Adequacy Checking of Personal Software Development Effort Estimation Models Based upon Fuzzy Logic: A Replicated Experiment

Comprobación de la Adecuación de Modelos de Estimación del Esfuerzo de Desarrollo de Software Personal Basados en Lógica Difusa: Un Experimento Replicado

Cuauhtémoc López Martín[1], Cornelio Yáñez Márquez[2], Agustín Gutiérrez Tornés[3] and Edgardo Felipe Riverón[4]
[1,2,4] Center for Computing Research, National Polytechnic Institute; P.O. 07738, Mexico, D.F.
[3] Systems Coordinator, Banamex, Mexico, D.F.; ITESM, Mexico, D.F.
cuauhtemoc@sagitario.cic.ipn.mx, cyanez@cic.ipn.mx,
agustin.tornes@itesm.mx, edgardo@cic.ipn.mx

Article received on March 09, 2006; accepted on July 10, 2007

Abstract
There are two main stages for using an estimation model (1) it must be determined whether the model is adequate to describe the observed (actual) data, that is, the model adequacy checking or verification; if it resulted adequate then (2) the estimation model is validated in its environment using new data. This paper is related to the first step. An investigation aimed to compare personal Fuzzy Logic Systems (FLS) with linear regression is presented. These FLS are derived from a replicated experiment using a sample integrated by ten developers. This experiment is based on both a common process and inside of a controlled environment. In six of ten cases the multiple range tests for Magnitude of Relative Error (MRE) by technique show that fuzzy logic is slightly better than linear regression. These results show that a FLS could be use as an alternative for the software development effort estimation at personal level.
Keywords: Software development effort estimation, Fuzzy logic, Linear Regression, Magnitude of Relative Error

Resumen
Existen dos fases principales en el uso de un modelo de estimación: (1) se debe determinar si el modelo es adecuado para describir los datos observados (reales), eso es, la comprobación de la adecuación del modelo o verificación del mismo; si éste resultara adecuado, entonces (2) el modelo de estimación se valida en su ambiente usando datos nuevos. Este artículo está relacionado con la primera etapa. Se presenta una investigación dirigida a la comparación de Sistemas de Lógica Difusa (SLD) personales. Estos SLD se derivan a partir de un experimento replicado con base en una muestra de diez desarrolladores, así como en un proceso de desarrollo común dentro de un entorno controlado. En seis de los diez casos, las pruebas de rango múltiple de la Magnitud del Error Relativo (MER) por técnica, muestran que la lógica difusa es ligeramente mejor que la regresión simple. Estos resultados muestran que un SLD podría ser utilizado como alternativa para la estimación del esfuerzo de desarrollo de software a nivel personal.
Palabras clave: Estimación del esfuerzo de desarrollo de software, Lógica difusa, Regresión lineal, Magnitud del error relativo

1 Introduction

Accurate and timely prediction of the development effort and schedule required to build and/or maintain a software system is one of the most critical activities in managing software projects (Idri et al., 2003; Jorgensen et al., 2000). In addition, software estimation has been identified as one of the three great challenges for half-century-old computer science (Brooks, 2003).

Software engineering estimation techniques can be used for a number of purposes inside of enterprises. These include (a) budgeting (b) trade-off and risk analysis and (c) project planning and control, and (d) software improvement investment analysis (Boehm et al., 1998). The consequences of effort overruns are, among others (a) lack of quality of the deliveries, (b) dissatisfied customers, and (c) frustrated developers (Jorgensen et al., 2000).

In accordance with the Mexican National Program for Software Industry Development (MNPSID), the 90% of software Mexican enterprises do not have formal processes to record, track and control measurable issues during the development process (Secretaria de Economia, 2002). This statistic implicitly means that in those enterprises the need for practicing software development effort estimation exists.

The MNPSID measures its goals of year 2010 based upon levels of the Capability Maturity Model (CMM). The CMM is an available description of the goals, methods, and practices needed in software engineering industrial practice. Twelve of the eighteen key process areas of the CMM are at least partially addressed by the Personal Software Process (PSP) (Humphrey, 1995). The measures recorded by the PSP are program size, effort and defects.

Unless engineers have the capabilities provided by personal training, they cannot properly support their teams or consistently and reliably produce quality products (Humphrey, 2000). It suggests that the software estimation activity could start through a personal level approach by developing academic programs.

Several cost and effort estimation techniques have been proposed and researched over the last 30 years (Mendes et al., 2002; Briand et al., 2000). Researchers aimed to (1) determine which technique has the greatest effort prediction accuracy, and (2) propose new or combined techniques that could provide better estimates. This paper is related to target number 2.

These techniques fall into the following three general categories (Mendes et al., 2002):

1) Expert judgement: Is a technique widely used, that aims to derive estimates based upon a previous experience of expert on similar projects. The means for deriving an estimate are not explicit and therefore not repeatable.

2) Algorithmic models: It is today the most popular in the literature (Mendes et al., 2002). It attempts to represent the relationship between effort and one or more characteristics of a project. The main cost driver in such a model is usually some notion of software size (e.g. the number of lines of source code as in this paper). Algorithmic models need calibration to be adjusted to local circumstances (as done in this study). Their general form is a linear regression equation (Kok et al., 1990) or non-linear as those used in COCOMO 81 (Boehm, 1981) and COCOMO II (Boehm et al, 2000).

3) Machine learning**:** Machine learning techniques have recently been used as a complement or alternative to the previous two techniques. Fuzzy logic models are included in this category (Mendes et al., 2002) as well as neural networks (Idri et al., 2002a), genetic programming (Burguess and Lefley, 2001), regression trees (Srinivasan and Fisher 1995), and case-based reasoning (Kadoda et al, 2000) .

In this paper development effort estimations are compared, the same seven programs are developed by all programmers, and simple linear regression (used by algorithmic models), and fuzzy logic (a machine learning technique) are used like estimating techniques. Lines of code and development time (effort) are gathered from a sample of developers. The seven small programs are based on a same process suggested by Humphrey (Humphrey, 1995). The research of this paper (a) checks the adequacy of fuzzy logic estimation model for determining its usefulness so that it can then be used in its environment (model verification), (b) in accordance with an ANOVA of MRE, it has as hypotheses that a fuzzy logic system is equal or better than a linear regression model so that can the fuzzy model be used for estimating the software development effort of small programs, and (c) it has been based on a replicated study (same programs, process, standards, and sample criteria, all these issues applied by different developers).

