ELSEVIER

Information and Software Technology 43 (2001) 61–72

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.nl/locate/infsof

On the problem of the software cost function

J.J. Dolado

Facultad de Informática, Universidad del País Vasco-Euskal Herriko Unibertsitatea, 20 009 Donostia, Spain

Received 24 February 2000; revised 15 June 2000; accepted 16 June 2000

Abstract

The question of finding a function for software cost estimation is a long-standing issue in the software engineering field. The results of other works have shown different patterns for the unknown function, which relates software size to project cost (effort). In this work, the research about this problem has been made by using the technique of Genetic Programming (GP) for exploring the possible cost functions. Both standard regression analysis and GP have been applied and compared on several data sets. However, regardless of the method, the basic size–effort relationship does not show satisfactory results, from the predictive point of view, across all data sets. One of the results of this work is that we have not found significant deviations from the linear model in the software cost functions. This result comes from the marginal cost analysis of the equations with best predictive values.

Keywords: Software cost function; Genetic programming; Cost estimation; Empirical research

1. Introduction

The prediction of the effort to be consumed in a software project is, probably, the most sought after variable in the process of project management. The determination of the value of this variable in the early stages of a software project drives the planning of the remaining activities. The estimation activity is plagued with uncertainties and obstacles, and the measurement of past projects is a necessary step for solving the question. The problem of accurate effort estimation is still open and the project manager is confronted at the beginning of the project with the same quagmires as a few years ago.

The methods for analyzing the data have been varied, but the referent technique is always the classical regression method. Other methods such as neural networks, case-based reasoning, fuzzy logic and expert judgment have been applied as estimation methods. The motivation for using different methods to infer the relationship is the implicit understanding that, depending on the method used, we could probably obtain a different relationship between the variables under study. Some of these methods provide a symbolic representation of the function relating the input to the output, thus characterizing the general pattern of behavior of the variables. These functions are known as *cost functions*, *economic functions* or *production functions*, and they provide a symbolic interpretation of the type of economy of scale prevailing in software development. In this way, neural networks or similar methods, which embed the pattern being analyzed into the data structure used for model building, are disadvantageous for the software manager, since they do not allow the manipulation of the model variables.

The overall procedure for the analysis that follows is to use both classical regression and GP on different publicly available data sets, and to select the equations which best values obtain from the predictive point of view (*level of prediction* and *relative error*). Classical regression makes some assumptions about the data distribution for characterizing the equations. On the contrary, GP does not make any assumption and, furthermore, it explores a larger set of potential equations. Therefore, we can observe which method performs better. Besides that, the predictive results of the estimation of effort, by using size as the independent variable, do not show good values in all data sets, that led us to believe that size, per se, may not be considered as the key cost driver or cost factor for software development.

Afterwards, we will take the first derivative (the marginal cost) of the best equations obtained with both methods across all data sets, so we can identify the function which characterizes the marginal cost in every data set. Comparing those curves we will infer that if we had to assign a model to the software cost function (size–effort relationship) the most convincing candidate is the linear model.

The rest of the article is structured as follows: Section 2 reviews the major questions for building software cost

E-mail address: dolado@si.ehu.es (J.J. Dolado).

0950-5849/01/$ - see front matter
PII: S0950-5849(00)00137-3

functions, examining the issue of model validation. Section 3 describes the Genetic Programming (GP) approach for deriving symbolic equations. Section 4 presents the results obtained by the methods and analyzes the issue of the linearity of the cost function. Finally, Section 5 discusses the implications of the results.

2. The software cost functions

The consumption of resources can be seen from two complementary viewpoints: as a cost function or as a production function [10,20,21]. The effort is the input to the process and the output is the software product. This problem is the dual one of finding a cost function for the software product, although from the methodological point of view the issues are similar in both cases. Almost all works in the literature discuss the problem of project estimation from the perspective of the identification of the cost function, in which the problem under study is to identify the function which relates the size of the product to the man-months employed. This is the approach followed here.

2.1. Building and validating empirical cost models

The underlying idea for building a model is to consider that the software product is manufactured in some way, and the factor that affects the cost of the project is the size of the software. The equation relating size and effort can be adjusted for different environment factors, such as productivity, tools, complexity of the product and others. The equations are calibrated to fit to the actual data points of the projects.

According to this view several patterns of equations have emerged [20], but none of them has produced sufficient evidence as to be considered the definitive cost (or production) function for software, if there is any. The functions proposed range from the linear model ($E = a_0 + a_1 S$), quadratic models ($E = a_0 + a_1 S + a_2 S^2$), Cobb–Douglas model ($E = aS^b$) and translog models ($\ln E = a_0 + a_1 \ln S + a_2 \ln^2 S$), E being the effort employed in developing a software project of size S.

The issue of the existence of economies or diseconomies of scale has caused heated discussions among researchers [5,6,24]. In a situation of economies of scale the returns of the process increase marginally as the volume of the production increases, that is, the *marginal cost* of producing an additional unit decreases as more units are produced. The opposite happens if there are diseconomies of scale, where the marginal cost of producing an additional unit of output increases. As they have different implications for the management of the project, it is desirable to identify the existence or not of economies of scale. For instance, in the case of Cobb–Douglas models, the exponent $b > 1$ indicates diseconomies of scale and $b < 1$ economies of scale. Depending on the data set used, both types of economies have been found [6]. However, Bieman et al. have criticized some of the conclusions related to this issue, arguing that few data sets showed values so significantly different from 1 as to assert, with clear evidence, the existence or not of such economies of scales. According to these authors, even more questionable is the association of specific values of b with small or large projects [8]. GP will help us to tackle this issue below, since it is the only technique which can provide equations analogous to those of classical regression. The examination of the marginal cost of all equations will ascertain if there are significant deviations from the linear model.

The study of the existence or not of economies of scale implies the study of the existence of non-linearities in the cost functions. Some results have tended to postulate the existence of non-linearities, confronting purely linear models. In one of the latest works, the quadratic function has been postulated as the most plausible candidate, based on the results of the Davidson and MacKinnon tests applied to alternative models of the same data set under study [20].

