Cuadrado-Gallego JJ, Rodríguez D, Sicilia MA *et al.* Software project effort estimation based on multiple parametric models generated through data clustering. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 22(3): 371~378 May 2007

Software Project Effort Estimation Based on Multiple Parametric Models Generated Through Data Clustering

Juan J. Cuadrado Gallego[1], Daniel Rodríguez[1], Miguel Ángel Sicilia[1], Miguel Garre Rubio[1] and Angel García Crespo[2]

[1] *Department of Computer Science, The University of Alcalá, Alcalá, Spain*

[2] *Department of Computer Science, Carlos III University, Madrid, Spain*

E-mail: {jjcg, daniel.rodriguezg, msicila, miguel.garre}@uah.es; acrespo@ia.uc3m.es

Received March 15, 2006; revised February 15, 2007.

Abstract Parametric software effort estimation models usually consists of only a single mathematical relationship. With the advent of software repositories containing data from heterogeneous projects, these types of models suffer from poor adjustment and predictive accuracy. One possible way to alleviate this problem is the use of a set of mathematical equations obtained through dividing of the historical project datasets according to different parameters into subdatasets called partitions. In turn, partitions are divided into clusters that serve as a tool for more accurate models. In this paper, we describe the process, tool and results of such approach through a case study using a publicly available repository, ISBSG. Results suggest the adequacy of the technique as an extension of existing single-expression models without making the estimation process much more complex that uses a single estimation model. A tool to support the process is also presented.

Keywords software engineering, software measurement, effort estimation, clustering

1 Introduction

Parametric estimation techniques are nowadays widely used to measure and/or estimate the cost associated to software development[1]. The Parametric Estimating Handbook[2] defines parametric estimation as "a technique employing one or more cost estimating relationships and associated mathematical relationships and logic". Parametric techniques are based on identifying variables that obtain numerical estimates from main input variables that are known to affect the effort or time spent in development.

One important aspect of the process of deriving models from databases is that of the heterogeneity of data. A measure of such heterogeneity is heteroscedasticity, i.e., non–uniform variance. It is well-known that heteroscedasticity is a problem affecting data sets that combine data from heterogeneous sources[3]. As a result, when using such software engineering databases, traditional application of regression equations to derive a single mathematical model results in poor adjustment to data and subsequent potential high deviations. This is due to the fact that a single model cannot capture the diversity of distribution of different segments of the database points. As an illustrative example, the straightforward application of a standard least squares regression algorithm to the points used in the reality tool of the ISBSG 8 database distribution results in measures of $MMRE = 2.8$ and $Pred(0.3) = 23\%$ (these measures are introduced later), which are poor figures of predictive quality.

The use of clustering techniques has been described as a solution to provide more realism to parametric models by decomposing the model in a number of sub–models, that are used for project estimation[4] with improved accuracy when compared with single models. The resulting predictive schemes have been called *segmented* models. One of the principal benefits of this kind of techniques is the fact that the search of segmented models satisfying some pre-established quality conditions can be automated through existing clustering methods.

The rest of this paper is structured as follows. Section 2 describes related work in segmented models for software estimation. The process for software project estimation using clustering techniques and associated tool is described in Section 3. Then, Section 4 reports on empirical work performed to validate the process using a publicly available repository. Finally, conclusions and future work are discussed in Section 5.

2 Related Work

Shepperd *et al.*[5] classify estimation and prediction techniques into three main categories: (i) expert judgement; (ii) algorithmic models; and (iii) machine learning. This work focuses on combining clustering as machine learning technique with classical regression models. The use of different clustering approaches has already been applied to several aspects of software management, including software estimation, software quality and software metrics.

Xu and Khoshgoftaar[6] use the fuzzy c-means algorithm for variable, the partitioning of the data into a number of clusters based on experiences.

Pedrycz and Succi[7] also use fuzzy c-means as a tool to derive prototypes related to software code measurements. Dick *et al.*[8] use the same algorithm for a simi-

Regular Paper

This work is supported by the Spanish Ministry of Science and Technology under Grant No. CICYT TIN2004-06689-C03.

lar setting in a knowledge discovery study. Nonetheless, these approaches do not deal with the heterogeneity of the project databases they use.

Lung, Zaman, and Nandi[9] have used the numerical taxonomy method for the clustering of software components at several development phases, but these analysis are driven by the structure of the code, which is rarely available in public historical software project databases.

Genetic algorithms have also been used for the selection of the mathematical model itself[10]. Nonetheless, the approach of the technique discussed here is that of adjusting the divergences in variance to the well-known exponential function used in parametric estimation, and not postulating different models.