Comparison Amongst Techniques

Experience has shown that there is not a best prediction technique outperforming all the others in every situation: some researches have found that estimation by analogy generated better results than stepwise regression, while other ones have reported opposite results (Idri et al., 2003). Hence, no one method or model should be preferred over all others. An alternative can be fuzzy logic from the computational intelligence.

Software Measurement

The most common application of software metrics is to develop models that predict the effort that will be required to complete certain stages of the software development (Gray and MacDonell, 1997).

In spite of the availability of a wide range of software product size measures, source lines of code (LOC) remains in favour of many models (MacDonell, 2003; Mendes et al., 2002). There are two measures of source code size: physical source lines and logical source statements (Park, 1992). This paper uses physical LOC for estimating the development effort.

Linear Least-Squares Regression and Correlation

The most commonly used methods for predictive model developments are those derived from inferential statistics based upon simple linear regression. Any form of linear regression is generally preceded by the use of scatter plots and correlation analyses in order to first intuitively, as well as quantitatively, determine the potential relationships that may exist in the data.

Values of correlation (r) close to -1 or 1 indicate a strong linear relationship between the variables; that is, when two sets of data are strongly related, it is possible to use a linear regression procedure to model this relationship. The linear regression equation using least squares can be expressed as follows:

$$E' = a + b\,(LOC)$$

In this study *LOC* represents the lines of code, *E* is the development effort, *a* is the y-value at which the straight-line intersects the y-axis, and *b* measures the steepness of straight line (Weiss, 1999).

Fuzzy Logic

Newer computation techniques on cost estimation that are non-algorithmic appeared in the 1990s. Fuzzy Logic with its offerings of a powerful linguistic representation can represent imprecision in inputs and outputs, while providing a more expert knowledge-based approach to model building (Ahmed et al., 2005).

One general disadvantage of statistical models is the manner in which their comprehensibility diminishes variables, interactions, and transformations are added. This problem can be at least partially overcome with the use of fuzzy logic, which was developed out of dissatisfaction with classical, all-or-nothing, logic. The central assertion underlying this approach is that entities in the real world simply do not fit into neat categories. For example, a software project is not small, medium, or large. It could in fact be something in between, perhaps mostly a large project but also something likes a medium project. This can be represented as a degree of belonging to a particular linguistic category (MacDonell and Gray, 1996**)**.

A fuzzy set is a set with a graded membership function, *m*, in the real interval [0, 1]. This definition extends the one of a classical set where the membership function is in the couple {0, 1}. Fuzzy sets can be effectively used to represent linguistic values such as low, young, and complex. The representation by a fuzzy set has next advantageous (1) it is more general, (2) it mimics the way in which the human- mind interprets linguistic values, and (3) the transition from one linguistic value to a contiguous linguistic value is gradual rather than abrupt (Idri et al., 2003).

All estimation techniques has an important limitation, which arises when software projects are described using categorical data (nominal or ordinal scale) such as small, medium, average, or high (linguistic values). A more comprehensive approach to deal with linguistic values is by using fuzzy set theory (Idri et al., 2002b). There are a number of ways through data fuzzification could potentially be applied to the effort estimation problem (Schofield, 2001). One of them is used in this study: to construct a rule induction system replacing the crisp facts with fuzzy inputs; an inference engine uses a base of rules to map inputs to a fuzzy output which can either be translated back to a crisp value or left as a fuzzy value.

Evaluation Criterion

A common criterion for the evaluation of estimation models is the Magnitude of Relative Error (MRE) (Briand et al., 1998) which is defined as follows:

$$\text{MRE}_i = \frac{|\,\text{Actual Effort}_i - \text{Predicted Effort}_i\,|}{\text{Actual Effort}_i}$$

The MRE value is calculated for each observation *i* whose effort is predicted. The aggregation of MRE over multiple observations (N), can be achieved through the Mean MRE (MMRE). In general, the accuracy of an estimation technique is inversely proportional to the MMRE:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i$$

Related Work

Papers were reviewed regarding aspects related to a replicated empirical research on software development effort estimation at a personal level based on fuzzy logic. Not any paper involving replication in its empirical research was found in the referenced papers (Ahmed et al., 2005), (Braz and Vergilio, 2004), (Gray and MacDonell, 1997), (Huang et al., 2004), (Idri et al., 2001), (Musflek et al., 2000), (Zhiwei and Khoshgoftaar, 2004). An additional paper involves replication (Briand et al., 2000), however, it does not use fuzzy logic for estimating the development effort. Moreover (López-Martín et al., 2005) considers fuzzy logic and practices of PSP, but it is not a replicated experiment. Likewise (Höst and Wohlin, 1997) is based on both a replicated experiment and at personal level, but it uses expert judgement for estimating.

2 Experimental Design

The study in this paper was carried out in a controlled environment thus, it involves control during a replicated and supervised experiment.

Empirical studies come in a wide variety of types, employing a variety of experimental designs. One of them is used in this paper: the replicated project study. Studies of this type employ multiple subjects, all working on the same project or application (Seaman, 1999).

This paper considers guidelines suggested in (Kitchenham et al., 2002) that involve the next six basic topic areas: experimental context, experimental design, conduction of the experiment and data collection, analysis, presentation of results, and interpretation of results.

An ANOVA for comparing in this experiment the results by developer is used. The dependent variable is the MRE of each program developed by programmers.

2.1 Population Being Studied

Developers with at least twelve months experience in developing software inside their enterprises as well as at least twelve months experience in their programming language represented the experiment population (a year in programming language experience is considered like *nominal* (Boehm, 1981)).

2.2 The Rationale and Technique for Sampling from that Population

Since the attributes must be relevant for the effort estimation, estimation researches used lines of code to correlate effort and attributes (Idri et al., 2001).

In this study, the whole population was integrated by 15 persons. From this group, ten developers were selected. The selection was based upon two criteria:

a) Correlation between program size (measured in physical lines of code)–effort (measured in minutes): those with results higher than 0.5 (this value is considered as moderated in (Lind et al., 2000).

b) Assumptions of residuals: (1) Independence, (2) Equal standard deviations and (3) Normality assumptions for MRE ANOVA must be met.

