) > SIZE (4.03%) ~ Complexity (4.03%)	Productivity (3.65%) ~ SIZE (4.34%) ~ Complexity (5.84%)

els should be used. Based on the Wilcoxon signed-rank test the linear regression is more favorable than the ANN across all quality indices only for Israeli companies in effort estimation. For other cases there is no conclusive advantage to either of the models.

5. Discussion

From the comparative analysis, we can draw the following conclusion: there is strong evidence that effort estimation is more precise than duration estimation. This result is congruent with [9] which found that project size correlates more strongly with effort than with duration since duration is estimated from effort which is not known in the early stages of the project. Another explanation is that effort for project development is defined as total hours spent on the project while duration is the total elapsed time in calendar months, including unpredictable interruptions of the development process. In practice, most project managers try to predict the total project time without bearing in mind the real time that should be spent on the project, which leads to poor quality prediction results.

It is evident from all the linear regression trials that the best solution for duration/effort project estimation is the use of non-linear transformation, mainly exponential transformation. Therefore, the non-linear relation between input and output variables was supported. The fact that the ANN demonstrated suitable results only strengthens our claim. In our trial for a neural network, we used a non-linear "tan-sigmoid" activation function, which permits the network to reproduce non-linear patterns in complex datasets. This means that the non-linear character of the neural network sits well with the datasets.

Concerning prediction capability, it should be noted that the ANN model provided adequate results with regard to predicting the project parameters. In our four trials, duration/effort project estimation for the Israeli/non-Israeli companies' datasets, the linear regression approach significantly outperformed the ANN model only for Israeli effort estimation. The feed-forward networks discussed thus far are among the simplest that exist in the neural network literature. For more complex network architectures, neurons in the same layer may be connected to one another and outputs may be connected to inputs or to hidden units, yielding a wide range of models characterized by simultaneity. Some advanced neural network techniques are related to more complex statistical methods such as kernel discriminate analysis, *k*-means cluster analysis or principal component analysis. Some neural networks, such as Kohonen's self-organizing maps and reinforcement learning, do not have any close parallel in statistics. These advanced methods exceed the scope of the present research and are not addressed here. However, they can be applied for project cost estimation purposes and have real potential for outstanding prediction performance. In addition, they can serve as a platform for future research. Our findings on this matter contradict those of [14] which found that the ANN outperformed regression. A possible explanation is that in their case there was a significant outlier which was left in the dataset. The ANN method which is not constrained to a linear function, can deal more successfully with observations

Table 9
Drastic sensitivity analysis

Model		Israeli companies		Non-Israeli companies	
		Duration	Effort	Duration	Effort
	SIZE (LOC)	[1000:10,000] (Optimal range of size based on MBER)	[1287;15,000]	[100;2570]	[2842;20,000]
Linear regression	Complexity (point)	[114;220]	[14;404]	No	[230;888]
	Productivity (hour per LOC)	[0.08;0.3]	No	[0.18;024]	[0.06;0.5]
	SIZE (SLOC)	[4910;13,000]	[1110;2700] [8430;15,000]	No[a]	No
Neural Network	Complexity (point)	[42;263]	[66;2170]	[230;1361] [1706;3000]	[230;2372] [3113;5291]
	Productivity (hour per LOC)	[0.03;063]	[0.06;1]	[0.03;0.09][0.1;0.59]	[0.06;1]
	SIZE (SLOC)	[1200;11,000]	[4910;16,000]	[12;2570]	No

[a] "No" means that there was no possibility to define the optimal range for a given variable.

that lie far from the best straight line. Nevertheless, industry practitioners consider ANNs a black box, while regression models are more intuitive and explanatory.

Regarding marginal sensitivity analysis, we scored input variables according to their effect on the model. Historical productivity, which has been shown to be an important input to software cost estimation models, plays a key role in both the linear regression and ANN prediction models. In addition, since an accurate measure of size is not available for estimation in the early stages of the development cycle, it would seem to be pragmatic to shift the objective from task parameter estimation to the estimation of a productivity coefficient that describes the effect of certain cost drivers and environmental factors on productivity [25].

