Effort Estimation for Corrective Software Maintenance

Andrea De Lucia
RCOST - Department of Engineering
University of Sannio
Palazzo Bosco Lucarelli, Piazza Roma
82100 Benevento, Italy
delucia@unisannio.it

Eugenio Pompella
EDS Italia Software S.p.A.
Viale Edison - Loc. Lo Uttaro
81100 Caserta, Italy
eugenio.pompella@eds.com

Silvio Stefanucci
RCOST - Department of Engineering
University of Sannio
Palazzo Bosco Lucarelli, Piazza Roma
82100 Benevento, Italy
stefanucci@unisannio.it

ABSTRACT

This paper reports on an empirical study aiming at constructing cost estimation models for corrective maintenance projects. Data available were collected from five maintenance projects currently carried out by a large software enterprise. The resulting models, constructed using multivariate linear regression techniques, allow to estimate the costs of a project conducted according to the adopted maintenance processes. Model performances on future observations were achieved by taking into account different corrective maintenance task typologies, each affecting the effort in a different way, and assessed by means of a cross validation which guarantees a nearly unbiased estimate of the prediction error. The constructed models are currently adopted by the subject company.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Management – *software maintenance, cost estimation.*

General Terms

Management, Measurement, Experimentation.

1. INTRODUCTION

Planning software maintenance work is a key factor for a successful maintenance project. Planning involves estimating size, effort, duration, staff, and costs in order to guarantee the control of the maintenance process and reduce the risks and the inefficiencies related to the maintenance work. To this aim, effort estimation is a valuable asset to maintenance managers in planning maintenance activities and performing cost/benefits analysis. Indeed, early estimates and accurate evaluations permit to significantly reduce project risks and can be useful to support and guide:

- software related decision making;
- maintenance process efficiency and parameters assessment;
- maintain versus buy decisions, comparing productivity and costs among different projects;
- resource and staff allocation, and so on.

Management can use cost estimates to approve or reject a project proposal or to manage the maintenance process more effectively. Furthermore, accurate cost estimates would allow organizations to make more realistic bids on external contracts.

Unfortunately, effort estimation is one of the most relevant problems of the software maintenance process [2, 9, 19, 20]. Predicting software maintenance effort is complicated by the many typical aspects of software and software systems that affect maintenance activities. The maintenance process can be focused on several different types of interventions: correction, adaptation, perfection, etc. [17]. Maintenance projects may range from ordinary projects requiring simple activities of understanding, impact analysis and modifications, to extraordinary projects requiring complex interventions such as encapsulation, reuse, reengineering, migration, and retirement [11]. Moreover, software costs are the result of a large number of parameters [4], so any estimation or control technique must reflect a large number of complex and dynamic factors.

In order to control maintenance costs, quantitative metrics for evaluating and predicting system characteristics must be used. Software project costs are essentially human resource costs and this entails that the effort (i.e. man-days needed for system maintenance) should be maintained under severe control. To this end, a linear or nonlinear relationship between software complexity/size and effort is commonly assumed [26]. The predictor variables typically constitute a measure of size (measured in terms of LOC or a functional size measure) and a number of productivity factors that are collected through a questionnaire [4, 5]. Therefore, the problem of effort evaluation is typically shifted to a complexity or size evaluation process.

This paper presents an empirical study from the experience of a major international software enterprise in conducting corrective maintenance projects. We used multiple linear regression analysis to construct effort estimation models validated against real data collected from five corrective maintenance projects.

The paper is organized as follows. Section 2 discusses related work. Section 3 reports a brief description of the corrective maintenance projects and the metrics adopted for evaluating and predicting the maintenance effort. Section 4 presents the statistical methodology and the experimental results. Discussion and concluding remarks are outlined in Section 5.

2. RELATED WORK

A number of research papers have been published describing the development and evaluation of formal models to assess and predict software maintenance task effort. Effort estimation is a crucial part and a key factor for successful software project planning and management. This is demonstrated by the variety of modeling approaches proposed in the context of software development [4, 38] ranging from decomposition techniques (top down, bottom up), analogy based estimations, algorithmic approaches (statistical, theoretical) to hybrid or mixed approaches, combining algorithmic methods and expert judgments.