Shepperd and Schofield[11] proposed estimation by analogy. In this approach, a searching algorithm finds in a repository a number of similar projects to the one to be estimated based on its known attributes. Then, the means or weighted means of these similar projects can be used as estimate.

Oligny *et al.*[12] approach estimation studies by the partitioning of the project database into "more homogeneous subsets". This study can be considered as supporting evidence for the segmentation approach described in this paper.

In spite of the scattered available evidence regarding the practical usefulness of partitioning historical databases, the use of clustering techniques for the problem described has to date the drawback of not being explicative of the composition of the segments, i.e., many concrete factors regarding characteristics of the development process and context are not considered as determinants of the resulting segments. On the contrary, these techniques apply a "blind" approach to characteristics that may be considered as relevant *a priori*.

In this paper, we describe the use of clustering techniques and a tool as a solution to provide more realism to parametric models by decomposing the model in a number of sub-models, used to estimate points near them.

3 Software Project Estimation Using Clustering Techniques

This section describes the process and techniques to estimate software projects using clustering techniques.

First, there may exist some attributes that allow us to divide the dataset into different datasets based on previous knowledge that facilitate the clustering technique. This process step is called partitioning. Then, each data group formed after the partitioning process is the input for the clustering process and finally, a regression curve for each cluster is calculated. In this work, we used the EM algorithm for obtaining the clusters and different types of regression models calculated for each cluster depending on the quality of the result obtained as explained in the following subsections.

3.1 EM Algorithm for Clustering Software Engineering Datasets

The EM algorithm is used to generate clusters. It generates a Probability Density Function (PDF) as a lineal combination of NC (number of clusters) distribution functions. Such PDF represents all projects used as input for the algorithm.

$$P(x) = \sum_{j=1}^{NC} \pi_j p(x; \Theta_j) \tag{1}$$

where π_j are the *apriori* probabilities of the clusters adding to 1, i.e., $\sum_{j=1}^{NC} \pi_j = 1$. The $P(x)$ denotes an arbitrary PDF and $p(x; \Theta_j)$ is the density function of de-component j. Each cluster is made up of instances belonging to each density function.

Once a the segmentation process is performed, clusters are generated. Then, for each cluster a regression curve is calculated. For example, when only two attributes are used, it is possible to visualise the clusters in two dimensions (Fig.1) and each regression would be represented as: $DV = A_0 \cdot (IV)^{A_1}$, where DV and IV are the dependent and independent variables respectively, and the adjustment parameters are A_0 and A_1. The independent variables could be used for partitioning as well as for clustering.

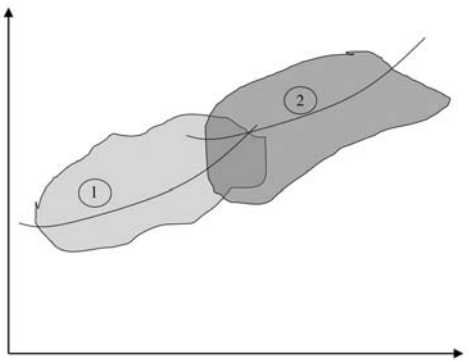

Fig.1. Example of regression curves after the clustering process.

For example, with only two parameters, *effort* as the dependent variable and *size* (function points) as the independent variable, given the size of a new project that we need to estimate, it is necessary to apply the corresponding equation of the appropriate cluster. With just two clusters, there are two intervals $[0, P]$ and (P, ∞); if the number of function points (fp) of the project to estimate is between 0 and P $(0 \leqslant fp \leqslant P)$, we will apply the equation of the first cluster, otherwise $(P < fp \leqslant \infty)$ we will apply the equation of the second cluster.

Fig.2 shows the PDF of the clusters. Assuming that those PDFs follow the normal distribution with means μ_i and standard deviation σ_i $(i = 1, 2)$, it is necessary to calculate D which represents the distance from μ_1 and defines the interval for the first cluster as follows:

$$\frac{D}{\sigma_1} = \frac{\mu_2 - \mu_1}{\sigma_1 + \sigma_2}. \tag{2}$$

Therefore, P defines the intervals $0 \leqslant fp \leqslant P$ and $P < fp \leqslant \infty$ and is obtained as:

$$P = \mu_1 + \frac{1}{1 + \frac{\sigma_2}{\sigma_1}}(\mu_2 - \mu_1). \quad (3)$$

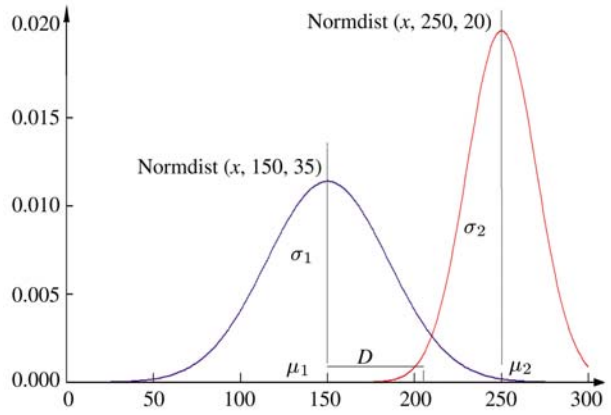

Fig.2. Probability distribution functions for two clusters.

