International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:11, 2008

A Metric-Set and Model Suggestion for Better Software Project Cost Estimation

Murat Ayyıldız, Oya Kalıpsız, and Sırma Yavuz

***Abstract*—**Software project effort estimation is frequently seen as complex and expensive for individual software engineers. Software production is in a crisis. It suffers from excessive costs. Software production is often out of control. It has been suggested that software production is out of control because we do not measure. You cannot control what you cannot measure. During last decade, a number of researches on cost estimation have been conducted. The metric-set selection has a vital role in software cost estimation studies; its importance has been ignored especially in neural network based studies. In this study we have explored the reasons of those disappointing results and implemented different neural network models using augmented new metrics. The results obtained are compared with previous studies using traditional metrics. To be able to make comparisons, two types of data have been used. The first part of the data is taken from the Constructive Cost Model (COCOMO'81) which is commonly used in previous studies and the second part is collected according to new metrics in a leading international company in Turkey. The accuracy of the selected metrics and the data samples are verified using statistical techniques. The model presented here is based on Multi-Layer Perceptron (MLP). Another difficulty associated with the cost estimation studies is the fact that the data collection requires time and care. To make a more thorough use of the samples collected, k-fold, cross validation method is also implemented. It is concluded that, as long as an accurate and quantifiable set of metrics are defined and measured correctly, neural networks can be applied in software cost estimation studies with success

***Keywords*—**Software Metrics, Software Cost Estimation, Neural Network.

I. Introduction

SOFTWARE becomes increasingly expensive to develop and is a major cost factor in any information system budget. Software development costs often get out of control due to lack of measurement and estimation methodologies.

Software cost estimation or software effort estimation is the process of predicting the effort required to develop a software system. Software engineering cost models and estimation techniques are used for a number of purposes including; budgeting, tradeoff and risk analysis, project planning and control, and software improvement investment analysis [1].

Manuscript received August 31, 2006. A Metric-Set and Model Suggestion for Better Software Project Cost Estimation

M. Ayyıldız is with the Computer Engineering Department,Yıldız Technical University, Istanbul, Turkey (e-mail: f0100301@ yildiz..edu.tr)

O. Kalıpsız is with the Computer Engineering Department,Yıldız Technical University, Istanbul, Turkey (e-mail: kalipsiz@ ce.yildiz..edu.tr).

S. Yavuz is with the Computer Engineering Department,Yıldız Technical University, Istanbul, Turkey (e-mail: sirma@ce.yildiz.edu.tr).

The accuracy of the software project cost estimation has a direct and significant impact on the quality of the firm's software investment decisions [2]. Accurate cost estimation can reduce the unnecessary costs and increase the organization's efficiency. For this reason, many estimation models have been proposed over the last 20 years. The review completed by Jørgensen and Shepperd [3] identifies 304 software cost estimation papers in 76 journals and classifies the papers according to research topic, estimation approach, research approach, study context and data set. Although there are number of different approaches, these models may be classified as algorithmic and non-algorithmic. Each of these techniques has their advantages as well as limitations. Unfortunately, despite the large body of experience with estimation models, the accuracy of these models is still far from being satisfactory [4]. Software development effort estimation with the aid of artificial neural networks (ANN) attracted considerable research interest especially at the beginning of the nineties [5]. Most of these studies are based on COCOMO'81 metric-set.

A key factor in selecting a cost estimation model is the accuracy of its metrics, since these models rely on metrics as their input. Metric can be defined as a quantitative measure of the degree to which a system, component, or process possesses a given attribute. It may seem easy to think of attributes of computer software products, processes, people or programming environments that can be measured. However, identifying meaningful attributes to measure and then finding measurement processes to produce reliable and reproducible assessments of these attributes is the real problem [6].

As the first step of our study, an augmented metric set was developed to take new technologies and more meaningful parameters into account. This augmented model, called Yıldız effort estimation model (YEEM), does include all the metrics existed in COCOMO and new ones which are derived from more recent software development projects, studies and experience.

In this paper, we present an augmented metric set and investigate the success of the MLP for both COCOMO'81 metric set and for the augmented metric set (YEEM). Tests were run for datasets containing the same number of samples. To investigate further, we have also experimented with larger datasets formed according to augmented metrics. Addressing the issues of the dataset characteristics and the amount of samples in the datasets is one of the purposes of this research. Since the amount of samples we have collected are still limited, 5-fold, 10 fold and 15-fold cross validation

techniques have been also applied.