Stensrud and Myrtveit [33] demonstrate that human performances are increased when a tool based either on analogy or on regression models is available. However, their work is not generalized to maintenance activities. Software development and software maintenance are two different activities that have different characteristics because the focus of software maintenance is the change of existing software and not the creation of new software. Analogy models [30] are based on historical data and previous project experience: the current software project is compared with one or more previous similar projects carried out in the same organization to relate their costs. They can be viewed as a procedure of encapsulating previous experience to produce a historical maintenance database from which appropriate information are extracted. The tool Angel [30] aims to automate the search for analogies process: similar cases are computed by evaluating their distance in terms of mathematical norms, e.g. Euclidean distance.

Algorithmic approaches involve the construction of mathematical models from empirical data following a well defined procedure. Boehm [4] presents one of the first approaches to estimate maintenance effort: he extends his COCOMO model for development costs to the maintenance phase through a scaling factor. This factor, named annual change of traffic, is an estimate of the size of changes expressed as the fraction of software total LOC which undergo change during a typical year. Many other works [5, 16, 32] extend the COCOMO model introducing new metrics and factors to estimate the development effort.

Lehman and Belady [3] propose a model for estimating the effort required to evolve a system from a release to the next. Their model has two terms accounting respectively for progressive activities (enhancing system functionality) and anti-regressive activities (compensating negative effects of system evolution) [21]. Ramil [28] uses historical data about different versions of the same application to calibrate effort models and assess their predictive power. The best linear regression models extracted are based on coarse-grained metrics (number of added, updated, and deleted sub-systems). Lindvall [22] also shows that coarse-grained metrics, such as the number of classes, have stronger accuracy than other finer-grained metrics on predicting changes. On the other hand, Niessink and van Vliet [27] use regression analysis and a size measure based on Function Points to predict maintenance effort.

Eick *et al.* [13] consider software system evolution to define code decay and a related number of measurements. These indexes are used to produce a cost estimation model quantifying the effort required to implement changes. Jorgensen [19] compares the prediction accuracy of different models using regression, neural networks, and pattern recognition approaches. In particular, the last approach involves the decomposition of a previous experience data set to select those data that are more similar to the current case and are consequently a good starting point for the estimate [6]. All the models assume the existence of a size estimate, measured as sum of added, updated, and deleted LOC during a maintenance task, which can be accurately predicted and sufficiently meaningful. He uses an industrial data set consisting of various maintenance tasks from different applications within the same organization. Other works aim at comparing software cost estimation techniques [7, 15, 18, 31, 37]

Some work has been proposed in the literature to estimate the effort of adaptive maintenance projects and massive maintenance projects. Fioravanti and Nesi [14] presented and evaluated a model and metrics for estimation and prediction of adaptative maintenance effort of object-oriented systems. De Lucia *et al.* [12] present effort estimation models resulting from an empirical study of a large incremental and distributed Y2K projects. The analyzed adaptive massive maintenance project was carried out by the same organization of the present study. Visaggio *et al.* [8] present a method and a tool for the estimation of the effort required in software process execution. They validate it on a legacy system renewal project and show that fine granularity and model recalibration are effective for improving model performance and verify that the estimation model is process dependent.

3. EXPERIMENTAL SETTING

Most estimating techniques rely on historical evidence in past projects. The basic premise is that one can develop accurate quantitative models that predict maintenance effort using historical project data. Even if the new projects will likely use many new elements in combination with the old ones, it is nevertheless important to document the experiences of where the estimates went wrong for future use: the past can be used as a guide to some things that might reasonably happen in the future.

A common practical problem with constructing cost estimation models is that the historical data sets frequently contain considerable numbers of missing values [6, 15, 34]. There are many techniques that have been developed to deal with missing data in the construction of quantitative models (missing data techniques or *MDT*s) [23]. The main advantage of the data set available for our study was that it did not contain missing values. This is due to the careful manner in which the data was collected.

In fact, the subject company has recently achieved the CMM level 3 and is currently planning the assessment to move to the CMM level 4. At CMM level 3, metrics are collected, analyzed, and used to control the process and to make corrections to the predicted costs and schedule, as necessary. Therefore, metric collection was crucial and supported by automatic tools, such as workflow management systems which are an aid to process management automation and improvement [1].

The data set was composed of five corrective software maintenance projects from different companies. The projects were mainly business applications of telecommunication, banking, insurance, and government organizations. This range of projects allows for generalizable conclusions that can be applied to other projects in the business application domain of the subject company.