When there are more than one attributes (e.g., function points, team size, etc.) the intervals will be calculated for each of the clusters defining the intersection between them.

3.2 Regression Analysis

After obtaining the clusters using the EM algorithm, it is necessary to calculate a regression equation for each cluster. Regression analysis models relationships between two or more variables, in this paper, the variables to relate are work effort (e) and function points (fp). There are several possibilities when performing regression analysis such as:

Exponential Regression. (4) represents the exponential regression:

$$e = A_0 \cdot fp^{A_1} \quad (4)$$

where e is the dependent variable work effort, A_0 and A_1 are the regression parameters, and fp is the independent variable, *function points*. Also, it is worth noting that the exponential regression must be linearized to simplify the calculation of the exponential function, with a double logarithmic transformation of the effort and function points variables:

$$\ln(e) = \ln(A_0) + A_1 \cdot \ln(fp). \quad (5)$$

Generally (4) is directly solved using complex numeric algorithms such as the Levenberg-Marquardt[13].

Linear Regression. (6) represents linear equation functions:

$$e = A_0 + A_1 \cdot fp. \quad (6)$$

The regression curve of the model is obtained using a set of pairs $(e_1, fp_1), \ldots, (e_n, fp_n)$. The closer the points to the regression curve, the more accurate the regression. According to the *least squares* principle, the curve accuracy depends on the vertical distances (offsets) between the points and the curve. The quality of the fit is measured with the sum of the squares of these deviations so that the best regression curve has the minimum sum of square offsets. Although the least squares method is the most frequently used, outliers affect the accuracy of the regression. There are two possibilities dealing with outliers: (i) it is possible to discard them being this option acceptable when outliers are due to wrongly recorded data or experimental errors; (ii) another possibility is to assign less weight to extreme points than least squares, for example using *least absolute deviation*. In this work, the second approach has been considered. As stated previously, it is necessary to calculate a regression equation for each cluster and for this work, we have implemented exponential or linear regression for each cluster in the following different ways.

Linear (*least-absolute-deviation*). (6) is used to fit the data to the curve and the least absolute deviation is used to minimize the error. This type of regression is appropriate when there is a strong linear dependency between the two variables e and fp and the data points are scattered.

Linear (*least-squares*). The regression has also the form of (6) but the least squares method is used to fit the curve. This method is appropriate when there is a strong linear dependency between the two variables e and fp and the data points are not too scattered.

Linear log-log Plot (*least-absolute*). The form of (4) is used for the regression but it is linearized using double logarithmic transformation. For evaluating the quality of the regression, least absolute deviation is used. It is appropriate when there is no linear dependency between the e and fp variables, and the data points are scattered.

Linear log-log Plot (*least-squares*). As before, it fits the data points to (4) but it uses least squares to minimize the error.

Non-Linear Power (*least-squares*). It also fits the points to the curve using (4) but in this case it does so without any transformation. The quality of fitting is performed using the least squares method. This approach produces similar results to previous ones.

All of these possibilities will allow us to perform different combinations to obtain the best regression for each cluster.

3.3 Validation

Mean Magnitude of Relative Error (MMRE) and Prediction at level p are *de facto* standard measures to measure accuracy for software engineering parametric estimation models $MMRE$ is defined as[14]:

$$MMRE = \frac{1}{n} \sum \left| \frac{e_i - \widehat{e}_i}{e_i} \right| \quad (7)$$

where e_i is the actual value of the variable, $\widehat{e}_i$ its corresponding estimate, and n is the number of observations.

Prediction at level p, $Pred(p)$, where p is a percentage, is defined as the quotient of the number of cases in which the estimates are within the p absolute limit of the actual values, divided by the total number of cases, i.e.,

$$Pred(l) = \frac{k}{N} \tag{8}$$

where k is the number of observations for which $MRE \leqslant l$. For example, $Pred(0.25) = 75$ means that 75% of the cases have estimates within the 25% range of their actual values.