II. Related Work

Most of the software cost estimation studies using of neural networks have focused on the accuracy of the approach when compared to algorithmic models and paid little attention on the suitability of the metrics [7]. Wittig and Finnie [8] employed a back-propagation multi-layer perceptron to predict the development effort on the Desharnias and Australian Software Metrics Association (ASMA) data sets. Both data sets were tested three times by randomly removing ten projects to be used as the test set and using the remaining projects as the training set. The results they got were very encouraging, with an overall MMRE of 29% for the Desharnais data set and 17% for the ASMA data set. Venkatachalam [9] has also chosen a back-propagation multi-layer perceptron to predict software effort and development time. The preliminary results, obtained with the data taken from the COCOMO database, were reported as promising. Jørgensen [10] reports the use of a multi-layer perceptron with a back propagation algorithm on a data set comprising 109 maintenance projects and compares four different approaches; regression models, a neural network model, a form of pattern recognition and a simple baseline rule of thumb model. In this study, the neural network model was found to perform less well than the best regression model, in terms of MMRE, but very favorably in terms of Pred (25). On the negative side, he found the neural network to be one of the least robust approaches. Serluca [11] obtains much better results by using a back-propagation network on the MERMAID-2 data set. Serluca concluded that neural networks require large training sets.

III. Research Method

In this study a new metric set (YEEM), which is initially derived from COCOMO ’81, COCOMO II Post Architecture metric-set, 21 real world software development projects, studies and more than 90 years of project management experience, is proposed to be used in software cost estimation models. While building the YEEM, metric suggestions taken from 28 Project managers have been added to preliminary metric matrix. The correlations have been analyzed and some of the metrics were removed accordingly. After a mathematical, statistical and empirical optimization processes, the resulted metric set can be categorized into six groups; product, resource, risk, technology, environment, plans and predictions. The YEEM augmented metric set consists of 56 sub-metrics, which are in a hierarchical structure.

The experimental data is collected in a leading international company in Turkey, which gives additional importance to project management discipline. Developing standardized project plans and complying with software development standards are among the main characteristics of the company. Configuration management rules are strictly defined and followed within the company. The project teams are in matrix form; staff from different departments and from different positions form the project group. In performance management, the success of the project has a direct effect on individual success.

A. YEEM Metric-Set

1) Metric-Set Definition

In recent literature, a few measures have been proposed for capturing properties of software artifacts in a quantitative way. However, few of these 'software metrics' have successfully survived the initial definition phase and are actually used in industry. This is due to a number of problems related to the theoretical and empirical validity of software metrics, the most relevant of which are summarized below [12]

1. Measures are not always defined in the context of some explicit well-defined measurement goal of the industrial interest they help to reach.

2. Even if the goal is explicit, the experimental hypotheses are often not made explicit.

3.A reasonable theoretical validation of the measure is often not possible because the attribute that a measure aims to quantify is often not well defined.

4.Most of measures have never been subject to an empirical validation.

5. Measurement definitions do not always take into account the environment or context in which they will be applied.

This situation has frequently led to some degree of fuzziness in measure definitions, properties, and underlying assumptions, making the use of the measures difficult, their interpretation hazardous, and the results of the various validation studies somewhat contradictory. These characteristics do not imply that progress cannot be made in the measurement field. For this purpose metrics must be defined in a methodological and disciplined way. The main tasks of this method are shown in Fig. 1.

We tried to construct a metric-set which is based on clear measurement goals. GQM is used as a goal-oriented measure definition approach to ensure that the metric selection or definition effort, the validation effort and the subsequent measurement effort all contribute to the achievement of a well-defined goal.

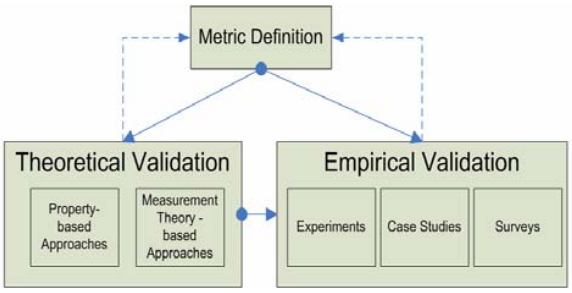

Fig. 1 Method for definition and validation of software metrics

2) The Metric Definition Methodology

Basili and Weiss data collection methodology has been used to construct the YEEM metric set. The methodology

consists of six steps with considerable feedback and iteration occurring at several places [14]:

a. *We established the goals of the data collection about software project cost indicator.*
b. *We developed a list of questions of interest.*
c. *We established metric-set categories.*
d. *We designed and tested collection forms.*
e. *We collected and validated data.*
f. *We analyzed the data.*

The metric definition activity further involves the precise definition of the object of measurement, including its constituents. The unambiguous definition of the domain of a metric is a prerequisite for the validation of the metric, as well as for the practical use of the metric afterwards. Finally, as metrics are functions that take an argument and produce a value, there must be a precise description of the mathematical, logical or other formulation and notation that is used to define and denote the function prescriptions.