The validation of a model is also a persistent issue in the construction of software cost functions. Depending on the authors, different measures of the goodness of fit of the model have been considered. The coefficient of multiple determination R^2, and adjusted R^2, obtained from regression analysis is being used as a measure of the capability of explanation of the model. Analysis of the residuals seems a necessary condition for examining the aptness of the model, and outlier examination has been suggested for examining the model stability. There is a wide consensus in using the *mean squared error* (mse) as an essential element for assessing a regression model. The mse was also considered by Matson et al. [26] as the best criterion to follow for assessing the regressions, jointly with the identification of outliers. However, some criticisms argue that the classical criterion of the least-squares method can be troublesome in the presence of outliers. Miyazaki et al. proposed improvements to this criterion by using the least squares of inverted balanced relative errors [28], but Shepperd and Schofield [29] subsequently refuted this. Anyway, we regard the mentioned criteria as measures of *explanation* of the fitted equations. For evaluating the *predictive capability* of a model, some well-known measures have been proposed: Relative error, predictions at level l, usually $l = 25$ (PRED(0.25)), and mean magnitude of the relative error (MMRE) (see Appendix A).

Over and above all these criteria, the characteristic that has to be fulfilled by an estimation equation is that of accuracy: "the model should be able to accurately estimate as many of the actual values as possible" [28]. In this way, and in order to allow comparison with other works, the main measures used in the next section are the mse, for building regression equations, and PRED(0.25) and MMRE (as defined in Ref. [11]) for evaluating their predictive power.

2.2. Cost accounting and software development

In designing a cost system, the essential question is to

identify the cost drivers and the cost(s) function(s). Regression analysis has been one of the classical tools for identifying those functions (see Ref. [19], chap. 10). There are two basic criteria, apart from those of statistical validity, which make a cost function valid and comparable to other functions. The first consists in that the function defined must have "economic plausibility", meaning by this that the relation identified should show a reasonable behavior, and should make economic sense. The second criterion consists in that the cost functions are only valid within the range of the data used to derive the equation. If cost functions are to be compared, they should be derived under the same assumptions.

With these ideas in mind we can restate some of these concepts for software cost estimation. First, it is noticeable that the paradigm used for accounting costs in software is the manufacturing one, in which the costs are related to the size of the product built. The collection of the data sets, cited in Appendix B and publicly available, shows that the main assumed cost driver is the size of the product, whatever the measure used may be.

It is worth noting here that the second criterion, the range of validity, has consequences in how the cost functions can be compared. Some of the estimation models provided in the literature construct the cost function by fitting an equation to the data points without caring about the reasonable behavior outside that range. Specifically, most of the functions proposed do not cross the origin (0,0), which is counterintuitive, and it also obstructs the comparison among functions. No effort can be assigned to a project of null size, and conversely, it is expected that any project, however small the size, will consume some effort. This forces the cost function, with effort as the dependent variable, to cross the origin. Above that point, the function is believed to be increasing. Although some vague interpretations can be found if the independent coefficient results positive in the equation, none can be perceived if the regression gives a negative coefficient.

Not all cost functions are built with the requirement of crossing the origin in mind, nor should they be obliged to fulfill it. Therefore, regression equations can only be interpreted within the range of the data and/or within the assumptions used to define the equations. Another issue is the elimination of outliers. We cannot remove points from the data sets solely on the basis of the method of analysis, if they have been collected under the same assumptions [17]. There is no scientific justification for eliminating proofs of the phenomenon under study simply because they do not fit adequately within the method and its corresponding results. Points can only be removed based on different assumptions when collecting the data, as Abran and Robillard [2] did with their original data set. This issue is taken seriously here, and no data point is taken out of the sample unless there is serious effects on the predictions. The usual measures of Cook's and Mahalanobis' distances for detecting anomalies have been computed, and this analysis has been embedded within the overall procedures used for obtaining the best predictive and accurate values, as reported throughout the rest of the paper.

Finally, we use all data points for model building and model evaluation for a very important reason, that is to make the results comparable to the works cited. However, we must be aware that when we use all data points, accuracy is probably overestimated, and real accuracy may be less than stated (therefore we are setting limits to the predictions). In some instances, we have not observed any practical benefit when using a sample (or even several different samples) of the data sets [14], but that may not always be the case.

3. Genetic programming for software cost estimation

GP is a type of evolutionary computation technique. An evolutionary algorithm provides a general structure for solving problems, which mimic the biological paradigm of the "survival of the fittest" [3]. GP is a variation on the genetic algorithms, in which the representation of the chromosomes is not a binary string but other data structures [25]. Since there is no restriction on the data structure to be evolved by the evolutionary algorithm, the result can be a program, an equation, a circuit, a plan or any other representation. The structures to be evolved in the symbolic regression problem are trees, which represent well-formed equations (traversed in order). That is, GP is used here as a symbolic regression technique. The structure of the genetic programming algorithm for symbolic regression is as follows [25,27,32]:

```
Genetic Programming Algorithm
NumGenerations = 0;
Generate initial population P (of size N)
of equations;
While not TerminateCondition do
  Evaluate Fitness of each equation
  For each equation in the population P
  select randomly one of
    (a) Mutation with probability Pm
    (b) Crossover with probability Pc
    (c) Direct reproduction with probabil-
    ity (1- Pm -Pc)
    Add the new equation to the new popula-
    tion
  endfor
endwhile
```

`Fitness` is a measure of how well an equation is a solution to the problem. `TerminateCondition` can be either a limit on `NumGenerations` (the number of generations already evolved) or an acceptable limit for the fitness value. Other parameters to control in the algorithm are the different probabilities for chromosome recombination, and

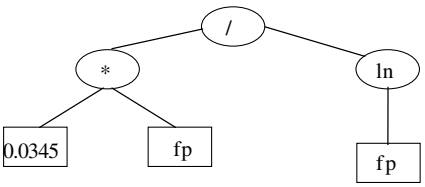

Fig. 1. Tree representation of the equation $0.03453 \times (fp)/(\ln(fp))$.

the selection of equations from the old population. The result of the algorithm is a set of equations evolved to better fit the data set.

There are works which deal with the application of GP to the identification of physical or chemical systems such as those described in Refs. [27,32]. We have already applied GP to software engineering problems. In Ref. [15] GP was applied to the problem of software size estimation, and in Ref. [14], we applied GP to software cost estimation in some data sets, but without the requirements stated in Sections 2.1 and 2.2.

In this work, we use the GP implementation of McKay et al. [27,32]. It is based on the general structure of an evolutionary algorithm, with some adaptations as explained below. The elements to be evolved are trees representing equations (see Fig. 1). That is, the population P of the algorithm stated above is a set of trees to which the crossover and mutation operators are applied. The terminal nodes are constants or variables, and the non-terminals are the functions which are available for system definition, which in the present work are $+$, $-$, *, /, $\wedge$, square, square root, natural logarithm and exponential.