According with Conte[14], the standard criterion for a model to be acceptable is when $Pred(25) \geqslant 0.75$ and $MMRE \leqslant 0.25$, but it is common to relax this constrain for $Pred(\%)$ to $Pred(30)$.

To obtain the input values for $MMRE$ and $Pred$, *cross validation* is used. The *v-fold cross validation* consists of dividing the dataset into v sets of instances, $C_1, \ldots, C_v$ (typically, these will be of roughly the same size). Then, we construct a dataset $D_i = D - C_i$, and test the accuracy of f_{D_i} on the samples in C_i. Having done this for all $1 \leqslant i \leqslant v$ we estimate the accuracy of the method by averaging the accuracy over the v cross-validation trials. Cross-validation has several important properties. First, the training set and the test set in each trial are disjoint. Second, the classifier is tested on each sample exactly once. Finally, the training set for each trial is $(v-1)/v$ of the original data set. Thus, for large v, we get a relatively unbiased estimate of the classifier behavior given a training set of size m[15].

It is possible to select the number of partitions for training and testing. Each training partition will generate different regression curves that will need to be tested with their corresponding testing partitions generating different $MMRE$ and $Pred(0.3)$ values that will need to be averaged for the final result.

3.4 Recursive Clustering Tool (RCT)

As we are using data mining capabilities for software project estimation, we need to provide its users, generally project managers, with the required tool support to hide all the unnecessary complexities for its practical use. Such complexities include data mining algorithms for clustering, and testing techniques for validating models, selection of the cluster for applying the corresponding regression equation, graphical representations, etc. As a result, we have implemented the Recursive Clustering Tool (RCT).

As different organizations can collect different attributes, the RCT tool does not impose variables or attributes for recording project data. Although the tool uses its own format (files with *fclt* extension), the RCT tool can interchange software projects repositories using Witten's *arff* format[16].

Once the project repository has been loaded into the tool, the partitions and clusters are created using the previously mentioned EM algorithm. To run the clustering technique, there is a number of compulsory parameters to set up such independent and dependent variables, and other optional parameters, for example, number of splits for cross-validation, whether to use of Minimum Description Length to calculate the number of clusters, select splits randomly, etc. Once the parameters have been specified, the RCT tool finds clusters and automatically associates the most appropriate type of regression for each cluster. There are multiple possibilities when generating reports from text based to graphical ones. Fig.3 shows a screenshot of the tools output, a graphical representation of the clusters generated (on the left hand side of the figure) and a table with the project repository (on the right hand side of the figure).

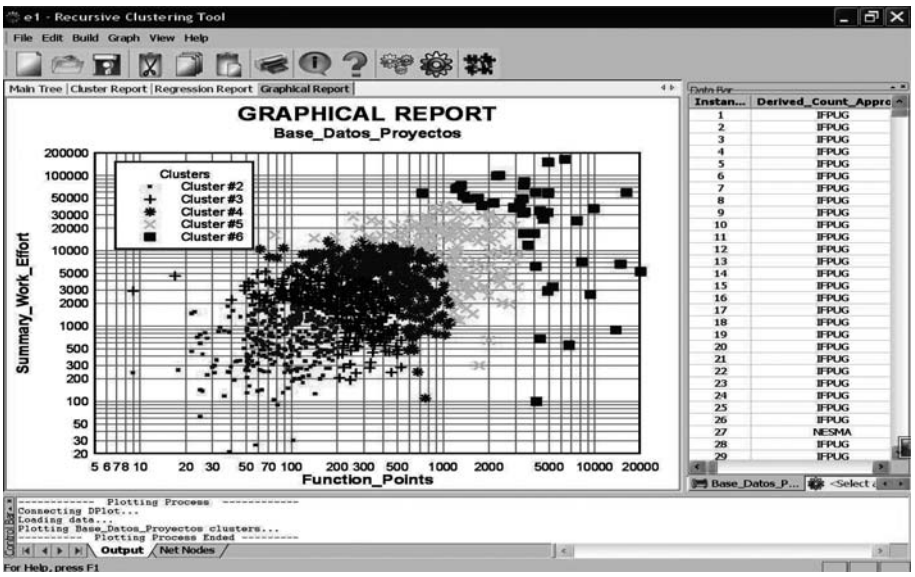

Fig.3. Graphical output of the clusters using the RCT tool.

4 Experimental Work

In this section, we discuss the experimental work performed to validate the process using a publicly available repository, ISBSG, and the results obtained.