The purpose of the definition phase is to define the goals of an experiment, formulated from the problem to be solved. In order to capture the goals of the experiment a template for goal definition based on the famous *Goal-Question-Metric* (GQM) paradigm. This goal template is: *Analyze <Object(s) of study> for the purpose of <Purpose> with respect to their <Quality focus> from the point of view of the <Perspective> in the context of <Context>* [14]

The object in this study is "software project development". The Objects of the study the entity that is studied in the experiment The Purpose defines what the intention of the experiment is. The Perspective tells us the viewpoint from which the experiment results are interpreted. The general perspective we take is that of professionals working with Software Project Development. The Context is the "environment" in which the experiment is run. All the experiments we have done were run in an industrial environment.

3) Theoretical validation

The main goal of theoretical validation is to assess whether a metric actually measures what it purports to measure [14]. In the context of an empirical study, the theoretical validation of metrics establishes their construct validity, i.e. it 'proves' that they are valid measures for the constructs that are used as variables in the study.

The validity of the measurement instruments used for the variables of an empirical study is a key factor in the overall study validity. On the other hand, knowledge of the scale type of the metrics helps when choosing the appropriate statistical techniques to analyze the data obtained in the experiments. The preferred theoretical validation approach is one in which measures can be used as ratio scales, allowing the use of a wide range of data analysis techniques, including parametric statistics, and thus providing maximum flexibility to the researcher [14]. Unfortunately, as Van den Berg and Van den Broek [15] remark, even though several attempts have been made at proposing methods and principles to carry out the theoretical validation of metrics, there is not yet a standard, accepted way of theoretically validating a software metric. Work on theoretical validation has followed two paths:

a. Property-based approaches (Axiomatic approaches) [12,16]
b. Measurement-theory based approaches [17,18, 19]

I*n this work the theoretical validation of YEEM metric-set will be performed using the distance framework.* This framework, proposed by Poels and Dedene [19], is a conceptual framework for software metric validation grounded in Measurement Theory [20].

The distance framework offers a measure construction procedure to model properties of software artifacts and defines the corresponding software metrics. In this sense, the framework has an added value above other measurement theoretic approaches that focus on metric validation. *The basic idea of distance framework is to define properties of objects in terms of distances between the objects and other objects that serve as reference points for measurement* [19].

4) The Distance measure construction procedure

The measure construction procedure prescribes five activities. The activities of the distance procedure are very briefly summarized below [17].

1. *Finding a measurement abstraction*
2. *Defining distances between measurement abstractions*
3. *Quantifying distances between measurement abstractions*
4. *Finding a reference abstraction*
5. *Defining a measure for the property*

5) Empirical validation

Empirical validation is crucial for the success of any software measurement project. Through empirical validation we can demonstrate with real evidence that the measures we have proposed serve the purpose they were defined for and that they are useful in practice, i.e. related to some external attribute worth studying and therefore helping to reach some goal. In our case we want to demonstrate the empirical validity of the YEEM metric-set, this means that we have to corroborate that they are really related to software project effort estimation.

We use three major types of empirical research strategy to get all related metrics and their importance.

i.Surveys. A survey is often an investigation performed in retrospect. The primary means of gathering qualitative or quantitative data are interviews or questionnaires. The results from the survey are then analyzed to derive descriptive or explanatory conclusions. We made interviews and questionnaire to get all related metrics and their importance.

ii.Experiments. Experiments are formal, rigorous and controlled investigations..

iii.Case studies. A case study is an observational study. Our case studies are aimed at tracking a specific attribute or establishing relationships between different attributes.