Most of the business of the subject company concerns maintaining third party legacy systems. The subject company realizes outsourcing of system conduction and maintenance, including help desk services, for several large companies. Very often the customers ask for a very high service agreement level and this requires an accurate choice and allocation of very skilled maintainers, with adequate knowledge of the application domain and technological environment of the maintenance project. This implies the careful definition of the maintenance process with well-defined activities, roles, and responsibilities to avoid inefficiencies. To this aim, the ordinary maintenance process of the subject company closely follows the IEEE Standard for Software Maintenance [17].

The data set was composed of 144 observations, collected from all five projects and corresponding to monthly maintenance periods for each project. For each observation, the following data were available and considered in our analysis (see Table 1):

- size of the system to be maintained;
- effort spent in the maintenance period;
- number of maintenance tasks, split in three categories:
  - ⇒ type A: the maintenance task requires software source code modification;
  - ⇒ type B: the maintenance task requires fixing of data misalignments through database queries;
  - ⇒ type C: the maintenance task requires intervention not comprised in the previous categories, such as user disoperation, problems out of contract, and so on.

Table 2 summarizes the descriptive statistics for system size, number of maintenance tasks, and project effort.

Other technical metrics, such as software complexity metrics, were not available for all the projects. In fact, for each new maintenance project, the subject company preliminarily collects a number of different technical metrics on a meaningful subset (about 20%) of the application portfolio to be maintained. The goal is to make an assessment of the software systems [11], to negotiate the customer service levels, and to select the skills required by the maintenance teams.

Table 1: Collected metrics

Metric	Description
NA	# of tasks requiring software modification
NB	# of tasks requiring fixing of data misalignment
NC	# of other tasks
N	# of tasks (N=NA+NB+NC)
SIZE	Size of the system to be maintained [kLOC]
EFFORT	Actual Effort [man-hours]

Table 2: Descriptive statistics of the data set

Metric	Min	Max	Mean	Median	Std.Dev.
NA	0	120	22.54	14	25.25
NB	0	559	135.16	51	146.23
NC	6	320	80.17	78.5	63.06
N	14	646	237.56	152	186.22
SIZE	293	5045	2630.60	2344	1420.86
EFFORT	243	1960	846.13	760	349.65

4. BUILDING EFFORT ESTIMATION MODELS

We focus on ordinary multivariate least squares regression [24, 35] as the modeling technique for effort estimation since this is one of the most common modeling techniques used in practice [7, 15]. Furthermore, the literature has recently shown evidence that ordinary least squares regression is as good as or better than many competing modeling techniques in terms of prediction accuracy [7, 18].

The ordinary least squares modeling technique minimizes the sum of squared error, whereas we are evaluating the resulting model in terms of the relative error. Miyazaki *et al.* [25] have noted this discrepancy in the software cost estimation literature. One solution to this incongruence is to use modeling techniques that optimize on the relative error. This has rarely been performed in the software engineering literature.

With the aim to obtain a simple and comprehensible model, we evaluated multivariate linear regression models including the number of tasks (total or of different types, as shown in Table 2) and the size of the system. The presence of the system size in the regression equation is consistent with the findings of Niessink and van Vliet [27], who also use regression analysis and a size measure based on Function Points to predict maintenance effort. Their results show that the size of the component to be changed has a larger impact on the effort than the size of the changes. Certainly, it would be very interesting to also consider the size of the maintenance tasks, but this metric is too fine grained for corrective maintenance and was not available.

With the aid of diagnostic plot we verified that the analysis of the residuals does not indicate any non-linearity and the distribution is reasonably normal: there are no particular trends or patterns in the plot of the variables against the residuals.

The correlation matrix (Table 3) shows the absence of strong correlations between the independent variables. The highest correlation value (0,6458) is between the number of tasks of type A and C, while all remaining correlation values between the number of tasks of different type are not meaningful. There is a high correlation between the total number of tasks N and its components NB and NC, while there is no correlation between N and NA. A potential reason for this is that maintenance tasks of types B and C are more recurrent than maintenance tasks of type A (see Table 2).

We first evaluated a linear model including the total number of maintenance tasks N and the size of the system being maintained:

$$\text{Model A:} \quad \mathit{Effort} = a_1 N + a_2\, \mathit{Size}$$

The reason was that this model is rather simple and uses the same variables as the model previously adopted by the software organization, although the latter was not linear.