4.1 ISBSG Dataset and Preprocessing

In this section, the described technique and tool are applied over the International Software Benchmarking Standards Group (ISBSG), a non-profit organization, that maintains a software project management database from a variety of organizations. In this work, we have used ISBSG release 8, which contains 2028 projects and around 60 attributes per project. The attributes can be classified as follows: (i) project context such as type of organization, business area, and type of development; (ii) product characteristics such as application type user base; (iii) development characteristics such as development platform, languages, tools, etc.; (iv) project size data: different types of function points (IFPUG, COSMIC, etc.); and (v) qualitative factors such as experience, use of methodologies, etc.

As with any data mining technique, it is necessary to preprocess the dataset as inconsistencies and noisy values found in the dataset can confuse the clustering process. Therefore, the first preprocessing step was to remove projects with null or invalid numerical values for the fields effort ("Summary Work Effort" in ISBSG-8) and size ("Function Points"). Then, the projects with "Recording method" for total effort other than "Staff hours" were removed. The rationale for this is that the other methods for recording were considered to be subject to subjectivity. For example, "productive time" is a rather difficult magnitude to assess in an organizational context.

Since size measurements were considered the main driver of project effort, the database was further cleaned for homogeneity in such aspect. Concretely, the projects that used other size estimating method ("Derived count approach") than IFPUG, NESMA, Albretch or Dreger were removed, since they represented smaller portions of the database. The differences between IFPUG and NESMA methods are considered to have a negligible impact on the results of function point counts[17]. Counts based on Albretch techniques were not removed since in fact IFPUG is a revision of these techniques, similarly, the Dreger method refers to the book [18], which is simply a guide to IFPUG counts.

Instance Removal. The first cleaning step consisted of removing projects recoded with low quality. In ISBSG the attribute "Data Quality Rating" catalogs the quality of the data recorded for each project from "*A*" to "*D*"; according to the description of the attribute, only those projects catalogued as "*A*" or "*B*" have been recorded consistently. Next, only those projects whit the value *Staff Hours* for the attribute "Recording Method" were considered. In this dataset, only this value is reliable, the rest of the values for this attribute were subjective ways of recording the effort; for example, the value *Productive Time* for this attribute can be difficult to measure across organizations. Another important aspect is that the database contains different types of function points as COSMIC or IFPUG. Also, there are numerous variations of IFPUG. In this work, only IFPUG and some of its variations were considered. For example, differences between IFPUG and NESMA are insignificant[17], Albretch is the precursor of IFPUG and Dreger[18] is the author of an IFPUG guide. All these values in the attribute "Derived count approach" were considered as IFPUG.

Attribute Removal. The removal of unnecessary attributes, also known as feature selection, is important as attributes can generate less accurate and more complex models. Furthermore, the clustering algorithm can be executed faster.

Data Consistency. It is also necessary to check the attributes to unify their values. For example, the programming language used sometimes can appear as COBOL 2 and other times as COBOL II; there are attributes that contain the value "don't know" and others appear as empty spaces. Once the preprocess has been performed, the reduced database consists of 1546 projects.

Partitioning. For this work, we partitioned the dataset according to whether CASE tools were used (CASET attribute) and whether a methodology was applied during the project (METHO attribute). Records with blanks or unknown values for these two attributes were ignored so that a total of 440 projects remained in the repository which, in turn, was also divided into training with 350 projects (80% approximately) and the remaining 90 projects are used testing, i.e., to calculate *MMRE* and *Pred*(%). Fig.4 shows the partitions generated for the training dataset ("CASET=yes/METHO=yes", partition#1 with a total of 35 projects, "yes/no", partition#2 with 235 projects, "no/yes", partition#3 with 8 projects, and finally "no/no", partition#4 with 72 projects).

4.2 Clustering and Regression

For each partition in the training dataset, a clustering process is performed using the EM algorithm. Fig.5 shows the clusters obtained for partition #1 (CASET=yes and METHO=yes) graphically. Also, Table 1 shows descriptive statistics for all the clusters generated in each partition.

Once the clusters are obtained, a linear or exponential regression using the different types described in Subsection 3.2 needs to be calculated for each cluster. Table 2 shows the different regression curves for each cluster calculated using the RCT tool. Table 2 also shows local *MMRE* and *Pred* prediction values for each cluster, which can help to check whether the number of clusters generated is appropriate or whether there are large disc-