2.3 The Process for Allocating and Managing the Treatments

The process followed by developers was integrated with some practices of the PSP, which includes plan, development (design, code, compile, and test) and post-mortem phases (Humphrey, 1995). Each member of the population developed the same seven small programs. Seven was a number established because of availability of developers. Ten sessions were carried out, in the first one both coding and counting standards were made. From second one only one program was developed (one daily). Finally, the ninth and tenth days were assigned to make final reports. The seven small programs were the following:

1. Estimating the mean of a sample of *n* real numbers.
2. Estimating the standard deviation of a sample of *n* real numbers.
3. Calculating the sum of two matrixes composed by real numbers.
4. Calculating the sum of the diagonal of a matrix composed by real numbers.
5. Transforming the quantity in numbers to letters.
6. Calculating the correlation between two series of numbers.
7. Computing the linear regression equation parameters.

The characteristics of the fuzzy logic model are the following: a) type: mamdani, b) *and* method: *min*, c) *or* method: *max*; d) implication: *min*, e) aggregation: *max*, and f) defuzzyfication: *centroid*.

2.4 Methods used to Reduce Bias and to Determine the Sample Size

Every developer had at least the same months of development experience. A set of seven small programs was common to all of them. The development effort of these programs was measured in minutes. They followed the same development process. They were constantly supervised and advising about the process. Moreover, each developer selected his/her own programming language.

Since a coding standard should also establish a consistent set of coding practices as a provided criterion for judging the quality of the produced code (Humphrey, 1995), it is necessary to use always the same coding standard. With the following characteristics, the code standard by developer met: each compiler directive, variable declaration, constant definition, delimiter (Pascal: begin, end; C, JAVA: { , }); assign sentence (Pascal: :=; C, JAVA: =), flow control statement (Pascal words: if-then, else, case-of, while-do, repeat, until, for-to-do; C, JAVA words: if, switch, case, while, do, for) was written in a line of code.

In accordance with (Humphrey, 1995) Table 1 is filled depicting the counting standard followed by every developer.

Table 1. Counting standard

Count type	**Type**
Physical/logical	Physical
Statement type	**Included**
Executable	Yes
No executable	
Declarations	Yes, one by text line
Compiler directives	Yes, one by text line
Comments	No
Blank lines	No
Delimiters	
{ and *};* *begin* and *end*	Yes

3 Conducting the Experiment and Data Collection

Once developers finished the documentation of their seven programs, fifteen developers with r>0.5 were identified. By each programmer of those fifteen, both a simple linear regression equation (see Table 2) and a fuzzy logic system (see Table 3) were generated. The following list will serve to understand the following tables:

DP	Developer (labelled with A,B,..,O)	EEFL	Effort Estimation using Fuzzy Logic
APS	Actual Program Size (*LOC*)	r	Correlation (between APS-AE)
AE	Actual Effort (*minutes*)	a, b	Values of *a* and *b* of the linear regression equation ($E'=a + b*LOC$)
MRE	Magnitude of Relative Error		
EELR	Effort Estimation using Linear Regression		

3.1 Fuzzy Rules

The term fuzzy identification usually refers to the techniques and algorithms for constructing fuzzy models from data. There are two main approaches for obtaining a fuzzy model from data (Zhiwei and Khoshgoftaar, 2004):

1. The expert knowledge in a verbal form that is translated into a set of if–then rules. A certain model structure can be created, and parameters of this structure, such as membership functions and weights of rules, can be tuned using input and output data.
2. No prior knowledge about the system under study is initially used to formulate the rules, and a fuzzy model is constructed from data based on a certain algorithm. It is expected that extracted rules and membership functions can explain the system behavior. An expert can modify the rules or supply new ones based upon his or her own experience. The expert tuning is optional in this approach.

This paper is based on the first approach. The fuzzy rules based on the correlation (r) between pairs of variables were formulated. Then three rules were derived:

1. If size is small then effort is low
2. If size is medium then effort is average
3. If size is big then effort is high

Table 2. Correlation and linear equation values

DP	r	a	b
A	0.628	63.1417	1.81771
B	0.603	-16.8564	1.66873
C	0.882	1.63874	1.44997
D	0.743	37.976	0.90404
E	0.771	-72.8775	2.71398
F	0.715	51.9521	0.56968
G	0.581	43.0669	0.49747
H	0.722	-5.40177	0.87933

DP	r	a	b
I	0.873	89.355	0.78325
J	0.757	73.1073	0.63646
K	0.578	-7.16366	2.58765
L	0.841	74.3926	1.24053
M	0.526	75.6593	1.9392
N	0.643	58.582	2.4352
O	0.680	132.537	1.10438

In Table 3 parameters of input and output Membership Functions (MF) by developer are depicted. In accordance with an interval the values of *a*, *b* and *c* parameters were defined. Based upon the first approach of this section, from values close or equal to minimum as well as maximum of both program sizes and efforts, the intervals were adjusted iteratively until obtain the smallest MMRE possible. This interval was divided by three segments: *small*, *medium* and *big* (program size) and *low*, *average* and *high* (effort). In Figure 1 an example of a membership function plot corresponding to program size of developer *A* is depicted. All membership functions of all developers are triangular and their scalar parameters (a, b, c) are defined as follows:

$MF(x) = 0$ if $x < a$

$MF(x) = 1$ if $x = b$

$MF(x) = 0$ if $x > c$

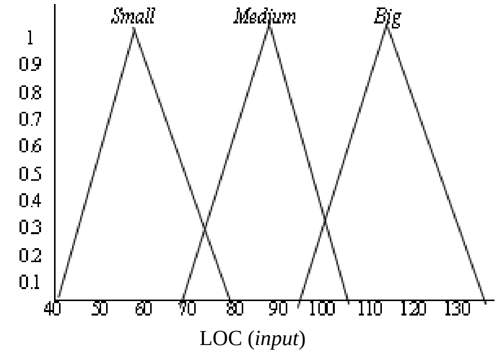


Fig. 1. Plot of Developer *A*

Table 3. Membership Function Characteristics

DP	Parame-ters	LOC (*input*)			Effort (*output*)		
		Small	**Medium**	**Big**	**Low**	**Average**	**High**
A	*a*	40	70	95	90	140	235
	b	60	85	115	145	215	285
	c	80	105	135	195	290	340
B	*a*	40	64	98	70	101	175
	b	58	90	114	101	152	227
	c	77	116	130	127	202	283
C	*a*	70	98	115	115	140	181
	b	88	115	130	133	171	210
	c	106	130	145	154	199	240
D	*a*	25	38	67	50	70	98
	b	38	62	83	68	90	123
	c	55	86	100	85	115	145
E	*a*	45	69	77	40	75	142
	b	60	76	85	75	130	180
	c	74	85	95	112	181	220
F	*a*	40	54	81	60	70	95
	b	56	82	105	72	90	107
	c	76	112	130	84	109	120
G	*a*	45	60	96	60	71	96
	b	57	88	112	69	89	112
	c	71	115	130	80	107	125
H	*a*	85	99	118	65	79	110
	b	98	112	129	80	101	124
	c	113	128	140	95	120	140