However, we observed that tasks of type A are less recurrent than the other two task types and argued that generally they require a greater effort to be accomplished. Therefore, we also evaluated the following models:

$$\text{Model B:} \quad \mathit{Effort} = a_1 NA + a_2 NBC + a_3\, \mathit{Size}$$

$$\text{Model C:} \quad \mathit{Effort} = a_1 NA + a_2 NB + a_3 NC + a_4\, \mathit{Size}$$

where NA, NB, NC, and Size are defined as in the previous section and NBC is the sum of NB and NC. It is worth noting that in all the models there is statistical evidence from the *p-value* that an intercept term is not needed. This is not surprising, as such a term would have suggested the existence of an effort type not directly related to the variables being included in the model.

Table 3: Metrics correlation matrix

	N	NA	NB	NC	SIZE
N	1.0000	0.2078	**0.9466**	**0.8340**	0.1010
NA	0.2078	1.0000	0.4002	**0.6458**	0.0759
NB	**0.9466**	0.4002	1.0000	0.0864	0.3079
NC	**0.8340**	**0.6458**	0.0864	1.0000	0.0012
SIZE	0.1010	0.1217	0.2857	0.2652	1.0000

Robust regression techniques [29, 35] were also explored to handle non-obvious outliers. Outliers not only influence the estimation of the regression coefficients, they can also have an even larger effect on standard errors, t-tests, R^2, and other regression statistics. Ordinary least squares analysis does not perform well when outliers occur. We had no evidence of outliers in the data set. We obtained few observations with a weight that can be considered outlier (2 with a weight less than 0.05 and 5 with a weight less than 0.10). However, removing these observations from the data set did not meaningful change the analysis results, so we maintained them in the data set.

4.1 Evaluating Model Performances

In the evaluation of the accuracy performances of the estimation models we applied measures based on two distinct classes of evaluation criteria. The first class is based on statistics developed from the statistical analysis of the variance, such as the coefficient of determination R^2. This represents the percentage of variation in the dependent variable explained by the independent variables of the model. An extension of R^2 is the adjusted R^2 that accounts for the number of independent variables in the regression model.

Table 4 shows the model parameters and R^2 values for all the models. Although the R^2 value is high, it does not assess the quality of future prediction, only the capability of fitting the sample data. If a cost estimation model is developed using a particular data set and the accuracy of this model is evaluated using the same data set, the accuracy obtained will be optimistic. The calculated error will be low and will not represent the performance of the model on another separate data set.

Table 4: Model parameters

Model	Var.	b_i (Coeff.)	p-value	R^2	Adj R^2
A	**N**	1.342904	<10E-07	0.8257	0.8245
	SIZE	0.169086			
B	**NA**	9.053286	<10E-07	0.8891	0.8876
	NBC	0.138275	<10E-07		
	SIZE	1.164826	<10E-07		
C	**NA**	7.86988	<10E-07	0.8963	0.8941
	NB	0.514121	0.032351		
	NC	2.81486	0.000001		
	SIZE	0.130507	<10E-07		

A class of evaluation criteria that assesses the quality of future prediction is based on residual analysis and includes the PRESS (PREdiction Sum of Squares) statistics [35].

The PRESS statistics are based on a systematic examination of the residuals. A residual is the difference between an observed value of the dependent variable *y* and the value predicted by the model. The PRESS statistics are developed from PRESS residuals, $\hat{y}_{i*} - y_i$, where y_i is the value of the *i-th* value of the dependent variable as observed in the data set and $\hat{y}_{i*}$ is the predictive value from a regression equation trained with all the observations except the *i-th*. Thus, in a sample data set of size *n*, there are *n* separate regression equations, each obtained from *n* – 1 observations and tested on the withheld datum[1].