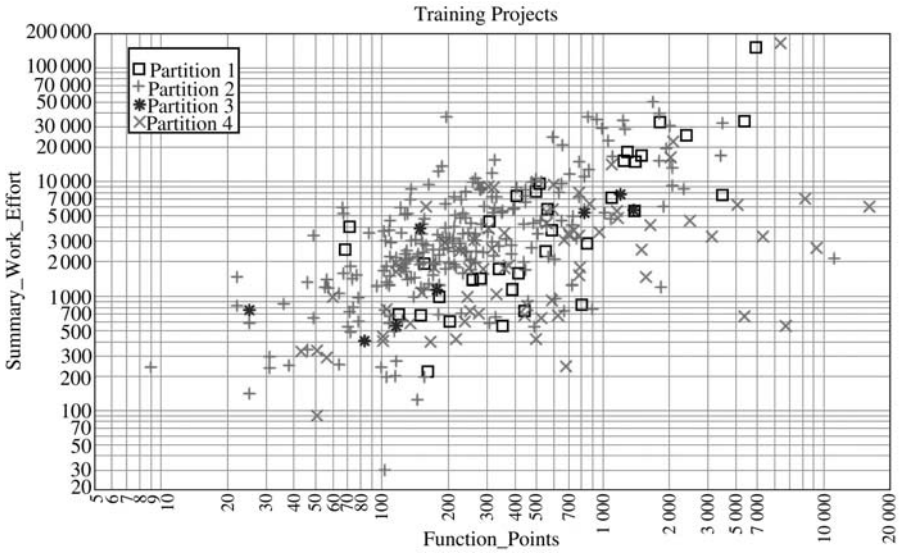

Fig.4. Partitions of the training dataset using METHO and CASET attributes.

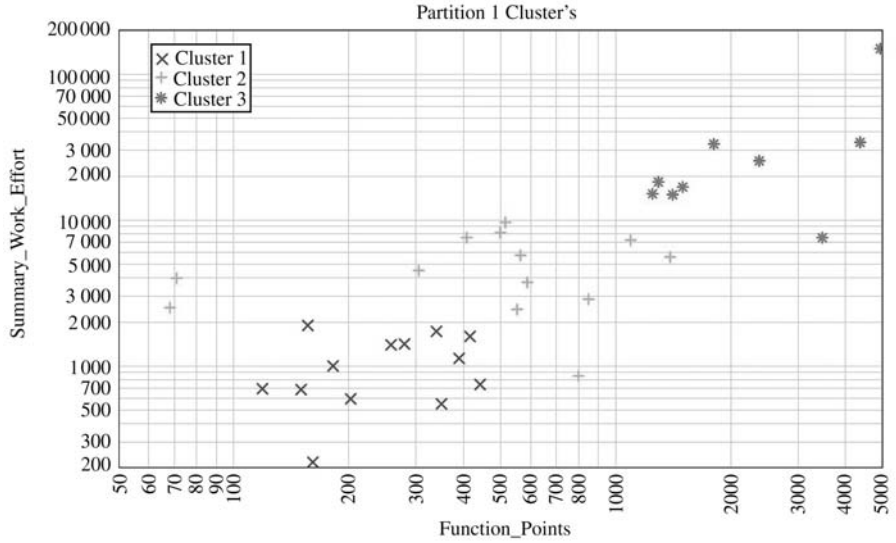

Fig.5. Clusters for Partition #1 (METHO="yes" and CASET="yes").

Table 1. Clusters Generated from Partitions

	Partition #1			Partition #2			Partition #3		Partition #4			
	Cl#1	Cl#2	Cl#3	Cl#4	Cl#5	Cl#6	Cl#7	Cl#8	Cl#9	Cl#10	Cl#11	Cl#12
No. Proj	13	13	9	128	82	25	4	4	17	19	23	13
Prob	0.35	0.38	0.26	0.52	0.39	0.11	0.50	0.50	0.21	0.27	0.33	0.18
Mean(fp)	266.31	564.75	2432.97	143.08	397.47	1905.28	100.00	886.42	285.51	217.90	784.25	5539.70
StdDev(fp)	109.26	351.46	1338.92	69.52	224.74	1997.92	54.64	469.60	169.66	99.05	377.95	3911.26