The fitness function, one of the main components of the algorithm, is defined according to classical measures for fitting equations to the data. Two variables have been used for this purpose: the mse and the correlation coefficient. The mse quantifies the error of the predictions made by the estimated equation. The correlation coefficient measures how predicted and actual values together may vary. Both measures have been used as fitness functions, the latter being preferred in the works of McKay et al. [27]. In the present work, the correlation coefficient has not been especially valuable, and in many cases was unable to discriminate between equations. We have chosen the mse as the fitness measure here, since it is a usual and consensued measure of assessing regression models. We compute the values of PRED(0.25) and MMRE, once the GP obtains the best equation in a simulation run.

Once a population of equations (a generation) has all its members with a fitness value assigned to each one, the next step is to choose the elements that will be used to form the next generation. GP can use several methods at this step, like the elitist strategy, tournament selection and fitness proportionate selection. Fitness proportionate selection is a method which selects an individual of the population with a probability proportional to its fitness, according to the value of f_i/F, where $F = \sum_{i=1}^{N} f_i$and f_i is the fitness of the individual i in the population N. We use the latter in this work because it allows to maintain diversity in the population; however, there are no clear guidelines on this issue.

The GP algorithm applies mutation and crossover operators to the selected elements. The algorithm selects some elements, according to their fitness, for *direct reproduction*, and it is possible to maintain a fixed proportion of the old population for the next generation. The *crossover* operation consists on exchanging subtrees of two equations, while maintaining the syntactic correctness of the new formed trees (equations). The choice of the operator ($+$, $-$, square root, etc.) is probabilistic. *Mutation* randomly takes an element (variable, function or constant) of the tree equation and replaces it with other elements. The next generation is formed by the new generated elements jointly with the part of the old population that directly passes to the next. In our case, simulations have shown that the best results have appeared with low values in the parameter `NumGenerations`. Once the tree that represents the equation has been completely constructed a non-linear least-squares optimization is performed in order to obtain the best values of the constants in every equation. The computational environment (Matlab 4.2.c) simplifies the final expression of the equations. In some cases, we have further simplified the equations manually, as in the cases of operations between constants, in order to have a more attractive presentation.

4. Application of genetic programming and classical regression

We have applied these two techniques to the data sets described in Appendix B. The parameters for GP have been set heuristically. The number of runs of the GP algorithm has varied between 100 and 200, in which the main parameters of the algorithm have manually been set with different values. Most of the times convergence has been very quick, and the best equations have been found with very few generations (three to six). The number of runs is large enough, to allow us to consider the results as representative.

Table 1 summarizes the main results obtained from applying the methods. The functions examined by curve estimation include linear, logarithmic, inverse, quadratic, cubic, power, exponential, growth and logistic (using the statistical package SPSS). We have further simplified some of the symbolic expressions obtained by GP, and presented in Table 1, for the sake of clarity. The number of points involved in each analysis should be borne in mind when comparing the level of prediction and the MMRE, since the data sets have different sizes. It is important to remark here that the syntax of the equations of GP obscures the fact that those equations are imperceptible deviations from the linear case, as the column of the marginal cost summarizes. GP has only tried to better adjust the fit.

Since GP is able to approximate the data with very

Table 1
Predictive values and marginal cost of each data set. The deviations from the linear trend are insignificant in both GP and curve estimation

Data sets		Curve estimation	Genetic programming	Marginal cost
Abran and Robillard [2]				
A	Equation	$2.649261 \times fp$	$6.435 \times fp^{0.8245}$	Unclear (insignificant economies of scale)
	PRED (0.25)	57.14	77.3	
	MMRE	0.2364	0.256	
Albrecht and Gaffney [1]				
B	Equation	$0.001099 \times fp^{1.487282}$	$1.06 \times 10^{-8} \times (1.775 \times 10^{6} \times fp + fp^{3})$	Insignificant diseconomies of scale
	PRED(0.25)	54.17	64	
	MMRE	0.5313	0.548	
Bailey and Basili [4]				
C	Equation	$1.856339 \times kloc^{0.868942}$	$kloc + \sqrt{1.416 \times kloc}$	Low economies of scale
	PRED(0.25)	61.11	73.7	
	MMRE	0.2935	0.269	
Belady and Lehman [7]				
D	Equation	$0.003067 \times loc^{1.0607091}$	$3.83 \times 10^{-5}\text{loc}^{3/2}(1 - 1.2 \times 10^{-6} \times loc)$	Unclear
	PRED(0.25)	33.33	35.3	
	MMRE	0.6258	0.71	
Boehm [9]				
E	Equation	$0.001852 \times \text{adjkdsi.}^{1.108397}$	$9.432 \times 10^{-3}\ adjkdsi$	Unclear
	PRED(0.25)	17.46	15.6	
	MMRE	1.1336	1.781	
Heiat and Heiat [18]				
F	Equation	$24.856892 \times reio^{0.698162}$	$25.21 \times reio^{0.6959}$	Very low economies of scale
	PRED(0.25)	94.29	94.4	
	MMRE	0.0892	0.087	
Academic environment [13]				
G	Equation	$loc^{0.730624}$	$0.08843 \times loc + (824.5/\ln(loc))$	Unclear (insignificant economies of scale)
	PRED(0.25)	43.75	46.9	
	MMRE	0.3888	0.423	
Kemerer [22]				
H	Equation	$0.348135 \times fp^{0.903874}$	Many terms	Unclear (insignificant diseconomies of scale)
	PRED(0.25)	33.33	62.5	
	MMRE	0.4435	0.584	
Miyazaki et al. [28]				
I	Equation	$0.946861 \times kloc^{0.986412}$	$kloc + 5.3 \times 10^{-11} \times kloc^{5}$	Very low economies of scale
	PRED(0.25)	42.55	47.9	
	MMRE	0.3999	0.506	
Shepperd and Schofield [29]				
J	Equation	$13.496936 \times files^{0.655187}$	$54.93 \times files^{0.6621}$	Very low economies of scale
	PRED(0.25)	50	57.9	
	MMRE	0.4489	0.456	
Desharnais [12]				
K	Equation	$21.47 \times fp^{0.935265}$	$32.9 \times fp^{0.8854}$	Low economies of scale
	PRED(0.25)	44.26	51.6	
	MMRE	0.5428	0.623	
Kitchenham and Taylor [23]				
L	Equation	$0.15474 \times fp^{0.816038}$	$0.03453 \times (fp/\ln(fp))$	Unclear (insignificant economies of scale)
	PRED(0.25)	27.27	32.4	
	MMRE	0.8458	1.143	