The sum of the squared prediction errors is the PRESS value:

$$PRESS = \sum_i (\hat{y}_{i*} - y_i)^2$$

1 Some authors refer to these approach as leave-one-out cross validation [24].

The smaller the PRESS, the better the predictability of the model. If a large value for the PRESS is due to one or a few large PRESS residuals, the SPR statistic may be a more accurate way to evaluate predictability. *SPR* is the sum of the absolute value of the PRESS residuals or prediction errors:

$$SPR = \sum_i \left| \hat{y}_{i*} - y_i \right|$$

PRESS and SPR measures allow to compare models on the same data set, but cannot be used as absolute indicators. For this reason the following statistics were also computed: MMRE (Mean Magnitude Relative Error) and MdMRE (Median Magnitude Relative Error). These are also calculated from the model developed using the training data set and evaluated on the test data set. The MRE_i (Magnitude of Relative Error) on an observation *i* is defined as:

$$MRE_i = \frac{\left| \hat{y}_{i^*} - y_i \right|}{y_i}$$

where y_i is the value of the i-*th* value of the dependent variable as observed in the data set and $\hat{y}_{i*}$ is the correspondent predictive value from the regression equation. MMRE is the average of the MRE_i. MMRE accuracy measure is a clear and intuitive evaluation criterion, i.e., it is easy to understand and meaningful for the maintenance managers who are the final users of cost estimation models. It is regularly used to compare the accuracy of cost estimation models and modelling techniques [7, 18]. Its shortcoming is that the value may be strongly influenced by a few very high relative error (RE) values. We therefore included the median magnitude of relative error (MdMRE) in the evaluation.

In addition, to evaluate model performances we computed the following variants of the measure PRED [10, 19]:

- $PRED_{25}$ = the % of the observations with RE <= 0.25.
- $PRED_{50}$ = the % of the observations with RE <= 0.50.

The need for a measure like PRED is caused by the large number of maintenance tasks in our data set. We believe that maintenance managers may, in most cases and specially for small maintenance tasks, accept a relative error between the actual and predicted effort of about 50%. Indeed, the literature and the experience show that accurate prediction is difficult: an average error of 100% can be considered "good" and an average error of 32% "outstanding" [36].

Table 5: Model predictive performances

	Model A	Model B	Model C
PRESS	21630440	14163410	13591780
SPR	46602.18	37397	35596.61
MMRE	42.53%	36.40%	32.25%
MdMRE	37.57%	29.16%	25.35%
$PRED_{25}$	31.25%	40.36%	49.31%
$PRED_{50}$	66.75%	74.56%	82.64%

The measures resulting from the leave-one-out cross-validation are shown in Table 5. As expected, it is clearly evident that model C performs better than the other models, although the performances of the other two models can also be considered rather good. In particular, the values for the PRED measures for model C are very promising: it predicts almost 50% of the cases within a relative error less than 25% ($PRED_{25}$) and about 83% of the cases with a relative error less than 50% ($PRED_{50}$). Also, the relative mean error is 32.25% and can be considered outstanding [36].

4.2 Assessing Predictive Performances through Leave-More-Out Cross-Validation

Due to the excellent values for the PRED measures we assessed the predictive capability of model C by performing a leave-more-out cross-validation. We randomly partitioned the data set into a training data set and a test data set. We used the training data set to build the estimation model and the test data set to evaluate the predictive performances of the model. We repeated this procedure for different partitions and averaged the prediction accuracy of the different models. The results are shown in Table 6. Each column is labeled L_X-T_{100-X} and shows the performances of the model obtained by randomly partitioning the data set into a learning data set composed of X% of the observations and a test data set composed of the remaining observations. As expected, the performances of the accuracy measures generally decrease as the size of the learning set decreases, although it should be remarked that the partitions are randomly selected. It is worth noting that in the leave-one-out cross-validation the test set is composed of only one observation and the number of test sets is equals to the number of observations.

Table 6: Leave more out cross validation with random partitions

	Learning % - Test %				
	L_{90}-T_{10}	**L_{80}-T_{20}**	**L_{70}-T_{30}**	**L_{60}-T_{40}**	**L_{50}-T_{50}**
MMRE	21.22	32.73	36.53	63.33	46.24
Max RE	45.73	114.45	104.68	406.26	285.98
MdMRE	19.76	27.77	32.80	43.04	37.37
$PRED_{25}$	57.14	45.00	35.56	26.67	32.47
$PRED_{50}$	100.00	75.00	68.89	63.33	61.23

To verify the existence of particularities due to only one maintenance project that can bias the analysis, we performed a cross-validation by using the observations of a single project as test set and the observations from the remaining projects as the learning set. In this way, the model has no knowledge about the characteristics of the project used as test set. The results are shown in Table 7. The predictive performances are good when using as test set the projects P2, P4, and P5, while problems can be observed when using as test sets the projects P1 and P3. The causes of these problems are not yet completely clear and need more investigations. For project P1, one of the reasons is likely the absence of maintenance tasks of type B whose number is an independent variable of the model; perhaps for projects like this the other two effort estimation models might be more appropriate.