DP	Parame-ters	LOC (*input*)			Effort (*output*)		
		Small	**Medium**	**Big**	**Low**	**Average**	**High**
I	*a*	15	40	99	90	114	171
	b	41	85	126	127	155	197
	c	71	128	155	152	198	220
J	*a*	55	76	118	100	117	150
	b	76	108	140	114	138	163
	c	100	130	160	130	159	175
K	*a*	38	55	73	50	113	209
	b	50	68	85	93	183	263
	c	60	82	96	141	262	315
L	*a*	28	61	98	80	127	182
	b	54	86	129	113	185	239
	c	81	113	160	153	240	289
M	*a*	41	53	81	72	115	225
	b	55	74	94	120	187	268
	c	67	95	107	175	261	302
N	*a*	25	32	55	84	119	187
	b	32	47	61	117	158	202
	c	40	62	68	148	200	220
O	*a*	40	48	76	145	172	208
	b	52	70	88	165	202	230
	c	64	88	100	187	233	250

4 Analysis

Once linear regression equations as well as fuzzy systems by developer were generated, actual effort data were compared with the results of these two estimation models. Both simple regression and fuzzy system by developer were applied to same data (actual program size as input data). The MRE results are depicted in Table 4.

The three assumptions of residuals for MRE ANOVA must be analysed. Given that each programmer has his/her own regression model as well as fuzzy logic system, this analysis is done by developer. The following three assumptions of residuals by developer must be met or the ANOVA procedure does not apply:

1) Independent samples: The samples taken from population are independent of one another. In this study the population of developers is made up of separate programmers and each of them developed their own programs, hence the data are independent.

2) Equal standard deviations: The standard deviations of the variable under consideration are the same for all the populations. In a plot of this kind the residuals should fall roughly in a horizontal band centred and symmetric about the horizontal axis. From Figure 2a to 2o equal standard deviation plots are depicted.

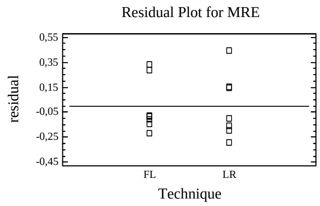
3) Normal populations: For each population, the variable under consideration is normally distributed. A normal probability plot of the residuals should be roughly linear. From Figure 3a to 3o normality plots are shown.

From plots of Figures 2 and 3, and based upon all data are independent, the Table 5 has been generated. In accordance with their plots, five developers have violated at least one assumption (labeled as C, E, K, M and N).

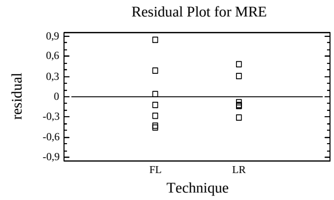
Table 4. MRE comparison

DP	APS	AE	EELR	MRE (AE-EELR)	EEFL	MRE (AE-EEFL)
	43	128	141.30	0.10	143	0.12
	56	95	164.93	0.74	143	0.51
	124	335	288.54	0.14	287	0.14
A	78	204	204.92	0.00	207	0.01
	130	250	299.44	0.20	287	0.15
	64	325	179.48	0.45	143	0.56
	70	132	190.38	0.44	143	0.08
	42	73	53.23	0.27	98.7	0.35
	102	120	153.35	0.28	181	0.51
	126	283	193.40	0.32	229	0.19
B	94	82	140.00	0.71	152	0.85
	77	153	111.64	0.27	152	0.01
	96	159	143.34	0.10	152	0.04
	106	85	160.03	0.88	196	1.31
	74	122	108.94	0.11	134	0.10
	84	118	123.44	0.05	134	0.14
	142	235	207.53	0.12	210	0.11
C	97	130	142.29	0.09	134	0.03
	125	159	182.88	0.15	195	0.23
	79	134	116.19	0.13	134	0.00
	98	127	143.74	0.13	134	0.06
	27	73	62.39	0.15	67.5	0.08
	64	77	95.83	0.24	91.7	0.19
	35	74	69.62	0.06	67.7	0.09
D	30	55	65.10	0.18	67.6	0.23
	98	141	126.57	0.10	122	0.13
	49	111	82.27	0.26	83.1	0.25
	60	63	92.22	0.46	91.7	0.46
	53	98	70.96	0.28	75.8	0.23
	69	89	114.39	0.29	75.8	0.15
	89	136	168.67	0.24	181	0.33
E	65	76	103.53	0.36	75.7	0.00
	84	216	155.10	0.28	169	0.22
	51	83	65.54	0.21	75.8	0.09
	50	43	62.82	0.46	75.9	0.77
	41	69	75.31	0.09	72	0.04
	54	69	82.72	0.20	72	0.04
	84	118	99.81	0.15	91.3	0.23
F	45	63	77.59	0.23	72	0.14
	129	115	125.44	0.09	107	0.07
	55	113	83.28	0.26	73.7	0.35
	56	81	83.85	0.04	75.1	0.07
	50	69	67.94	0.02	69.8	0.01
	56	62	70.93	0.14	69.7	0.12
	129	112	107.24	0.04	111	0.01
G	83	83	84.36	0.02	89	0.07
	102	74	93.81	0.27	97.4	0.32
	81	121	83.36	0.31	89	0.26
	81	70	83.36	0.19	89	0.27
	110	78	91.32	0.17	96.6	0.24
	89	68	72.86	0.07	80	0.18
	139	124	116.83	0.06	125	0.01
H	95	98	78.13	0.20	80	0.18
	117	86	97.48	0.13	99.9	0.16
	136	138	114.19	0.17	125	0.09
	136	93	114.19	0.23	125	0.34