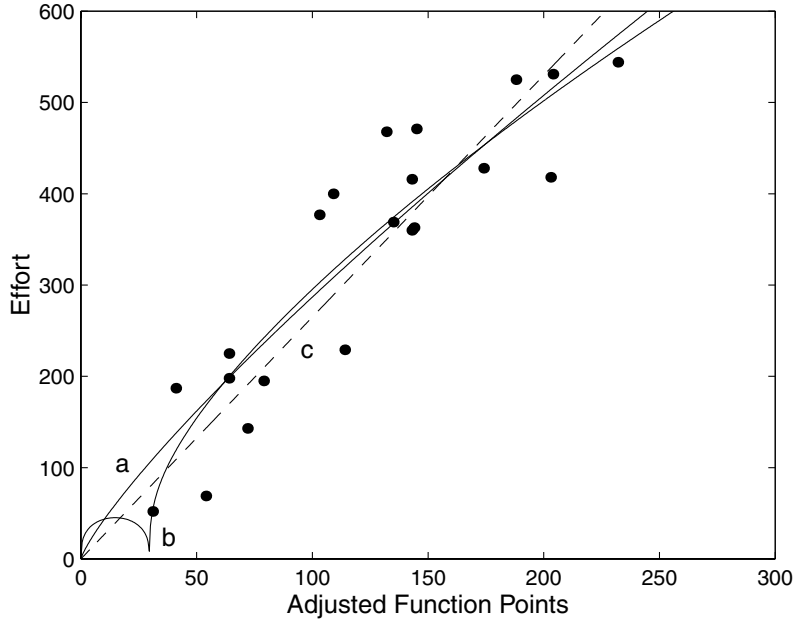

Fig. 2. Some curves in the data set of Abran and Robillard. *a* is the GP derived equation; *b* is also derived by GP; *c* is the linear equation.

different functions, by minimizing the mse, the assumptions under which the functions should be built play an important role. For instance, for data set A (Fig. 2), curve *b* obtains a slighter improvement (PRED(0.25) = 81.8% and MMRE = 0.211) than curve *a*, but at the cost of having a shape which is unable to represent any reasonable hypothesis of the effort–size relationship. Moreover, the strange behavior of this and other curves may go undetected by the usual residual analysis, since the curve merely tries to approximate the points with no concern for the shape in the rest of the *x*–*y* range. In data set A, the procedures for curve estimation have found the same predictive values for the linear equation -reported- and for the log–log equation; however, GP finds a power equation which improves the results. The same situation occurs in other data sets, in which other curves fit the data by minimizing the mse, but with no economic plausibility. The values and functions reported for GP in Table 1 have the property of being

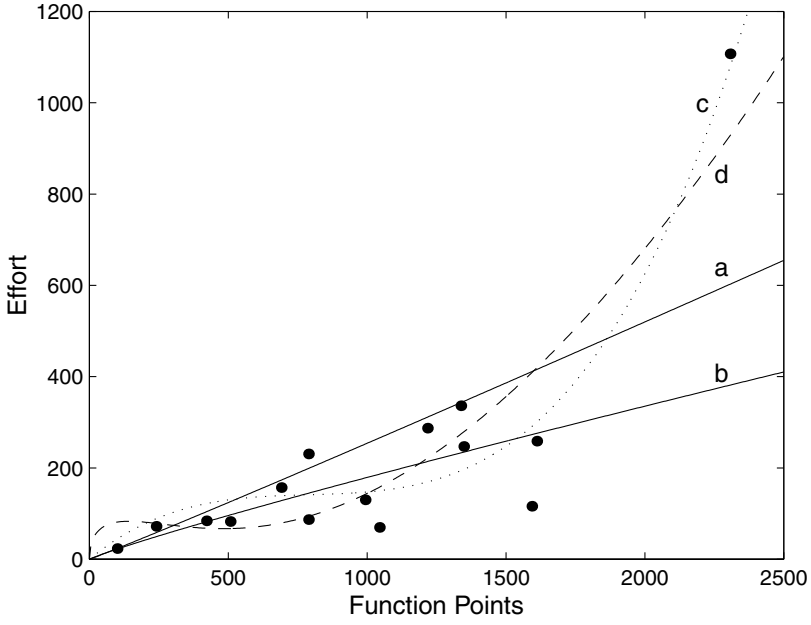

Fig. 3. Curves approximating the Kemerer's data set. Line *a* obtains PRED(0.25) = 62.5% and MMRE = 0.584 (by GP). Line *b* is the power equation. Lines *c* and *d* (by GP) provide slightly better values than *a* or *b*.

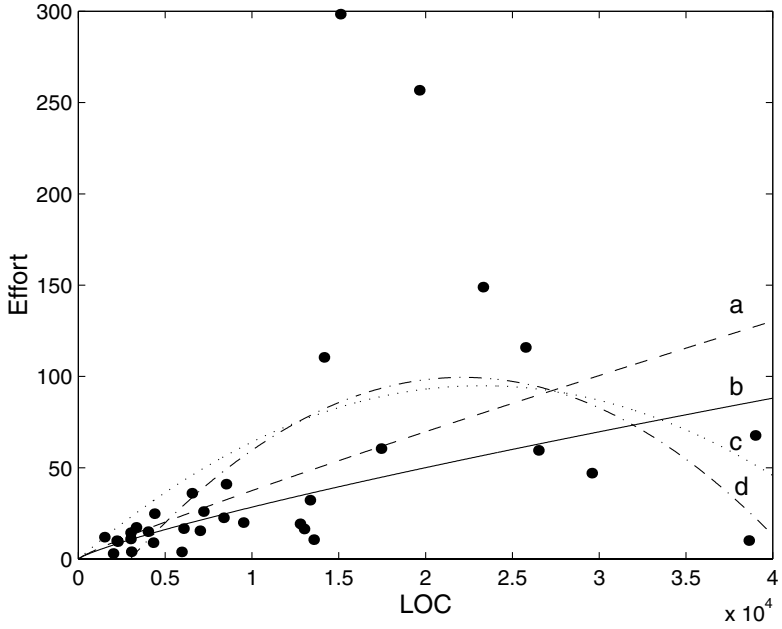

Fig. 4. Kitchenham and Taylor's data set. *a* is the GP derived equation; *b* is the power equation; *c* is the quadratic equation, crossing the origin; and *d* is the quadratic equation of Table 4 in Ref. [6].

smooth, although GP obtained insignificant gains in the values in some specific cases. In all cases, the behavior is increasing, meaning that as the size increases so does the effort.