Finally, we investigated whether the performances of the adopted corrective maintenance process improved over the time. In particular, we compared the model trained on the oldest 50% observations against the model trained on the newest 50% observations for all the projects. The results are shown in Table 8. The first and the second columns show the performances of the model when trained and tested on the oldest 50% and on the newest 50% observations of the data set, respectively. It is worth noting that the model performs better on the newest observations in the data set and this can be considered as an index of the process improvement. However, there is not a great improvement as the model trained and tested on the oldest observations is rather good, thus demonstrating a given stability and maturity of the corrective maintenance process of the organization. To confirm this, we tested the model trained on the oldest observations on the newest observations and vice-versa; the results shown in the third and fourth columns of Table 8, respectively, confirm this thought, although the model trained on the newest observations still performs better.

Table 7: Leave more out cross validation with project partitions

	Maintenance Project				
	P1	**P2**	**P3**	**P4**	**P5**
MMRE	64.65	25.60	79.20	31.99	35.17
Max RE	280.96	79.59	382.46	56.59	214.21
MdMRE	50.63	21.48	66.96	33.72	26.33
$PRED_{25}$	33.33	60.61	10.00	33.33	47.22
$PRED_{50}$	46.67	84.85	30.00	86.67	83.33

Table 8: Oldest vs newest observations prediction performances

	L_{Old}-T_{Old}	L_{New}-T_{New}	L_{Old}-T_{New}	L_{New}-T_{Old}
MMRE	35.00	27.32	50.40	36.24
Max RE	205.17	172.24	304.10	235.5
MdMRE	28.34	21.31	34.73	28.65
$PRED_{25}$	46.48	56.34	35.14	38.89
$PRED_{50}$	80.28	84.51	60.81	77.78

5. CONCLUSION

In this paper we have presented an empirical study aiming at building corrective maintenance effort estimation models. A data set obtained from five different corrective maintenance projects was used as case study to experimentally validate and compare model performances through multivariate linear regression models. Our analysis started from using the experience of the subject company on estimation processes and the model adopted for corrective maintenance. This model was the result of many years of experience and its application to the data produced fair although not satisfactory results ($PRED_{25}$ = 33.33% and $PRED_{50}$ = 53.47%). However, the model previously adopted by the subject company included as independent variables the total number of maintenance tasks and the size of the software systems, thus not taking into account the differences in the effort required to accomplish tasks of different type. Our observation was that the effort required to perform a maintenance task of type A might be sensibly different than the effort required to perform a task of type B or C. For this reason we decided to use the number of tasks of the different types to improve the model and take into account the difficulty and the effort needed for the different maintenance task types.

Moreover, if the number of maintenance tasks of different type might be more difficult to estimate for some projects, the other two models based on more coarse-grained metrics also exhibit good performances. In particular, the model based on the total number of tasks and the size of the systems being maintained uses the same independent variables as the model previously adopted by the organization. However, the old model was not linear while the new model is based on the observations of other studies that the size of the system being maintained linearly impact on the effort [27]. The new models are currently being adopted within the subject company.

Although more complicated and maybe more accurate effort estimation models can be constructed, we have deliberately kept the simplicity and understandability of our model to what can be calculated via existing and simply available metrics and a handheld calculator. This was done to permit the working engineer and manager a quick, easy and reasonably accurate method of gauging the maintenance task effort of the maintenance project in question. The prediction performances of our models are nevertheless very interesting according to the findings of Vicinanza *et al.* [36], in particular considering that what is really wanted by software management is not to predict accurately, but to control over the final results.

Another great advantage of our models is their easy and immediate applicability for *ex ante* prediction. In fact, any effort estimation model has to know in advance the values of the independent variables included in the model; as these values are unknown when the model is applied they must be estimated. The total number of maintenance tasks of a maintenance period is early, easily, and with reasonable accuracy estimable from the maintenance history of the software system. Typically companies conducting third party maintenance collect this metric, or related metrics, such as the fault rate and meantime between faults. Indeed, such metrics have a crucial role in the contract bid, as they generally constitute a periodic indicator of the promptness of the service level delivered, that is a basic point for the economic agreement. Moreover, whenever the maintenance tasks are classified and the number of maintenance tasks of different type can be reliably estimated from the past history, project managers can use more accurate models than the one based on the total number of maintenance tasks.