For most trials (especially for effort estimation), the input variables are affected in the same manner in both the linear regression and ANN models. From a mathematical point of view this is not surprising. The mathematical expression describing the ANN model is similar to the formula that is presented in the linear regression models' past exponential transformation and both obey the exponential function.

Drastic sensitivity analysis revealed that neither the linear regression nor the ANN models have any advantage. Also, effort estimation is preferable to duration estimation, corroborating results that appear in the literature [9]. In sum, our conclusion is not surprising: the best prediction model is the one that is based on the most reliable input information (historical productivity) and estimates the less ambiguous output parameter – project effort, just as the expert attempts to predict the most stable and usable project parameters on the basis of past experience.

Additional observations follow from the comparative analysis of the Israeli and non-Israeli companies' datasets due to the fact that the product complexity variable is operationalized differently for each dataset: for the Israeli dataset, we counted the number of diverse files included in the design; for the non-Israeli companies, we used a function point calculation. When comparing the linear regression and neural network models, we made the following inferences. First, using the number of files as product complexity is extremely useful for the linear regression model. The model significantly outperforms the ANN model and demonstrates stronger results than the linear regression model when product complexity is operationalized based on function point calculation. This is particularly impressive considering that the non-Israeli dataset contained significantly more samples than the Israeli dataset. Thus, since using the function point as a basis for product complexity significantly promoted the results of the ANN model, even though the non-Israeli dataset included significantly more samples than did the Israeli dataset, there is no strong evidence that only a function point leads to outstanding results. Additional investigation must be carried out in order to clarify this issue.

6. Related work

This research relates and contributes to an already existing large body of literature on software estimation models and techniques. Examples of this work include the original parametric and regression-based models: function point analysis [2,4,16], COCOMO models [6] and the OLS regression model [14]. However, despite decades of research in this domain, software estimation is still considered a challenge for most researchers and practitioners, and no research approach has currently been recognized as having produced estimation models that are considered to perform well enough to meet market needs and expectations [14].

Table 10
Wilcoxon signed-rank test results

Statistics	Israeli		Non-Israeli	
	Duration	Effort	Duration	Effort
MBRE	(Neural network) > (Regression)	(Neural network) > (Regression)	(Neural network) < (Regression)	(Neural network) ⟨⟩ (Regression)
Z	−3.511	−9.114	−3.57	−0.492
P	0	0	0	0.623
MMRE	(Neural network) ⟨⟩ (Regression)	(Neural network) > (Regression)	(Neural network) ⟨⟩ (Regression)	(Neural network) ⟨⟩ (Regression)
Z	−1.187	−9.16	−0.338	−0.616
P	0.235	0	0.735	0.538
MMER	(Neural network) > (Regression)	(Neural network) > (Regression)	(Neural network) < (Regression)	(Neural network) ⟨⟩ (Regression)
Z	−4.919	−8.901	−5.038	−0.98
P	0	0	0	0.327
MMSE	(Neural network) > (Regression)	(Neural network) > (Regression)	(Neural network) < (Regression)	(Neural network) ⟨⟩ (Regression)
Z	−3.767	−9.166	−2.57	−1.862
P	0	0	0.01	0.063
PRED(25)	(Neural network) ⟨⟩ (Regression)	(Neural network) < (Regression)	(Neural network) ⟨⟩ (Regression)	(Neural network) ⟨⟩ (Regression)
Z	−0.649	−8.066	−1.257	−1.121
P	0.516	0	0.209	0.262