In the case of Kemerer's data set H — plotted in Fig. 3 —, the hypothesis under which curve *d* (PRED(0.25) = 62.5% and MMRE = 0.584) and cubic *c* (PRED(0.25) = 40%, MMRE = 0.4873; the quadratic behaves similarly) show decreasing cost and economies of scale in the intermediate zone should be justified. The same situation occurs in data set I, where it is difficult to choose, with statistical criteria, between quadratic or cubic functions and other types of functions, unless other assumptions are made about the data. In addition, a similar situation occurs in the data set J where one could be tempted to choose the cubic function (PRED(0.25) = 50% and MMRE = 0.6118). However, in this case it obtains slightly worse results than the power or the GP equations, also dismissing an unreasonable behavior (similar to curves *c* and *d* of Fig. 4). In all these cases, the marginal cost will point out the linear model as the most plausible one.

Table 1 shows that data set F obtains surprisingly good predictions; data sets A and C acceptable predictions; B, H, J, K moderately good; and D, E, G, I, L obtain very bad or moderately bad results. In cases D and E, the use of other cost drivers applied to the size can produce moderately good values (see Appendix B). Size measures embedding some type of calculation in the final count, such as function points, do not show better values than other, less elaborated measures, such as LOC or REIO. We have tested for data set J other alternative models based on number of changes and files, with the data provided in Ref. [29], under the hypothesis that different cost drivers could be used for each activity.

It is sometimes difficult to choose among functions, as Matson et al. pointed out when analyzing the data set B, because the statistical parameters do not provide enough criteria for discrimination. In that case (Fig. 3 in Ref. [26]) there were two possible functions which appeared equally to fit the data. Therefore, the final criterion for selecting a function when the data available is deficient is left to the software manager, who is responsible for discarding unrealistic behaviors. In other situations the underlying trend of the marginal cost can be hidden by a misleading equation; this happens in the same article of Ref. [26], where the published log–log equation, for the data set of 104 points, is $\ln effort = 2.51 + 1.00 \times \ln fp$. However, undoing the transformation, we obtain $effort = 12.3049 \times fp^{1.00}$, that is simply a linear equation (the transformation applied is $\ln y = \ln b_0 + b_1 \ln t \Rightarrow y = b_0 t^{b_1}$). The predictive values reported were MMRE = 0.87 and PRED (0.25) = 64%. However, the value of PRED(0.25) seems to be computed on the transformed data and not on the original variables (it is quite strange to find a data set with a MMRE above 0.8 with such level of prediction).

Now, let us discuss the results obtained in the data set of Kitchenham and Taylor (L), which has been the subject of controversy in Refs. [6,20]. These two works have argued that a quadratic function is the best curve that fits the data, against the pure line of regression. The first work [6] also found that, for eleven data sets, the coefficient y_0 in the quadratic equation $y_0 + y_1 x + y_2 x^2$, was different from 0 in the 11 data sets and y_2 was different from 0 in 6 of the 11 data sets. Eliminating some outliers with the methods

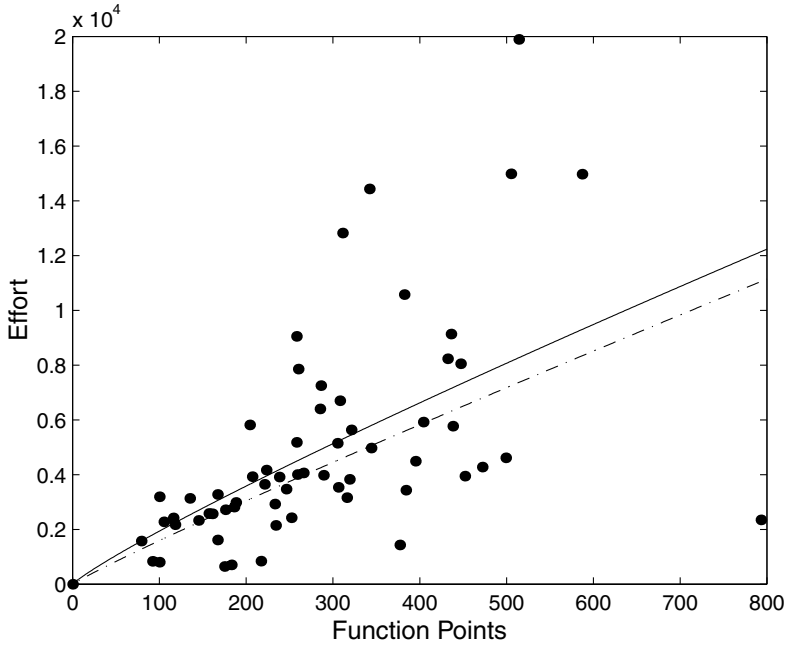

Fig. 5. Desharnais' data set. The solid line was obtained by genetic programming and the dot-dashed line is the power equation.

described in (Ref. [6], pp. 278) the linearity assumption was also rejected for modeling the size–effort relationship. All models use a constant (independent term) in the equation. The evaluation of the goodness of fit is done according to the values of the R^2. Hu rejected the use of the linear model based on the use of the *P*-test for determining the truth of alternative software production functions.