We also had empirical evidence that the corrective maintenance process under study was quite stable with little improvement over project time. This is due to the long dated experience of the subject company and its maintenance teams in conducting corrective maintenance projects on legacy systems running on traditional mainframe platforms. Perhaps this is one of the reason the company does not collect for this type of projects data concerning other factors, such as personnel skills that also generally influence maintenance projects [19]. However, this is also a limitation of the effort estimation models derived from our study that can only be applied to the analyzed domain and technological environment. The subject company is currently involved in several legacy system migration and web-based integration projects and it is likely that the software systems being maintained in the future will be based on heterogeneous technological platforms thus requiring different skills in the maintenance teams. Future work will be devoted to introduce in the maintenance projects different metric plans aiming at characterizing the difference in the maintenance projects of the subject organization.

6. ACKNOWLEDGEMENTS

We would like to thank A. Persico for helping in collecting and analyzing the data. The work described in this paper is supported by the project “Virtual Software Factory”, funded by Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) and jointly carried out by EDS Italia Software, University of Sannio, University of Naples “Federico II”, and University of Bari.

7. REFERENCES

[1] Aversano, L., Betti, S., De Lucia, A., and Stefanucci, S. Introducing Workflow Management in Software Maintenance Processes. Proceedings of Int. Conference on Software Maintenance, Florence, Italy, IEEE CS Press, 2001, 441-450.

[2] Banker, R.D., Datar, S.M., Kemerer, C.F., and Zweig, D. Software Complexity and Maintenance Costs. *Communications of ACM*, vol. 36, no. 11, 1993, 81-94.

[3] Belady, L., and Lehman, M. An Introduction to Program Growth Dynamics. *Statistical Computer Performance Evaluation*, W. Freiberger ed., Academic Press, NY, 1972, 503-511.

[4] Boehm, B.W. *Software Engineering Economics*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1981.

[5] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, vol. 1, 1987, 57-94.

[6] Briand, L., Basili, V., and Thomas, W. A Pattern Recognition Approach for Software Engineering Data Analysis. *IEEE Transactions on Software Engineering*, vol. 18, no. 11, 1992, 931-942.

[7] Briand, L., Langley, T., and Wieczorek, I. A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. *Proceedings of 22nd Int. Conference on Software Engineering*, Limerick, Ireland, IEEE CS Press, 2000, 377-386.

[8] Caivano, D., Lanubile, F., and Visaggio, G. Software Renewal Process Comprehension using Dynamic Effort Estimation. *Proceedings of Int. Conference on Software Maintenance*, Florence, Italy, IEEE CS Press, 2001, 209-218.

[9] Coleman, D., Ash, D., Lowther, B., and Oman, P. Using Metrics to Evaluate Software System Maintainability, *IEEE Computer*, 1994, 44-49.

[10] Conte, S., Dunsmore, H., and Shen V. *Software Engineering Metrics and Models*. Benjamin/Cummings Publishing Company, 1986.

[11] De Lucia, A., Fasolino, A., and Pompella, E. A Decisional Framework for Legacy System Management. *Proceedings of Int. Conference on Software Maintenance*, Florence, Italy, IEEE CS Press, 2001, 642-651.

[12] De Lucia, A., Pannella, A., Pompella, E., and Stefanucci, S. Assessing Massive Maintenance Processes: an Empirical Study. *Proceedings of Int. Conference on Software Maintenance*, Florence, Italy, IEEE CS Press, 2001, 451-458.

[13] Eick, S.G., Graves, T.L., Karr, A.F., Marron, J.S., and Mockus, A. Does Code Decay? Assessing the Evidence from Change Management Data. *IEEE Transactions on Software Engineering*, vol. 27, no. 1, 2001, 1-12.

[14] Fioravanti, F., and Nesi, P. Estimation and Prediction Metrics for Adaptative Maintenance Effort of Object-oriented Systems. *IEEE Transactions on Software Engineering*, vol. 27, no. 12, 2001, 1062-1084.