6) YEEM metric-set suggestion

Cocomo 81 and Cocomo II metric-sets has been mostly used in Software cost prediction studies. Therefore YEEM metric-set is based on Cocomo metric sets. 28 Project managers' metric suggestions added to preliminary metric matrix. Then, the correlations have been captured and the related metrics was removed. After a mathematical, statistical and empirical optimization processes, the resulted metric set can be grouped in 6 parts which are: Product, Resource, Risk, Technology, Environment, Plans & Predictions

All of the metrics in Cocomo metric-set has been used in YEEM metric-set. However there is a hierarchical structure. The resulted 6 parts has 26 metrics which are shown in Table I. The main aim of this paper is suggesting a metric-set which provide us to get better results from software development prediction studies and models. This paper suggests that using hybrid model provide us better results. So using function point and predictions as input metrics makes a hybrid model. Some metrics which are not used in Cocomo metric-set are also added the suggested metric set.

TABLE I
YEEM METRIC-SET

Category	Metric
A. Product	1.Complexity
	2.Function Point
	3.Importance of the final product
	4.Budget Size
	5.Expected Features' Level
B. Resource	6.Qualifiedness of Project's Members
	7.Personnel Continuity (PCON)
	8.Team-Size
	9.Hardware situation
	10.Use of Software Tools (TOOL)
C. Risk	11.Budget (Change) Risk
	12.Human-Resurce's Risk
	13.Hardware Risk
	14.Product definition and scope change risk
D. Technology	15.Usage Easiness of Software Development Tools
	16.Usage know-how of Software Development Tools
F.Environment	17.General Features
	18.Responsibility
	19. Pressure
	20.Time Management
	21.Average Productivity
F. Plans and Prediction	22. Predicted time plan
	23.Predicted budget size
	24. Prediction Average Error
	25. Toleration
	26.Required Development Schedule (SCED)

a. Product

One of the main categories of the suggested set is product size. It has 5 metrics: complexity, FP, importance of final product, budget size and expected features' levels.

It is very important to find the required resources by estimating size of the product. The most of the project managers said that complexity of the product (CPLX), size of the database (DATA) and reusability (RUSE) are most important factors. CPLX, DATA and RUSE are also used in COCOMO model. These are sub-metrics of the complexity metric.

Function points are a measure of the size of computer applications and the projects that build them. It is important to stress what function points do not measure. Function points are not a perfect measure of effort to develop an application or of its business value, although the size in function points is typically an important factor in measuring each Using the function point as an input metric has two advantages. It gives us product size and it gives us product's appearance size.

Project budget is one of the most important components of a software development project. So if we can get the size of the budget, it will give us roughly size and the importance level. Importance of the final product is a very important metric to get the total pressure, capability of the project's member and managerial responsibility and support. The expected features can be flexibility, reliability (RELY), user friendship, maintainability, security and documentation (DOCU).

b. Resource

Resource size and the resource qualifiedness are the main metric in the most of the metric sets. If you want to calculate the cost of doing something, you have to know two things: size of the product and the resource.

Qualifiedness of project's members is one of the resources metric. It has 3 or more sub-metrics which are qualifiedness level of project manager, qualifiedness level of software developers (ACAP) and qualifiedness level of system and business analysts (PCAP).Team-size is a resource metric. It has 3 or more sub-metrics which are number of project manager, developer, analyst and tester. We have to multiply these sub-metrics to availability to get the pure human resource. We can use personnel continuity (PCON) metric.

Hardware situation is another resource metric which have three sub-metrics. These are time constraint (TIME), storage constraint (STOR) and platform volatility (PVOL).

c. Risk

The capability to effectively deliver software on time and within budget is based upon a variety of risk factors.

As each project commences, the size, complexity, and various risk factors are assessed, and an estimate is derived. Initially, the resulting estimate would typically be based upon industry data that reflect average occurrences of behavior given a project's size, complexity, and performance profile. Over time, as an organization develops a historical baseline of

information regarding its own behaviors, performance profiles would reflect a more accurate representation of likely outcomes. This information can be used to predict and explore what-if scenarios on future

Risk exposure, or risk impact, is quantified as: risk exposure = probability of undesirable outcome x cost of undesirable outcome. So if we can get the probability of undesirable outcome and the cost of undesirable outcome, we can calculate the risk exposure. Most projects have budget risk, human-resource's risk, hardware risk and definition and scope change risk. These four metrics has two sub-metrics: probility and effect.

d. Technology

The software tools and tool experience has very significant effect which incredibly change the result. So we have to mention it to get better result. The structure of the establishment can affect the cost of the software development. The averages of lateness, pressure, responsibility and time management can greatly affect the cost. Usage easiness and know-how of software development tools are the two metric in technology category. Usage know-how has 4 sub-metrics: platform experience (PEXP), application experience (AEXP), language and tool experience (LTEX), multi-site development (SITE).

e. Environment

General Features metric can include 4 sub-metrics; Average delayed-project percentage in the enterprise, Average lateness in the enterprise, Average lateness percentage in the enterprise, and Standard deviation of lateness in the enterprise.