Other exploratory data analysis techniques, such as case-based reasoning [13,31], ANN and clustering, have been evaluated rigorously for estimating effort [14]. With accuracy comparable to algorithmic methods in some studies, case-based (analogy) estimation is a promising approach that is potentially easier to understand and apply. A case-based approach called ESTOR developed for software effort estimation [18] was found to perform significantly better than COCOMO and function point analysis on restricted samples of problems. Witting and Finnie [29,30] tested a neural network model on data gathered from commercial 4GL software development projects, across a large range of sizes. As is typical of software developments, the range of productivity and other development factors in their dataset was also large, accentuating the estimation problem [14]. Despite these difficulties, their neural network model predictions were reasonably accurate in comparison with other published results, indicating the potential of the use of this approach. Another study by Samson et al. [27], which used a Cerebellar Model Arithmetic Computer (CMAC) Albus multidimensional perceptron to predict software effort, compared linear regression with a neural networks approach using the COCOMO dataset. With MMRE of 520,7% and 428,1%, respectively, both approaches appeared to perform badly. An attempt to accommodate outliers made the regression less successful in predicting for the other observations (CMAC, not being confined to a linear function, can deal more successfully with observations that lie far from the best straight line). Srinivasan and Fisher [28] also reported the use of a neural network with a back propagation learning algorithm. This work compared the CartX and backpropagation learning methods to traditional approaches to software effort estimation. They found that these learning approaches were competitive with SLIM [25], COCOMO, and function points [16].

7. Concluding remarks

The estimation of development schedules and effort remain a complex problem that is attracting considerable research attention. One of the disadvantages of the ML algorithm is the unclear and closed structure of the computation process. Obviously, it is difficult for the project manager who is not a specialist in complex mathematical algorithms to understand the common sense behind such methods. Moreover, the credibility of an estimate depends entirely on the transparency of the method, data, definitions, and assumptions that were used to derive it.

In order to estimate development effort at an early stage in the development life cycle, it is necessary to identify the attributes of the project that are predictive of the effort involved. Product complexity is one of the model's inputs that reflect these attributes. However, the product features in the early stages of the project are not the same as in the later stages, especially for development projects, and poorly defined requirements and additional constraints can cause changes in the complexity of a given product over time. A similar problem exists with regard to size: product size in the later stages of a project is a more stable metric than in the earlier stages. Thus, size and complexity in the early stages of a project are also estimated values and can generate additional noise in the prediction model.

Though the collection of data from previously completed local projects is necessary for successful estimation [15], the past productivity coefficient is not an actual measure of risk but rather an indication of an organization's potential productivity based on its past experiences with "similar" development projects. Clearly, the technology, tools and other features of the project currently being estimated may be far different from any projects actually executed in the past by the organization, and the effects on the estimation model may be unpredictable. Thus, one limitation of the current research is the assumption that historical projects had comparable characteristics and that this assumption can be applied to all projects in the databases. Failing to recognize this limitation can lead to an additional noise effect. Another limitation is that for comparison purposes we used a narrow class of diverse structures of the neural network models discussed in the literature. Although this approach has been used in similar studies, it can be seen as limiting the estimation model. Finally, as claimed by [9], a distinction should be made between large and small projects (PC, mid-range and mainframe environments). This research did not assume a different relationship for effort and duration on this matter.

The greatest challenge for future research is in determining an optimal neural network structure for a given class of tasks, such as project attribute prediction. As with all non-linear estimation methods, it is difficult to find the global minimum of the error function. For example, an estimation problem with 35 weights could yield several quintillion local minima, some of which could produce very accurate forecasts if they are reasonably close to the global minimum. An effective procedure based on trial and error should therefore be developed. Adding more project metrics to improve results can be seen as one additional direction for future research, while team maturity and technological complexity of the project are also possible candidates. Finally, it is possible to explore alternative neural network models (e.g., the cascade correlation model [16]) to dynamically construct neural network architecture.