[15] Gray A., and MacDonnell, D. A Comparison of Techniques for Developing Predictive Models of Software Metrics. *Information and Software Technology*, vol. 39, 1997, 425-437.

[16] Hastings, T.E., and Sajeev, A.S.M. A Vector-Based Approach to Software Size Measurement and Effort Estimation. *IEEE Transactions on Software Engineering*, vol. 27, no. 4, 2001, 337-350.

[17] IEEE Std. 1219-1998, Standard for Software Maintenance. IEEE CS Press, Los Alamitos, CA, 1998.

[18] Jeffery, R., Ruhe, M., and Wieczorek, I. A Comparative Study of Two Software Development Cost Modeling Techniques Using Multi-Organizational and Company-Specific Data. *Information and Software Technology*, vol. 42, no. 14, 2000, 1009-1016.

[19] Jorgensen, M. Experience With the Accuracy of Software Maintenance Task Effort Prediction Models. *IEEE Transactions on Software Engineering*, vol. 21, no. 8, 1995, 674-681.

[20] Kemerer, C.F., and Slaughter, S. An Empirical Approach to Studying Software Evolution. *IEEE Transactions on Software Engineering*, vol. 25, no. 4, 1999, 493-509.

[21] Lehman, M., and Belady, L. *Program Evolution: Processes of Software Change*. Academic Press, Austin, 1985.

[22] Lindvall, M. Monitoring and Measuring the Change-Prediction Process at Different Granularity Levels: An Empirical Study. *Software Process Improvement and Practice*, no. 4, 1998, 3-10.

[23] R. Little and D. Rubin, *Statistical Analysis with Missing Data.* John Wiley and Sons, 1987.

[24] Meyers, R.H. *Classical and Modern Regression with Applications*. Duxbury Press, Boston, 1986.

[25] Miyazaki, Y., Takanou, A., Nozaki, H., Nakagawa, N., and Okada, K. Method to Estimate Parameter Values in Software Prediction Models. *Information and Software Technology*, vol. 33, 1991, 239-243.

[26] Nesi, P. Managing Object Oriented Projects Better, *IEEE Software*, vol. 15, no.4. 1998, 50-60.

[27] Niessink, F., and van Vliet, H. Predicting Maintenance Effort with Function Point. *Proceedings of International Conference on Software Maintenance*, Bari, Italy, IEEE CS Press, 1997, 32-39.

[28] Ramil, J.F. Algorithmic Cost Estimation Software Evolution. *Proceedings of Int. Conference on Software Engineering*, Limerick, Ireland, IEEE CS Press, 2000, 701-703.

[29] Rousseeuw, P.J., and Leroy, A.M. *Robust Regression and Outlier Detection*. N.Y., John Wiley & Sons, 1987.

[30] Shepperd, M., Schofield, C., and Kitchenham, B. Effort Estimation Using Analogy. *Proceedings of Int. Conference on Software Engineering*, Berlin, Germany, IEEE CS Press, 1996, 170-178.

[31] Shepperd, M., and Kadoda, G. Comparing Software Prediction Techniques Using Simulation. *IEEE Transactions on Software Engineering*, vol. 27, no. 11, 2001, 1014-1022.

[32] Smith, R.K., Hale, J.E., and Parrish, A.S. An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation. *IEEE Transactions on Software Engineering*, vol. 27, no. 3, 2001, 264-271.

[33] Stensrud, E., and Myrtveit, I. Human Performance Estimating with Analogy and Regression Models. *IEEE Transactions on Software Engineering*, vol. 25, no. 4, 1999, 510-525.

[34] Strike, K., El Emam, K., and Madhavji, N. Software Cost Estimation with Incomplete Data. *IEEE Transactions on Software Engineering*, vol. 27, no. 10, 2001, 890-908.

[35] Stuart, A., and Ord, J.K. *Kendall's Advanced Theory of Statistics*. 5th Edition, vol. 2, London, Edward Arnold, 1991.

[36] Vicinanza, S., Mukhopadhyay, T., and Prietula, M. Software Effort Estimation: an Exploration Study of Export Performance. *Information System Research*, vol. 2, no. 4, 1991, 243-262.

[37] Walkerden, F. and Jeffery, R. Software Cost Estimation: A Review of Models, Process, and Practice. *Advances in Computers*, vol. 44, 59-125, 1997.

[38] Wellman, F. *Software Costing*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1992.