Table 2. Regression Curves and Prediction Values for Each Cluster

Cluster	Equation	*MMRE*	*Pred*(< 0.3) (%)	Regression Type
Cl#1	$e = 670.28 + 6.31fp$	1.50	35.00	Linear log-log Plot Fitting (least-absolute-deviation)
Cl#2	$e = 4.84fp^{1.04}$	1.52	39.28	Linear log-log Plot Fitting (least-absolute-deviation)
Cl#3	$e = 46.3fp^{0.76}$	1.29	33.33	Linear log-log Plot Fitting (least-absolute-deviation)
Cl#4	$e = 50.16fp^{0.71}$	2.05	22.22	Linear log-log Plot Fitting (least-absolute-deviation)
Cl#5	$e = -246 + 6.87fp$	1.18	26.00	Linear Fitting (least-absolute-deviation)
Cl#6	$e = -3910.53 + 14.18fp$	1.34	26.67	Linear Fitting (least-squares)
Cl#7	$e = 1.35fp1.10$	0.57	25.00	Linear log-log Plot Fitting (least-absolute-deviation)
Cl#8	$e = 42.01fp^{0.61}$	0.57	25.00	Linear log-log Plot Fitting (least-squares)
Cl#9	$e = 36.56fp^{0.81}$	1.47	52.08	Linear log-log Plot Fitting (least-absolute-deviation)
Cl#10	$e = 11.04fp^{0.94}$	1.00	31.67	Linear log-log Plot Fitting (least-squares)
Cl#11	$e = 6.50fp^{1.01}$	0.97	23.34	Linear log-log Plot Fitting (least-squares)
Cl#12	$e = 670.28 + 6.31fp$	1.40	30.76	Linear Fitting (least-squares)

repancies among the different clusters. A global evaluation and comparison with single models is discussed in the next subsection.

4.3 Testing Process

Once we have obtained a regression for each cluster, we need to perform an overall evaluation of the accuracy of these multiple models and its comparison against single models.

The testing process is performed in the same way as the partitioning and clustering process. The testing dataset, which is composed of 90 projects, is divided into four partitions taking into account the METHO and CASET attributes. Then, each partition will be used to test the corresponding clusters, for example, the partition with METHO=yes and CASET=yes will be used to test clusters #1 and #2.

For the single model, only one value of *MMRE* and *Pred*(%) indicates the predicted accuracy of the whole model using single cross validation. However, when multiple models are used, each regression curve has an associated *MMRE* and *Pred* values that needs to be averaged to obtain the accuracy of the whole model.

Table 3 compares the *MMRE* and *Pred* values when a single model is used to the averaged *MMRE* and *Pred* (see Table 2) when multiple models through clustering are used. As it can be observed, both *MMRE* and *Pred* have been significantly improved. The *MMRE* value indicates that the accuracy is doubled and *Pred* is improved in around 10%. Although this improvement is not good enough for software engineering effort estimation standards due to the heterogeneity of the ISBSG repository, the above commented results of this process suggests that this is an valid approach. Furthermore, both prediction values improve, this is not always the case when applying machine learning techniques to software effort estimation.

Table 3. *MMRE* and *Pred* Comparison of a Single Model vs. Multiple Models

	MMRE	*Pred* (< 0.3) (%)
Single Model	2.17	26.75
Using Clustering	1.03	35.60

5 Conclusions and Future Work

This paper presented a process to generate multiple regression models for software effort estimation using clustering. Such a process was validated using a publicly available repository, the International Software Benchmarking Standards Group (ISBSG) database, which provides software management data from multiple organizations. A tool, called RCT, was also presented to facilitate the estimation process using this technique.

The estimation process proposed here consists of the following steps: (i) the project management repository is divided into partitions according to important attributes; (ii) subsequently, clusters are obtained for each partition, in this work, using the EM algorithm; (iii) then, a regression equation is calculated for each cluster; (iv) finally, once we need to perform a new estimate, a regression equation will be selected according to the available data.

From the results and observations performed in the experimental work, it is possible to conclude that the accuracy of the estimates is improved when compared to single models. We believe that this accuracy can be greatly improved if further attributes were taken into account for both the clustering process and regression equations.

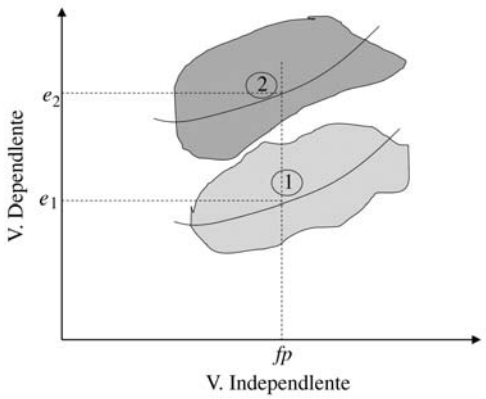

Fig.6. Superimposed clusters.

For future work, we intend to perform further empirical studies taking into account further attributes for clustering and regression. This will not only improve the accuracy of estimates, but also alleviate problems when clusters are superimposed. For example, as shown in Fig.6, when only effort (e) and size (fp) are used, there could be two possible values for the effort (e_1 y e_2). Further work will also compare this technique with other machine learning techniques such as estimation by analogy[11].

References

[1] Boehm B, Abts C, Chulani S. Software development cost estimation approaches — A survey. USC Center for Software Engineering Technical Report USC-CSE-2000-505, 2000.