References

[1] A Guide to the Project Management Body of Knowledge, PMBOK Guide, 2004 ed.
[2] A.J. Albrect, J.E. Gaffney Jr., Software function, source lines of code and development effort prediction: a software science validation, IEEE Transactions on Software Engineering 24 (1978) 345–361.
[3] A. James Anderson, An Introduction to Neural Networks, Massachusetts Institute of Technology, MA, 1995.
[4] J. Axelsson, Cost Models for Electronic Architecture Trade Studies, Volvo Technological Development Corp., 1999.
[5] D. Baccarini, The concept of project complexity: a review, ISBG Journal of Project Management 14 (1996) 201–204.
[6] B.W. Boehm et al., Software Cost Estimation with COCOMO II, Prentice-Hall, 2000.
[7] B. Boetticher, Applying machine learners to GUI specifications in formulating early life cycle project estimations, in: T.M. Khoshgoftaar (Ed.), Software Engineering with Computational Intelligence, Kluwer Academic Publishers, 2003.
[8] G. Boetticher, An assessment of metric contribution in the construction of a neural network-based effort estimator, in: Second Int. Workshop on Soft Computing Applied to Soft. Engineering, 2001. Available from http://nas.cl.uh.edu/boetticher/publications.html.
[9] P. Bourque, S. Oligny, A. Abran, B. Fourrnier, Developing project duration models in software engineering, Journal of Computer Science and Technology 22 (2007) 348–357.
[10] J. Callahan, B. Moretton, Reducing software product development time, International Journal of Project Management 19 (2001) 59–70.
[11] Charted Institute of Building (CIOB), Procurement and project performance. Occasional Paper N45, CIOB, Ascot, England, 1980.
[12] COSMIC-FFP Measurement Manual, Common Software Measurement International Consortium, 1999.
[13] S. Delany et al., The limit of CBR in software project estimation, German Workshop on Case-based Reasoning, 1998.
[14] I.F. De Barcelos Tronto, J.D.S. da Silva, N. Sant'Anna, Comparison of artificial neural network and regression models in software effort estimation, in: Proceedings of International Joint Conference on Neural Networks, Orlando, FL, USA, August 12–17, 2007.
[15] S.E. Fahlman, The recurrent cascade-correlation architecture, Technical Report CMU-CS-91-100, School of Computer Science, Carnegie Mellon University, 1991.
[16] D. Garmus, D. Herron, Function Point Analysis: Measurement Practice for Successful Software Projects, Addison-Wesley Information Technology Series, First printing, November 2000.
[17] S. Gonzalez, Neural networks for macroeconomic forecasting: a complementary approach to linear regression models, Finance Canada Working Paper 2000-07, 2000.
[18] A. Griffin, The effect of project and process characteristics on product development time, Journal of Marketing Research 34 (1997) 24–35.

[19] F.G. Heemstra, Software cost estimation, Information and Software Technology 34 (10) (1992) 627–639.
[20] K. Hornik, M.B. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[21] ISBSG, International Software Benchmarking Standards Group, Data Repository, Release 9, 2004.
[22] C. Jones, Applied Software Measurement Assuring Productivity and Quality, McGraw-Hill, New York, 1996.
[23] E.W. Larson, D.H. Gobeli, Significance of project management structure on development success, IEEE Transactions on Engineering Management 36 (1989) 119–125.
[24] M.H. Meyer, J.M. Utteback, Product development cycle time and commercial success, IEEE Transactions on Engineering Management 42 (1995) 297–304.
[25] L.H. Putnam, A general empirical solution to the macro software sizing and estimating problem, IEEE Transactions on Software Engineering 24 (1978) 345–361.
[26] F. Mosteller, J.W. Tukey, Data Analysis and Regression: A Second Course in Statistics, Addison-Wesley, Reading, MA, 1997.
[27] B. Samson, D. Ellison, P. Dugard, Software cost estimation using Albus Perceptron, Information and Software Technology 39 (1997) 55–60.
[28] K. Srinivasan, F. Fisher, Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering 21 (2) (1995) 126–137.
[29] W. Witting, G. Finnie, Using artificial neural networks and function points to estimate 4GL software development effort, Journal of Information Systems 1 (2) (1994) 87–94.
[30] W. Witting, G. Finnie, Estimating software development effort with connectionist models, Information and Software Technology 39 (1997) 369–476.
[31] S. Vicinanza, M.J. Prietula, T. Mukhopadhyay, Case-based reasoning in software effort estimation, in: Proc. 1st Int. Conference Information Systems, 1990, pp. 149–158.