However, assuming that the (0,0) is the origin of the effort–size curves for the data set of Kitchenham and Taylor, the predictive capabilities of the different models are PRED(0.25) = 9.09% and MMRE = 1.7635 for the quadratic function; PRED(0.25) = 21.21% and MMRE = 1.1077 for the cubic; PRED(0.25) = 27.27% and MMRE = 0.8458 for the log–log; and PRED (0.25) = 21.21% and MMRE = 1.0922 for the linear regression (see Fig. 4). Therefore the log–log model obtains the best predictions (the power curve is $0.015474 \times \mathrm{LOC}^{0.816038}$), followed by the linear model. The cubic model obtains better results than the quadratic function. GP obtains PRED(0.25) = 32.4% and MMRE = 1.143 for the best-fitted equation. Kitchenham and Taylor concluded that the values in the log–log equation were not very different of having a linear equation. The line is marked as *b* in Fig. 4, and as we will show below, the

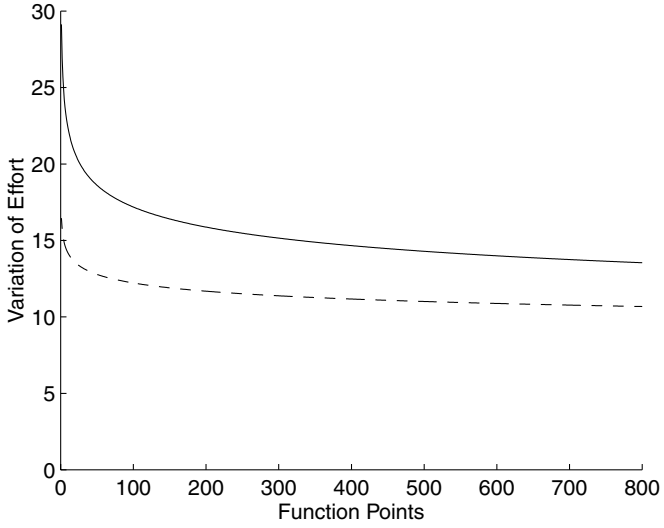

Fig. 6. Marginal cost in Desharnais' data set. The solid line is the variation of the GP curve and the dashed line is the variation of the log–log curve. Both lines show an insignificant variation for small projects, and they are constant in the rest.

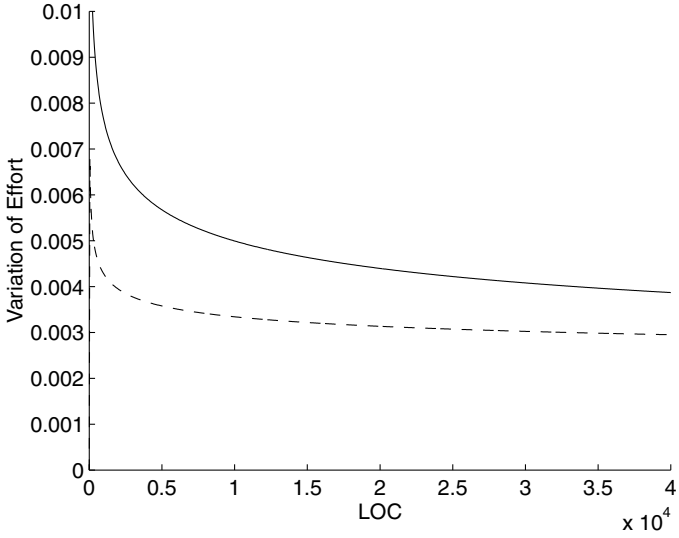

Fig. 7. Marginal costs in the Kitchenham and Taylor's data set.

marginal variation is so low that it is difficult to distinguish it from a pure linear equation. Besides the bad predictive values of the quadratic functions, the shape of lines *c* and *d* in Fig. 4 needs more justification from the managerial point of view than lines *a* or *b*. This is an indication, from our point of view, that the linear model stands out of the other candidates.

The data set of Desharnais has also been used for used for analyzing the effort–size relationship, using function points as the independent variable. We have analyzed different subsets of the data, obtaining the log–log equation as the best curve. For the whole data set the best curve obtained is $21.467186 \times fp^{0.935265}$, with PRED(0.25) = 44.26% and MMRE = 0.5428 (see Fig. 5). The remaining curves obtain worse predictive values, and the curves of marginal cost show how it is difficult to identify economies or diseconomies of scale. Again, the marginal cost analysis will tell us that the linear model is the most plausible one. All the models which Desharnais proposed had non-zero intercepts in the equations, therefore rendering comparison between equations difficult.

Besides the interest in the capacity to make predictions, the question of the type of economy of scale is of paramount importance. The *marginal cost* provides the project manager with information about the parameters that minimize the cost with respect the independent variable (size in this work).

The data sets analyzed in this work are varied in size and in the independent variables, what makes the comparison of the marginal costs more difficult. Table 1 summarizes the type of economy of scale that seems to prevail in each data set, obtained by analyzing the marginal cost of the equations providing the best predictive values (derivatives made with Mathematica 3.0). As indicated in Table 1, except for data set C, it is quite difficult to assert a definitive type of behavior distinct from the linear case, based on the marginal cost. Figs. 6 and 7 are examples of this situation, since it can be observed that the variation of the marginal cost is insignificant, relative to the values of the independent variable size, and it is almost constant. The word "unclear" in Table 1 means that the examination of the variations do not provide values high enough to characterize the trend as economies or diseconomies of scale, because the derivatives do not significantly deviate from the linear case. The controversial data set L behaves almost linearly in both the power equation and in the GP equation (see Fig. 7), as already reported in ref. [23]. Data set D has insignificant diseconomies of scale below 200,000 LOC, and minimal economies of scale above that limit. However, the deviations from the linear trend are almost imperceptible. The data set C is the only one that shows a return to scale clearly identifiable (economies of scale), but its magnitude is so low that its usefulness to the software manager is questionable.

We observe, as a general conclusion, that the independent variable size, does not define the type of economies of scale or marginal return with clarity. In most cases, the marginal return is so low that asserting that economies or diseconomies of scale exist is very questionable. From the point of view of the capabilities of the two methods, GP achieves better values in the PRED(0.25) in eleven out of the twelve data sets, but sometimes at the cost of having a slight worse value of the MMRE. Only in data sets A and H, GP provides a significant improvement over classical regression

5. Conclusions

The collection of data for project estimation usually focuses on measuring effort and size as the key variables, and later on adjusting the basic relationship with several cost drivers. This method of making estimates can provide only moderately good results, as we have seen above. The

data collecting process is rooted in the manufacturing paradigm, which considers that costs depend on the volume of production. The manufacturing paradigm is probably not the most adequate metaphor to portray software development. The results based on this paradigm cannot ensure a specific form of the cost function.

Minor variations of the different criteria of the goodness of fit hide the true problem of the size–cost relationship. The construction of a predictive theory for effort estimation requires the gathering of empirical facts which support the theory. As stated in the results of Table 1, size is a cost factor but, as presently modeled, its predictions are not reaching levels comparable to the know-how of project managers in some cases. Genetic programming allows the exploration of a vast space of cost functions, according to the criteria set in the fitness function, providing confidence in the limits of prediction by symbolic cost functions.