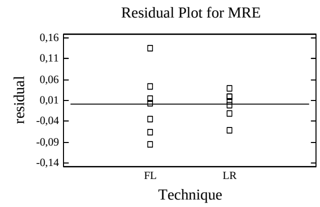
DP	APS	AE	EELR	MRE (AE-EELR)	EEFL	MRE (AE-EEFL)
	36	95	117.55	0.24	123	0.29
	42	125	122.25	0.02	126	0.01
	129	215	190.40	0.11	196	0.09
I	44	110	123.82	0.13	129	0.17
	149	185	206.06	0.11	196	0.06
	39	155	119.90	0.23	123	0.21
	20	100	105.02	0.05	122	0.22
	59	125	110.66	0.11	115	0.08
	78	124	122.75	0.01	118	0.05
	134	160	158.39	0.01	163	0.02
J	90	105	130.39	0.24	129	0.23
	131	165	156.48	0.05	163	0.01
	99	150	136.12	0.09	137	0.09
	96	120	134.21	0.12	134	0.12
	39	116	93.75	0.19	95.5	0.18
	49	91	119.63	0.31	94.7	0.04
	95	244	238.66	0.02	262	0.07
K	86	140	215.37	0.54	262	0.87
	75	314	186.91	0.40	200	0.36
	58	51	142.92	1.80	157	2.08
	56	179	137.74	0.23	120	0.33
	30	85	111.61	0.31	116	0.36
	34	110	116.57	0.06	116	0.05
	109	285	209.61	0.26	220	0.23
L	78	140	171.15	0.22	176	0.26
	155	230	266.67	0.16	236	0.03
	29	120	110.37	0.08	117	0.03
	40	140	124.01	0.11	116	0.17
	42	177	157.11	0.11	124	0.30
	59	177	190.07	0.07	152	0.14
	106	301	281.22	0.07	264	0.12
M	70	155	211.41	0.36	188	0.21
	53	277	178.44	0.36	122	0.56
	52	73	176.50	1.42	122	0.67
	56	219	184.26	0.16	140	0.36
	26	131	121.90	0.07	116	0.11
	34	134	141.38	0.06	127	0.05
	67	219	221.75	0.01	203	0.07
N	39	144	153.56	0.07	152	0.06
	41	199	158.43	0.20	159	0.20
	39	85	153.56	0.81	152	0.79
	41	197	158.43	0.20	159	0.19
	50	210	187.76	0.11	173	0.18
	61	208	199.90	0.04	194	0.07
	90	211	231.93	0.10	229	0.09
O	96	245	238.56	0.03	229	0.07
	56	230	194.38	0.15	184	0.20
	44	165	181.13	0.10	166	0.01
	46	148	183.34	0.24	166	0.12