Responsibility metric can include 4 or more sub-metrics which are responsibility level of Project Manager, management, team members and the customer.

Pressure metric can include at least 3 sub-metrics, which are managerial pressure, customer pressure, marketing pressure.

And the time management can include 4 sub-metrics: (i) Average interruption number in one day for a member, (ii) Average interruption-duration time, (iii) Average returning to normal state time after interruption, (iv) Average overtime

f. Plans and Prediction

Expert judgment is still the dominant technique in practice today for estimation of software project size and effort. Using expert judgment result as an input metric is suggested. Thus, it will increase the accuracy of the result.

B. Neural Network Model

1) Algorithm

The model used in this study is a multi-layer feed-forward network that is used with the back propagation algorithm. Back-propagation involves performing computations backwards through the network to determine the gradient of the cost function. Then the weights are adjusted in the direction of the negative gradient. The mechanism by which weights are updated is known as training algorithm. The selected training algorithm is the Levenberg-Marquardt [18].

COCOMO '81 and YEEM metrics are used in turn, as network inputs to predict the software cost.

2) Data Processing

a. Checking for Randomness and Outliers

In this study, the training and test data sets have been verified for randomness, using the 'Run' test [21].

Three outliers were identified in COCOMO '81 data set, and consequently only remaining 60 samples are used in this study. Similarly 9 out of the 109 samples, collected according to proposed metric set, are left out since they represented bigger projects.

b. Pre-processing and Post-processing of Data

In this study, min-max normalization [22] is used, to squash the data values into the intervals [0, 1].

c. Organization of the Data Sets

Organization of the data sets used in experiments, and the amount of data points in each set are given in Table II. They are subdivided into training and test sets to be used to test the neural network models. Dataset 2, containing the same amount of data with Dataset 1, is formed to compare the results of new metric set with COCOMO. And Dataset 3 was formed to asses the effect of data size on the success of models.

TABLE II
ORGANIZATION OF THE DATA SETS

	Data Set 1	Data Set 2	Data Set 3
Metric Set	COCOMO	YEEM	YEEM
Total Number of Data	60	60	100
Number of Training Data	45	45	75
Number of Test Data	15	15	25

IV. RESULTS

A. Evaluation Criteria

Magnitude of Relative Error (MRE) is a common criterion for the evaluation of similar cost estimation models in literature [23, 24]. The MRE value is calculated for each observation *i*, whose cost is predicted. The aggregation of MRE over whole test set can be achieved through Mean MRE (MMRE).

Although MMRE may be sensitive to individual predictions with excessively large MREs, careful selection of data, by checking the data against outliers or randomness, should provide sound results. It is always beneficial to check the standard deviation of the results as well. The results for each data set, are given in Table III.

TABLE III
RESULTS FOR COCOMO AND YEEM METRIC SETS USING MLP

ANN Model	Data Set	Total Number of Samples	MMRE
MLP	COCOMO	60	1.805
MLP	YEEM	60	0.280
MLP	YEEM	100	0.207

B. Cross Validation

Error estimation techniques, such as the v-fold cross-validation, can help in estimating the generalization performance as well as in selecting good parameters.

The v-fold cross-validation makes a more thorough use of the samples [14]. MLP neural network is used to implement the 5-fold, 10-fold and 15-fold cross validation methods. And the WEKA data-mining tool was used for these experiments [25]. Table 4 shows the results for the 5-fold, 10-fold and 15-fold cross validation methods using MLP.

TABLE IV
RESULTS OF CROSS VALIDATION METHOD USING MLP

	Data Set 1	Data Set 2	Data Set 3
Metric Set	COCOMO	YEEM	YEEM
Total Number of Data	60	60	100
MMRE 5-fold cv	1.898	0.216	0.128
MMRE for 10-fold cv	1.890	0.222	0.120
MMRE for 15-fold cv	1.664	0.189	0.095

V. CONCLUSION

In this paper we have proposed a new augmented metric set for software cost estimation and investigated the applicability of neural networks into software cost estimation studies.