The assumptions under which the models are built, must be included within the scientific endeavors in software management research. The development of a theory of the behavior of the software costs should be based not only on different statistical tests, but also on a set of assumptions which can appropriately describe the phenomenon under study. The development of scientific theories usually requires frameworks or paradigms that can accommodate all the empirical facts related to the object of research. In the software engineering field the principles of software measurement have only recently been stated, and they must be applied to every measurement task (e.g. Ref. [16]). However, the need for a specific paradigm that embraces all the aspects of software estimation in a unified form still exists.

It is relevant to quote here some phrases of Simon related to the process of discovery, since some of the estimation equations proposed and accepted in other works depend on the assumptions used: "… we seem to devote much more time to seeking out possible regularities in phenomena than simply to proving that regularities we have noted are really there, and are not products of our imaginations" [30], pp. xvi. In fact, as the above results have shown, we need a theory (or different theories) which explains the different results obtained with empirical data. The present state of the art in software estimation does not provide a theory that explains all the data available. Consequently the final scientific goal is "…to discover candidate theories that might help explain the facts" [30], pp. xvii. The next step will be to test those alternative theories, perhaps by using more knowledge than that which is provided by statistical analysis [31]. As an example, we can adjust quadratic functions, not monotonically increasing, to some data sets, obtaining moderate predictive values but with no other rationale for that behavior apart from specific statistical tests.

Based on the predictive capabilities of the models that we have built in Section 4, we have not found a reasonable level of good predictions attained solely by the independent variable 'size'. In most cases the economies or diseconomies of scale have minor variations in the marginal cost, therefore characterizing the software costs simply as an increasing linear function of the size of the project.

The linear model has consistently obtained the best predictive values regardless of the method used, even when the GP method is able to explore a vast space of equations. The main consequence of this result is that the project manager can explain in a straightforward way the fundamental trend of the project costs and he/she may focus the attention in identifying other factors that influence the costs.

Acknowledgements

The analysis by Genetic Programming has been possible thanks to the software of the Newcastle Symbolic Optimization Research Group of the University of Newcastle-Upon-Tyne, to whom the author is very grateful. The comments of two anonymous referees have greatly improved the readability of the article. This work was supported in part by CICYT, under projects CICYT TIC98-1179-E, CICYT TIC99-0351 and by UPV-EHU 141.226-EA083/98.

Appendix A. Measures for model evaluation

Let e_i be the actual value of a variable, $\hat{e}_i$ its corresponding estimate and $\bar{e}$ the mean of the values. Let k, n be the number of independent variables and the number of observations (projects), respectively. The following variables are defined

- *Mean magnitude of relative error* (MMRE) is defined as

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i-\hat{e}_i}{e_i}\right|.$$

  Thus if the MMRE is small, then we have a good set of predictions. A usual criterion for accepting a model as good is that the model has a MMRE ≤ 0.25.
- *Prediction at Level l* (PRED(l)), where l is a percentage, is defined as the quotient of number of cases in which the estimates are within the l absolute limit of the actual values, divided by the total number of cases. For example, PRED(0.1) = 90 means that 90% of the cases have estimates within the 10% range of their actual values. A standard criterion for considering a model as acceptable is PRED(0.25) $\geq$ 75.
- *Coefficient of multiple determination,* R^2 and *adjusted* R^2, are some usual measures in regression analysis, denoting the percentage of variance accounted for by the

independent variables used in the regression equations

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (e_i - \hat{e}_i)^2}{\sum_{i=1}^{n} (e_i - \bar{e})^2}$$

and $R^2_{\text{adjusted}} = R^2 - \frac{k-1}{n-k} \times (1 - R^2)$.

Appendix B. Data sets

Twelve data sets (some of them well known) have been analyzed. The independent variable in all cases is a measure of the size of the product being developed. It takes several forms such as LOC, adjusted delivered source instructions, function points (fp) and others. The dependent variable is the development cost, and it accounts for the human effort consumed in the project.

A. Data set from Abran and Robillard [2] (21 points). It is a subset from a total of 36 projects, from which the points considered as outliers were removed. The independent variable used is "adjusted function points", and the effort is measured in "person-days".

B. Data set from Albrecht and Gaffney [1] (24 points). This data set corresponds to projects developed at IBM Data Processing Services, and Matson et al. [26] and Shepperd and Schofield [29] have analyzed it. The latter study shows values of PRED(0.25) = 33% and MMRE = 0.62 when using *estimation by analogy* and PRED(0.25) = 33% and MMRE = 0.9 when using linear regression.

C. Data set from Bailey and Basili [4] (18 data points). The authors tested several models, including the linear, exponential and log–log. The base models were adjusted by means of multipliers. They concluded that no model could stand above the others. In the present work the multipliers are not used, the independent variable used is "thousand of total lines of code" and the effort is measured in man-months.

D. Data set from Belady and Lehman [7] (33 data points). The independent variable is LOC and the effort is measured in man-months. The generalized COPMO model [11] applied to this data set gave values of PRED(0.25) = 64% and MMRE = 0.27.

E. Data set from Boehm [9] (63 points). It is one of the most analyzed data sets. The independent variable used is "adjusted delivered source instructions", which takes into account the variation of effort when adapting software. The COCOMO model is built upon these data points, by introducing many factors in the form of multipliers. An independent evaluation made by Conte et al. [11] provides the following values of the evaluation of the model: Basic COCOMO gives PRED(0.25) = 27% and MMRE = 0.6. Intermediate COCOMO (with 16 cost drivers) gives PRED(0.25) = 76% and MMRE = 0.19.

F. Data set from Heiat and Heiat [18] (35 data points). They proposed a model to estimate effort (in person-hours) for small-scale projects by using as independent variable the REIO, with *REIO = total number of corrected relationships at data stores + total number of data flows which connect the system to external entities* (see the reference for details of how to construct this measure). The authors also provide the LOC and corresponding function points, but in order to see how another measure of the size product works, the REIO is used here as the independent variable.

G. Data set from an academic environment (48 data points). This data set is a combination of the data provided in Ref. [13] and other projects for which the data effort-LOC 4GL was collected (data available from the author). In this environment, the function point count was found to be unrelated to effort. Therefore, here the LOC 4GL measure is used as independent variable and the dependent variable is effort measured in person-hours.