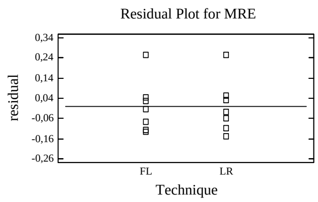
2a. Developer A



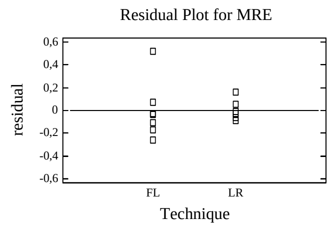
2b Developer B



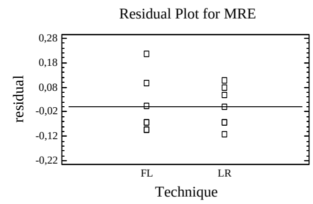
2c. Developer C



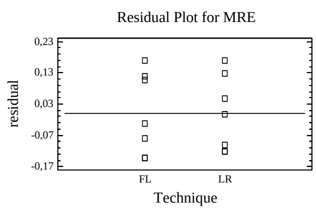
2d. Developer D



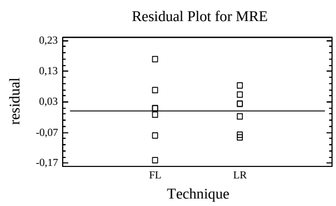
2e. Developer E



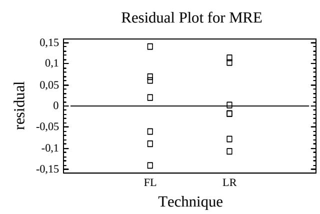
2f. Developer F



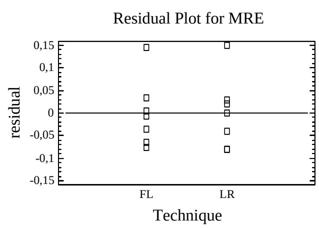
2g. Developer G



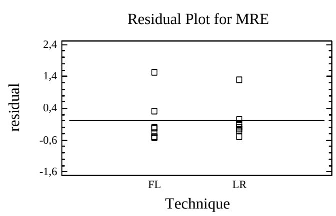
2h. Developer H



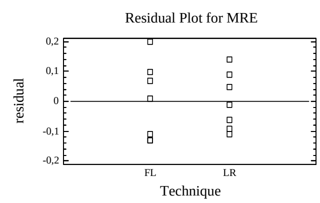
2i. Developer I



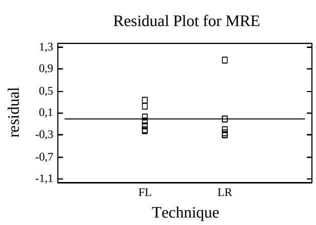
2j. Developer J



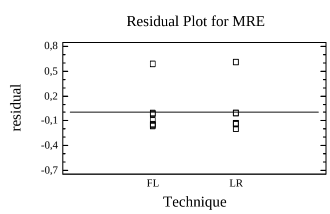
2k. Developer K



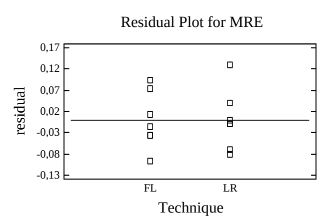
2l. Developer L



2m. Developer M

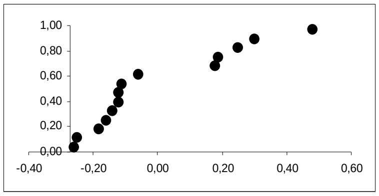


2n. Developer N

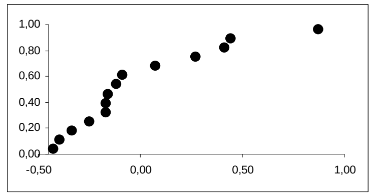


2o. Developer O

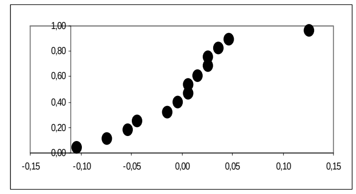
Fig. 2. Equal standard deviation plots



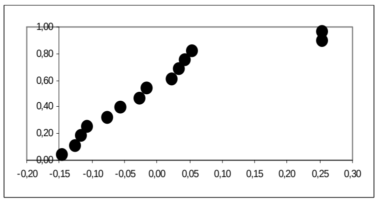
3a. Developer A



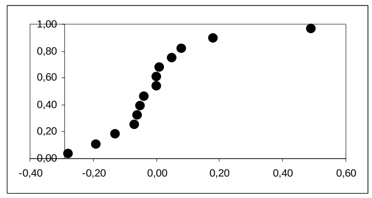
3b. Developer B



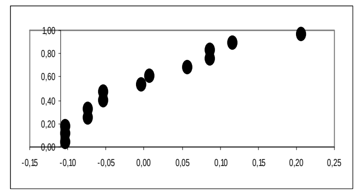
2c. Developer C



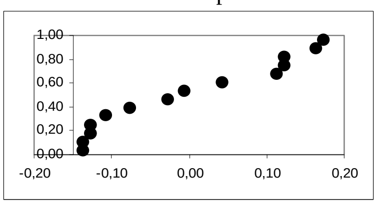
3d. Developer D



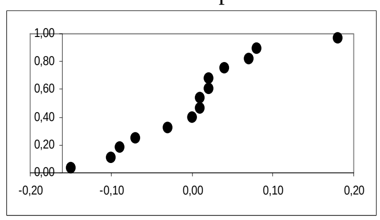
3e. Developer E



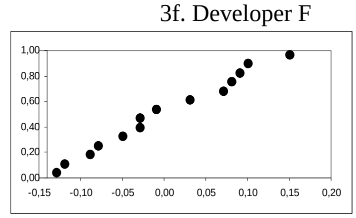
3f. Developer F



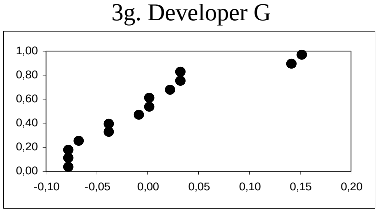
3g. Developer G



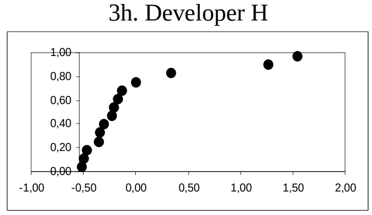
3h. Developer H



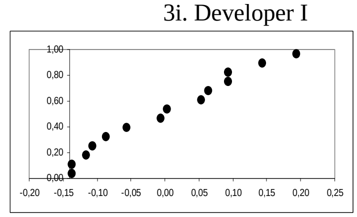
3i. Developer I



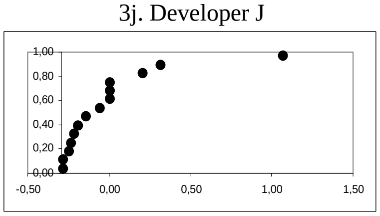
3j. Developer J



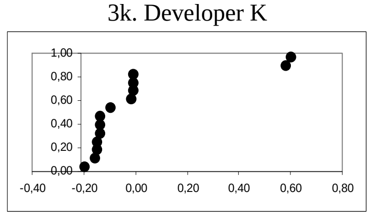
3k. Developer K



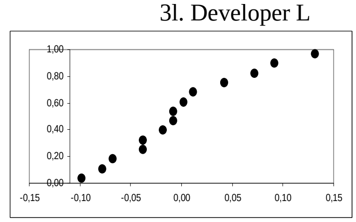
3l. Developer L



3m. Developer M



3n. Developer N



3o. Developer O

Fig. 3. Normality plots

Table 5. Analysis of residuals (NV: no violated, V: violated)

Residual Assumption	**Developer**														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Independence	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV
Equal standard deviations	NV	NV	V	NV	V	NV	NV	NV	NV	NV	NV	NV	V	NV	NV
Normality	NV	NV	NV	NV	NV	NV	NV	NV	NV	NV	V	NV	V	V	NV

5 Presentation of Results

Given that five of fifteen developers have violated at least an ANOVA residual assumption, ten of them must only be considered for generating conclusions of this study. For the ten developers, the MMRE by estimation model is calculated.

In Table 6 the best (lower) MMRE by developer are highlighted in bold. The Linear Regression (LR) model presented four cases (40%) with best MMRE, while Fuzzy Logic (FL) systems showed six developers (60%) with best MMRE. In Table 7 the result of the analysis of variance for MRE by each of ten developers is depicted. The p-values test the statistical significance of each of the factor (technique). If a p-value is less than 0.05 then it has a statistically significant effect on MRE at the 95 % confidence level. Since the p-values of the F-test is greater than 0.05, there is not a statistically significant difference between the mean MRE from one level (linear regression) of technique to another (fuzzy logic) at the 95 % confidence level.

Table 6. MMRE comparison (DP: developer, LR: Linear Regression, FL: Fuzzy Logic)

DP	**MMRE**	
	LR	FL
A	0.30	**0.22**
B	**0.40**	0.47
D	0.21	**0.20**
F	0.15	**0.14**
G	**0.14**	0.15

DP	**MMRE**	
	LR	FL
H	**0.15**	0.17
I	**0.13**	0.15
J	0.09	**0.08**
L	0.17	**0.16**
O	0.11	**0.10**

A Multiple Range Tests for MRE by Technique indicate the technique having the better result by developer.

In addition, Table 8 applies a multiple comparison procedure to determine which means are significantly different from the others. The method currently being used to discriminate among the means is Fisher's least significant difference (LSD) procedure. In Table 8 each of the absolute values in the “difference” column is lower than its LSD value. It indicates that both techniques are not significantly different each other.