From the results obtained, it can be concluded that the controversial results existing in literature are mostly due the selection of metrics. Considerable part of those studies uses historical COCOMO '81 data and reports very poor results. During the experiments we have also observed that the results obtained by using COCOMO'81 data were very inconsistent and poor. Despite experimenting with different neural network architectures and training algorithms it was not possible to draw any meaningful conclusion.

On the other hand when we used the augmented metric set (YEEM), which is constructed by identifying meaningful and up to date attributes, the results were consistent and reproducible. The new metric set and the data collected accordingly are tested with MLPs. 15-fold cross-validation result seems to be the most reliable one among the three cross-validation settings we have tried.

Since we have experimented with different size of data sets, it was also possible to see that the amount of samples has an effect on the success of the models.

REFERENCES

[1] Devnani-Chulani, S.: Bayesian Analysis of Software Cost and Quality models. University of Southern California, Doctor of philosophy Thesis. (1999)
[2] Al-Sakran, H.: Software Cost Estimation Model Based on Integration of Multiagent and Case-Based Reasoning. Journal of Computer Science Volume 2(3) (2006) 276-282
[3] Jørgensen M. and Shepperd M.: A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering. (2006)
[4] Leung, H., Fan, Z.: In Handbook of Software Engineering and Knowledge Engineering (Ed,Chang, S. K.). Volume 2 World Scientific. (2002)
[5] Han, J. and Kamber, M.: Data mining concepts and techniques. Academic Press. San Francisco. (2001)
[6] Kan, S.H.: Metrics and Models in Software Quality Engineering. Adisson Wesley. (2002)
[7] Hughes, R.T.: An Evaluation of Machine Learning Techniques for Software Effort Estimation. University of Brighton. (1996)
[8] Wittig, G., Finnie, G.: Estimating Software Development Effort with Connectionist Models. Information and Software Technology. Volume 39 (1997) 469-476
[9] Venkatachalam, A.R.: Software Cost Estimation Using Artificial Neural Networks. International Joint Conference on Neural Networks. Nagoya. (1993)
[10] Jørgensen, M.: Experience with the Accuracy of Software Main Task Effort Prediction Models. IEEE Transactions on Software Engineering, Volume 21(8) (1995) 674-681
[11] Serluca, C.: An Investigation Into Software Effort Estimation using a Back-Propogation Neural Network. M.Sc. Thesis. Bournemouth University. (1995)
[12] Briand L., Morasca S. and Basili V. (2002). An Operational process for goal-driven definition of measures. IEEE Transactions on Software Engineering, 30(2), 120-140.
[13] Zuse H. (1998) A Framework of Software Measurement. Walter de Gruyter Berlin.
[14] Fenton N. and Pfleeger S. (1997). Software Metrics: A Rigorous Approach. 2nd. edition. London. Chapman & Hall.
[15] Van Den Berg and Van Den Broek. (1996). Axiomatic Validation in the Software Metric Development Process. In Chapter 10: Software Measurement, Edited by Austin Melton, Thomson Computer Press.
[16] Weyuker E.J. (1988). Evaluating Software Complexity Measures. IEEE Transactions on Software Engineering. 14(9). 1357-1365.
[17] Whitmire S. (1997). Object Oriented Design Measurement. John Wiley & Sons. Inc.
[18] Reed, R. D. and Marks, R. J.: Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks. MIT Press. (1999)
[19] Poels G. and Dedene G. (2000). Distance-based software measurement: necessary and sufficient properties for software measures. Information and Software Technology. 42(1). 35-46.
[20] Krantz D., Luce R.D., Suppes P. and Tversky A. (1971). Foundations of Measurement. Vol. 1. Academic Press. New York.
[21] Knuth E. D.: The art of computer programming. 2nd ed. Addison-Wesley. (1981)
[22] Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann. (1999)
[23] Briand, L., El Emam, K., Surmann, D., Wieczorek, I., and Maxwell, K.: An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. In Proceedings of the International Conference on Software Engineering. (1999) 313-322
[24] Idri A., Abran A., Khoshgoftaar T.: Fuzzy Case-Based Reasoning Models for Software Cost Estimation. Soft Computing in Software Engineering: Theory and Applications. Springer-Verlag. (2003)
[25] Witten, I. H. and Frank, E.: Data Mining: Practical machine learning tools and techniques. 2nd Ed. Morgan Kaufmann. San Francisco. (2005)