H. Data set from Kemerer [22] (15 data points). It was used to test different estimation methods. The independent variable used is function points. Kemerer's data obtains PRED(0.25) = 40% and MMRE = 0.62 when using *estimation by analogy* and PRED(0.25) = 13% and MMRE = 1.07 when using linear regression [29].

I. Data set from Miyazaki et al. [28] (47 data points). It was used to test the "robust regression" method. Using three independent variables and the method of "least-squares of inverted balanced relative errors" the result was PRED(0.25) = 29.2% and PRED(0.25) = 18.8% when using the least squares method. The independent variable in the present work is KLOC (thousands of LOC).

J. Data set from Shepperd and Schofield [29] (18 data points). This data set has been used to test the method of estimation by analogy. The independent variable is "number of files". The results of adjusting a regression line are PRED(0.25) = 44% and MMRE = 0.86, and estimating by analogy the values obtained were PRED(0.25) = 44% and MMRE = 0.39.

K. Data set from Desharnais [12] (61 data points). This data set relates function points to effort, using the concepts of 'entity' and 'transaction' for identifying the function points.

L. Data set from Kitchenham and Taylor [23] (33 data points). This data set is composed of 33 projects developed in the same language (S3, a high level language). The data relates LOC to effort (man-months). It has been the subject of the controversy about the existence of economies of scale.

References

[1] A.J. Albrecht, J.R. Gaffney, Software function, source lines of code, and development effort prediction: a software science validation, IEEE Transactions on Software Engineering 9 (6) (1983) 639–648.

[2] A. Abran, P.N. Robillard, Function point analysis: an empirical study of its measurement processes, IEEE Transactions on Software Engineering 12 (12) (1996) 895–910.

[3] T. Bäck, U. Hammel, H-P. Schwefel, Evolutionary Computation: Comments on the History and Current State, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 3–17.
[4] J.W. Bailey, V.R. Basili, A meta-model for software for software development resource expenditures. Proceedings of the Fifth International Conference on Software Engineering, 1981, pp. 107–116.
[5] R.D. Banker, C.F. Kemerer, Scale economies in new software development, IEEE Transactions on Software Engineering 15 (10) (1989) 1199–1205.
[6] R.D. Banker, H. Chang, C.F. Kemerer, Evidence on economies of scale in software development, Information and Software Technology 36 (5) (1994) 275–282.
[7] L.A. Belady, M.M. Lehman, The characteristics of large systems, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, Cambridge, MA, 1979, pp. 106–138.
[8] J.M. Bieman, N. Fenton, D.A. Gustafson, A. Melton, L.M. Ott, Fundamental issues in software measurement, in: A. Melton (Ed.), Software Measurement, International Thompson Computer Press, 1995, pp. 39–52.
[9] B.W. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[10] P.E. Byrnes, T.P. Frazier, T.R. Gulledge, Returns-to-scale in software production: a comparison of approaches, in: T.R. Gulledge, W.P. Hutzler (Eds.), Analytical Methods in Software Engineering Economics, Springer, Berlin, 1993, pp. 75–98.
[11] S.D. Conte, H.E. Dunsmore, V.Y. Shen, Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park, CA, 1986.
[12] J.-M. Desharnais, Analyse statistique de la productivité des projects de dévelopment en informatique à partir de la technique des points de fonction, Masters Thesis, Univ. du Québec à Montreal, Décembre, 1988.
[13] J.J. Dolado, A study of the relationships among Albrecht and Mark II function points, lines of Code 4GL and effort, Journal of Systems and Software 37 (2) (1997) 161–173.
[14] J.J. Dolado, L. Fernández, Genetic programming, neural networks and linear regression in software project estimation, in: INSPIRE III, Process Improvement through Training and Education, C. Hawkins, M. Ross, G. Staples, J.B. Thompson (Eds.), The British Computer Society, 1998, pp. 157–171.
[15] J.J. Dolado, A validation of the component-based method for software size estimation, IEEE Transations on Software Engineering to appear.
[16] N. Fenton, Software measurement: a necessary scientific basis, IEEE Transactions on Software Engineering 20 (3) (1994) 199–206.
[17] A.R. Gray, S.G. MacDonell, A comparison of techniques for developing predictive models of software metrics, Information and Software Technology 39 (1997) 425–437.
[18] A. Heiat, N. Heiat, A model for estimating efforts required for developing small-scale business applications, Journal of Systems and Software 39 (1) (1997) 7–14.
[19] C.T. Horngreen, G. Foster, S. Datar, Cost Accounting. A Managerial Emphasis, 8th ed., Prentice-Hall, Englewood Cliffs, NJ, 1994.
[20] Q. Hu, Evaluating alternative software production functions, IEEE Transactions on Software Engineering 23 (6) (1997) 379–387.
[21] Q. Hu, R.T. Plant, D.B. Hertz, Software cost estimation using economic production models, Journal of Management Information Systems 15 (1) (1998) 143–163.
[22] C.F. Kemerer, An Empirical Validation of Software Cost Estimation Models, Communications of the Association for Computing Machinery 30 (5) (1987) 416–429.
[23] B.A. Kitchenham, N.R. Taylor, Software project development cost estimation, Journal of Systems and Software 5 (1985) 267–278.
[24] B.A. Kitchenham, Empirical studies of assumptions that underlie software cost-estimation models, Information and Software Technology 34 (4) (1992) 211–218.
[25] J.R. Koza, Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press, Cambridge, MA, 1992.
[26] J.E. Matson, B.E. Barret, J.M. Mellichamp, Software development cost estimation using function points, IEEE Transactions on Software Engineering 20 (4) (1994) 275–287.
[27] B. McKay, M.J. Willis, G.W. Barton, Steady-state modelling of chemical process systems using genetic programming, Computers and Chemical Engineering 21 (9) (1997) 981–996.
[28] Y. Miyazaki, M. Terakado, K. Ozaki, H. Nozaki, Robust regression for developing software estimation models, Journal of Systems and Software 27 (1) (1994) 3–16.
[29] M. Shepperd, C. Schofield, Estimating software project effort using analogies, IEEE Transactions on Software Engineering 23 (11) (1997) 736–743.
[30] H.A. Simon, Models of Discovery, Reidel, Boston, 1977.
[31] H.A. Simon, On judging the plausibility of theories, Models of Discovery, Reidel, Boston, 1977.
[32] M. Willis, H. Hiden, M. Hinchliffe, B. McKay, G. Barton, Systems modelling using genetic programming, Computers and Chemical Engineering 21 (1997) 1161–1166.