[2] Parametric Estimating Initiative. Parametric Estimating Handbook, 2nd Edition, 1999.

[3] Stensrud E, Foss T, Kitchenham B, Myrtveit I. An empirical validation of the relationship between the magnitude of relative error and project size. In *Proc. the Eighth IEEE Symp. Software Metrics*, Ottawa, Canada, 2002, pp.3–12.

[4] Cuadrado-Gallego J J, Sicilia M A, Garre M *et al.* An empirical study of process-related attributes in segmented software cost-estimation relationships. *Journal of Systems and Software*, 2006, 79(3): 351∼361.

[5] Shepperd M, Schofield C, Kitchenham B. Effort estimation using analogy. In *Proc. 8th Int. Conf. Software Engineering*, IEEE Computer Society Press, Berlin, 1996, pp.170∼178.

[6] Xu Z, Khoshgoftaar T. Identification of fuzzy models of software cost estimation. *Fuzzy Sets and Systems*, 2004, 145(1): 141∼163.

[7] Pedrycz W, Succi G. Genetic granular classifiers in modeling software quality. *The Journal of Systems and Software*, 2002, 76(3): 277∼285.
[8] Dick S, Meeks A, Last M *et al.* Data mining in software metrics databases. *Fuzzy Sets and Systems*, 2004, 145(1): 81∼110.
[9] Lung C H, Zaman M, Nandi A. Applications of clustering techniques to software partitioning, recovery and restructuring. *Journal of Systems and Software*, 2004, 73(2): 227∼244
[10] Dolado J. On the problem of the software cost function. *Information and Software Technology*, 2001, 43(1): 61∼72.
[11] Shepperd M, Schofield C. Estimating software project effort using analogies. *IEEE Trans. Software Engineering*, 1997, 23(11): 736∼743.
[12] Oligny S, Bourque P, Abran A, Fournier B. Exploring the relation between effort and duration in software engineering project. In *Proc. World Computer Congress*, Beijing, China, August 21∼25, 2000, pp.175∼178.
[13] Marquardt W. An algorithm for least squares estimation of non-linear parameters. *J. Soc. Indust. Appl. Math.*, 1963, 11: 431∼441.
[14] Conte S D, Dunsmore H E, Shen V Y. Software Engineering Metrics and Models. Menlo Park: Benjamin/Cummings, CA, 1986.
[15] Kohavi R, John G. Automatic parameter selection by minimizing estimated error. In *Proc. 12th Int. Conf. Machine Learning*, San Francisco, 1995, pp.304∼312.
[16] Witten I H, Frank E. Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations. San Francisco: Morgan Kaufmann Publishers, USA, 2005.
[17] NESMA. NESMA FPA counting practices manual (CPM 2.0), 1996.
[18] Dreger J B. Function Point Analysis. Englewood Cliffs, NJ: Prentice Hall, 1989.

Juan J. Cuadrado Gallego is currently with the Department of Computer Science at the University of Alcalá, Madrid, Spain and the University Oberta of Catalunya, Barcelona, Spain. He previously hold positions at the University of Valladolid and Carlos III University, Madrid, Spain, where he obtained his doctorate in computer sciences engineering in 2001. His research interests are in the area of software engineering and more specifically in software measurement. He is the president of the Spanish Function Points Users Group (SFPUG).

Daniel Rodríguez is a lecturer in the Department of Computer Science at the University of Reading. He received his degree in computer science from the University of the Basque Country, Spain, in 1995 and his Ph.D. degree from the University of Reading, UK, in 2003. His research interests are primarily in the area of software engineering (SE) including empirical software engineering and the application of data mining to SE.

Miguel Ángel Sicilia obtained an M.Sc. degree in computer science from the Pontifical University of Salamanca, Madrid, Spain in 1996 and a Ph.D. degree from the Carlos III University in 2003. Currently, he leads the Information Engineering Unit at the Computer Science Department, University of Alcalá. His research interests are primarily in the areas of adaptive hypermedia, learning technology and human-computer interaction.

Miguel Garre Rubio received his B.Sc. degree from the University of Murcia, Spain and his doctoral degree in Computer Science from the University of Alcalá, Spain. Currently, he is with the University of Alcalá, Spain with the Open University of Spain. His research interests are in the area of software engineering and software measurement.

Angel Garcia Crepo is a lecturer and subdirector of teaching at Carlos III University, Madrid, Spain. He holds a doctoral degree in industrial engineering by Polytechnic University, Madrid, Spain and an MBA by the IE Business School. As a director of the advanced systems integration team, his research interest include software engineering.