Table 7. MRE ANOVA (DP: developer)

DP	**F-ratio**	**p-value**
A	0.31	0.5864
B	0.09	0.7725
D	0.00	0.9682
F	0.10	0.7560

DP	**F-ratio**	**p-value**
G	0.02	0.8840
H	0.27	0.6106
I	0.22	0.6497
J	0.01	0.9190

DP	**F-ratio**	**p-value**
L	0.03	0.8720
O	0.01	0.9090

Table 8. Multiple Range Tests for MRE by Technique

Developer	Technique	LS Mean (MMRE)	Difference	LSD value
A	FL	0.2242	-0.071	0.278
	LR	0.2957		
B	FL	0.4657	0.061	0.452
	LR	0.4042		
D	FL	0.2042	-0.002	0.153
	LR	0.2071		
F	FL	0.1342	-0.017	0.117
	LR	0.1514		
G	FL	0.1514	0.010	0.146
	LR	0.1414		

Developer	Technique	LS Mean (MMRE)	Difference	LSD value
H	FL	0.1714	0.024	0.101
	LR	0.1471		
I	FL	0.1500	0.022	0.106
	LR	0.1271		
J	FL	0.0857	-0.004	0.089
	LR	0.0900		
L	FL	0.1614	-0.010	0.132
	LR	0.1714		
O	FL	0.1057	-0.004	0.079
	LR	0.1100		

To create the fuzzy logic rules MATLAB 6.1 was used, while to the linear regression equations, ANOVA and plots Statgraphics 4.0 was used.

6 Conclusions and Future Research

This paper applied a model adequacy checking (verification) when each Fuzzy Logic System (FLS) was compared with linear regression. For software development effort estimation at personal level these two models were used. From data of a replicated experiment using a sample integrated by ten developers these FLS were derived. Each programmer developed seven small programs based on a common process as well as inside a controlled environment.

After comparisons based on MMRE as well as ANOVA, a fuzzy model adequacy checking was done, showing that a FLS can represent an alternative for estimating the software development effort at personal level when (a) correlation between program size (lines of code) and effort (minutes) was higher than 0.5, and (b) the three assumptions of residuals for MRE ANOVA were met.

In this experiment 22 developers participated, 7 of them were excluded because they did not obtain a $r>0.5$. Some of them presented inconsistent behaviour when they developed their programs. In future experiments a cooperation of others developers will be necessary.

Future research will involve (a) the use of the generated fuzzy logic systems to estimate effort of new programs by developer, that is, to validate the fuzzy models whose adequacy was checked in this paper, and (b) the generation of a generalized fuzzy logic model for estimating the development effort at personal level.

Acknowledgements

We would like to thank Center for Computing Research, National Polytechnic Institute, Mexico as well as CONACYT. Moreover, to Development Team which is working for Federal Commission of Electricity at Guadalajara, Jalisco, México as well as the bachelor students of Computational Systems Engineering of the University del Valle de Atemajac (UNIVA), Guadalajara.

References

1. **Ahmed M. A, Saliu M. O.** and **AlGhamdi J.**, "Adaptive fuzzy logic-based framework for software development effort prediction", *Information and Software Technology*, Vol. 47, No. 1, 2005, pp. 31-48.
2. **Boehm B. W.** and **Fairley R. E.**, "Software Estimation Perspectives", IEEE Software November 2000, pp 22-26.

3. **Boehm B., Abts Ch.** And **Chulani S.**, “Software Development Cost Estimation Approaches – A Survey”. Chulani Ph. D. Report 1998.
4. **Boehm B.**, *Software Engineering Economics*, Prentice Hall, 1981.
5. **Boehm B.** COCOMO II. Prentice Hall. 2000.
6. **Braz, M.** and **Vergilio S.**, “Using Fuzzy Theory for Effort Estimation of Object-Oriented Software”, *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI 2004.
7. **Briand L.C., Langley T.** and **Wieczorek I.**, “A replicated Assessment and Comparison of Common Software Cost Modeling Techniques”, *IEEE International Conference on Software Engineering* (ICSE), 2000, Limerick, Ireland.
8. **Briand L.C., Emam K.E., Surmann D.** and **Wieczorek I.**, “An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques”. *ISERN*-1998-27.
9. **Brooks F. P. Jr.**, “Three Great Challenges for Half-Century-Old Computer Science”, *Journal of the ACM*, Vol. 50, No. 1, January 2003, pp. 25-26.
10. **Burguess C. J.** and **Lefley M**. Can genetic programming improve software effort estimation? A comparative evaluation. Information and Software Technology. Elsevier. 2001.
11. **Gray A. R.** and **MacDonell S. G.**, “Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation”, *Proceedings of NAFIPS* 1997.
12. **Höst M** and **Wohlin C.,** “A subjective effort estimation experiment”, *Information and Software Technology, Elsevier*, 1997.
13. **Huang X., Ren J.** and **Capretz L.F.**, “A Neuro-Fuzzy Tool for Software Estimation”, *Proceedings of the 20th IEEE International Conference on Software Maintenance*, 2004.
14. **Humphrey W**. The Personal Software Process, Technical Report CMU/SEI-2000-022.
15. **Humphrey W**. A Discipline for Software Engineering, Addison Wesley, 1995.
16. **Idri, A., Abran, A.** and **Khoshgoftaar T. M.**, “Computational Intelligence in Empirical Software Engineering”, *First USA-Morocco Workshop on Information Technology*, ENSIAS, Rabat, 2003.
17. **Idri A., Khoshgoftaar T. M., Abran A.** Can Neural Networks be Easily Interpreted in Software Cost Estimation. World Congress on Computational Intelligence. Hawai. 2002a.
18. **Idri, A., Abran, A.** and **Khoshgoftaar T.**, “Estimating Software Project Effort by Analogy Based on Linguistic Values”, *Proceedings of the Eight IEEE Symposium on Software Metrics* (METRIC), 2002b.
19. **Idri, A., Abran, A.** and **Khoshgoftaar, T.**, “Fuzzy Analogy: a New Approach for Software Cost Estimation”, *International Workshop on Software Measurement* (IWSM´01), Montréal, Québec, Canada, August 28-29, 2001.
20. **Jorgensen M., Kirkeboen G., Sjoberg D., Anda B.** and **Brathall L.**, “Human Judgement in Effort Estimation of Software Projects”, *International Conference on Software Engineering*, Limerick, Ireland, 2000.
21. **Kadoda G., Cartwright M., Chen L., Shepperd M.** Experiences Using Case-Based Reasoning to Predict Software Project Effort. Proceedings of the EASE Conference Keele, UK. 2000.
22. **Kitchenham B. A., Pfleeger S. L., Pickard L. M., Jones P. W., Hoaglin D. C., Emam K. E.** and **Rosenberg J.**, “Preliminary Guidelines for Empirical Research in Software Engineering”, *IEEE Transactions on Software Engineering*, Vol. 28, No. 8, August 2002.
23. **Kok P., Kitchenhan B.A, Kirakowski J**. The MERMAID Approach to Software Cost Estimation. Proceedings ESPRIT Technical week. 1990.
24. **Lind D. A., Mason R. D., Marchal W. G.,** Basic Statistics, McGraw Hill, 2000.
25. **López-Martín Cuauhtémoc, Leboeuf J., Yáñez Cornelio** and **Agustín Gutiérrez T., “**Software Development Effort Estimation Using Fuzzy Logic: A Case Study”, *Encuentro Internacional de Ciencias de la Computación, IEEE Computer Society Press*, September, 2005, pp. 113-120.
26. **MacDonell S. G.** and **Gray A. R.**, “Alternatives to Regression Models for Estimating Software Projects”, *Proceedings of the IFPUG Fall Conference*, 1996.
27. **MacDonell S. G.**. “Software source code sizing using fuzzy logic modelling”. *Elsevier Science*, 2003.
28. **Mendes E., Mosley N.** and **Watson I.**, “A Comparison of Case-Based Reasoning Approaches to Web Hypermedia project Cost Estimation”, ACM, 2002.

29. **Musflek P., Pedrycz W., Succi G.** and **Reformat M.,** "Software Cost Estimation with Fuzzy Models", *Applied Computing Review*, Vol. 8, No. 2, 2000, pages 24-29.
30. **Park R. E.**, "Software Size Measurement: A Framework for Counting Source Statements", *SEI, Carnegie Mellon University*, September 1992.
31. **Schofield C.**, "Non-Algorithmic Effort Estimation Techniques", *ESERG*, TR98-01.
32. **Seaman C. B.**, "Qualitative Methods in Empirical Studies of Software Engineering", *IEEE Transactions on Software Engineering*, Vol. 25, No. 4, July/August, 1999
33. **Secretaría de Economia**, "Programa para el Desarrollo de la Industria del Software", June 2002.
34. **Srinivasan K., Fisher D**. Machine Learning Approaches to Estimating Software Development Effort. IEEE Transactions on Software Engineering, Vol. 21, No. 2, 1995.
35. **Weiss** N.A. Introductory Statistics. Addison Wesley. Fifth Edition. 1999.
36. **Zhiwei Xu Z.** and **Khoshgoftaar T. M.**, "Identification of fuzzy models of software cost estimation". *Elsevier Fuzzy Sets and Systems*. Volume 145, Issue 1, 1 July 2004, pp.141-163.

Appendix

The following lists include the identifiers, names, programming language, and their job/university of developers:
a) Development Team, Federal Commission of Electricity from Guadalajara, Director's email: omar.delacruz@cfe.gob.mx. *A: (Alatorre Carranza N., C) B: (De la Cruz Preciado O., Pascal); C: (Flores Gómez C., COBOL); D: (Galindo Gauna R., C); E: (García Ramos M., C); F: (Guerra Martínez A., Pascal); G: (Guzmán Martínez A., C); H: (Hernández Hernández P., COBOL) I: (Hernández Ramos A., COBOL); J: (Partida Menchuca L., COBOL).*
b) Bachelor Students, University del Valle de Atemajac (UNIVA), Guadalajara, http://www.univa.mx/, Director's email: martin.rodriguez@univa.mx. *K: (Becerril Ramírez J., Delphi); L: (Herrera Rábago F., Pascal); M: (Navarro Rodríguez J., C); N: (Santana Ruelas J., C) O: (Vargas Mora D., JAVA).*

Cuauhtémoc López Martín *Graduated in Computer Engineering in the Universidad Autonoma de Tlaxcala in 1995. He received a M. Sc. Degree in Information Systems from Universidad de Guadalajara in 2000, and a Ph. D. in Computer Science in the Center for Computing Research of the National Polytechnic Institute of Mexico, in 2007. He has been professor at three universities and he has developed software for several organizations. His research interests are on software engineering, specifically on software development effort estimation.*

Cornelio Yáñez Márquez *Received his BS degree in Physics and Mathematics from the Escuela Superior de Física y Matemáticas at IPN in 1989; the MSc degree in Computing Engineering from the CINTEC-IPN in 1995, and the PhD from the Center for Computing Research of the National Polytechnic Institute of Mexico in 2002, receiving the Lázaro Cárdenas Award 2002. Currently, he is a titular professor at the CIC-IPN. His research interests include associative memories, mathematical morphology and neural networks.*

Agustín Francisco Gutiérrez Tornés *He received a B.Sc. degree in Economics (Universidad de La Habana, Cuba, 1970). He received a Ph.D. degree in Agricultural Sciences (Economics Department, Akademia Rolnicza, SGGW-AR, Wassaw, Poland, 1984). He is currently Systems Coordinator at BANAMEX, S.A. Moreover, he acts as Subject Professor at the Monterrey Technological and Higher Studies Institute (ITESM, México city). He has a wide experience as researcher and professor of topics related to Software Engineering. He has worked as consultant and professor for several international universities and organizations of the United Nations.*

Edgardo Manuel Felipe Riverón *He received the B.Sc. degree in Electrical Engineering from the Higher Polytechnic Institute Jose Antonio Echeverria, in Havana, Cuba, in 1967. He received the Ph.D. degree in Technical Sciences from the Computer and Automation Research Institute, in Budapest, Hungary, in 1977. He is currently Full Professor and Senior Researcher at the Center for Computing Research of the National Polytechnic Institute of Mexico. His research interests are on Image Processing and Image Analysis, Computer Vision and Pattern Recognition, in particular color quantization, retina analysis, biometric solutions, document analysis and mathematical morphology applications.*