A Case Study in Productivity Benchmarking: Methods and Lessons Learned

Lionel C. Briand, Khaled El Emam, Isabella Wieczorek
Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6, Kaiserslautern , D-67661, Germany
e-mail: {briand, elemam, wieczo}@iese.fhg.de

International Software Engineering Research Network Technical Report ISERN-98-08

A Case Study in Productivity Benchmarking: Methods and Lessons Learned[1]

Lionel C. Briand, Khaled El Emam, Isabella Wieczorek
Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6, Kaiserslautern , D-67661, Germany
e-mail: {briand, elemam, wieczo}@iese.fhg.de

Abstract

Productivity benchmarking allows software development projects and organizations to compare themselves to the market place in a given sector of industry. However, in practice benchmarking presents many difficulties such as identifying a meaningful basis of comparison. The European Space Agency (ESA) outsources many software projects. They have accumulated a large cost database from these projects. In this paper, we present a method for productivity benchmarking, as well as the productivity benchmarks we derived for one of our customers based on the ESA database. Furthermore, we provide usage scenarios for these models by describing how these models can be practically applied for benchmarking purposes. We developed alternative types of benchmarks using two different modelling techniques, namely least squares regression and regression trees. The most accurate model is obtained using least-squares regression, explains 92% of the variation in project effort, i.e., R^2=0.92, corresponding to an average magnitude of relative error (MRE) of 0.34. Nevertheless, regression tree models are more intuitive and easier to apply for benchmarking purposes.

1. Introduction

In the past, the word "benchmark" has been used in various ways. In this paper, we define productivity benchmarks as instruments that allow an organization/project to *compare* its productivity to those of other *similar* organizations/projects. The ability to benchmark the productivity of their projects provides software organisations a number of advantages. These include the ability to determine whether they are competitive in a given business sector, and whether a significant productivity improvement is required for sustaining a particular business.

In practice, productivity benchmarking presents a number of difficulties. For example, what is meant by "compare productivity" and how does one determine if another project is "similar"? Of course there are the simplistic answers to these questions. However these generally do not provide satisfactory results when operationalized.

In this article we present what we believe are the fundamental principles of software engineering benchmarking and the results of a study aimed at building productivity benchmarks using data collected by the European Space Agency. In particular, the benchmarks are applicable to European projects in the space and military application domains. The benchmarks have been constructed using different analytical techniques, namely ordinary least squares regression and regression trees.

2. Background

For the construction of benchmarks, three issues need to be clarified. First, we define the general concepts of benchmarking productivity. Second, we explain how productivity values are compared. Third, how is similarity defined.

[1] This work was in part supported by Daimler-Benz Aerospace, RIO6, Bremen.

2.1 Productivity Benchmarking Concepts

Benchmarking productivity is a data intensive process. This means that it is necessary to have a benchmarking database containing productivity measures as well as other variables for a set of completed projects.

The basic benchmarking process is to identify a subset of projects in the database that are similar to the project to be benchmarked. Then the project is compared to this subset to determine whether its productivity is better or worse and by how much (see also [9]).

A variation on this theme is to build a model of productivity from the database. Then to estimate the expected productivity for projects similar to the project to be benchmarked. One can compare their actual project's productivity with the expected productivity distribution for similar projects to determine whether they are better or worse. Both of these approaches have been applied in our study.

This process may seem somewhat similar to the process of cost estimation. However there are differences in the details. Benchmarking is usually performed after the project is completed, while cost estimation is performed at the start of a project. Therefore, the variables in the database that can be used in a cost estimation context should only be the ones that are available at the start of the project. However, for benchmarking, one can also use variables that are only available at the end of the project.

2.2 Comparing Productivity

Let us assume that we have found the "similar" projects, and now we want to compare our productivity with the productivity of those similar projects. An obvious approach is to take the average productivity of the similar projects and see whether we are above or below it. This has a number of disadvantages. First, as is common with software project productivity data, there are likely to be some extreme productivity values for some of these similar projects. Such extreme values can have a large impact when using the average, hence giving a distorted benchmark result. This problem can be easily solved by using a robust statistic, such as the median, instead of the average. The second difficulty is that if we know, let's say, that our project is above the median productivity of similar projects, this still does not tell us by *how much*. Therefore, we need some measure of "distance" from the median project and our project. Such a measure of distance should also be easily interpretable. Simply computing an absolute distance would not really be interpretable since the variation in productivity among similar projects would not be taken into account.

To put the approach we use in context, we first consider another domain where a similar concept to benchmarking has been in existence for quite some time.

In the domain of education, test scores have been benchmarked for a long time. This means that a student's raw score is converted into a score that reflects his/her standing relative to some "norm" group. There are a multitude of ways in which this can be done (see [8]). One of these approaches is to convert a raw score into a percentile score. This has the advantage that no assumptions about the distribution of scores in the population need be made, and also that the derived score is intuitive.

In the context of productivity benchmarking, the same concept can be applied. The 50^{th} percentile would be the median. This would indicate that 50% of the similar projects have a productivity that is the same or lower as your project. The 75^{th} percentile is the value where 75% of the similar projects have the same or lower productivity as your project. The distance from the median can be measured in terms of the percentage of similar projects.

However, for productivity benchmarking, instead of converting a raw productivity value into a percentile value, we can define productivity ranges that are equivalent to percentile

ranges. We choose to have 4 ranges that are equivalent to the quartiles of the productivity distribution of similar projects. The 4 ranges represent four productivity levels. Two levels are below the median and two levels are above the median. Four levels were chosen because it was felt that this provided sufficient granularity for meaningful interpretation without assigning too much unwarranted precision to the benchmark results.

2.3 Defining Similarity

The next question is how do we find "similar" projects ? Similarity has to be defined with respect to some set of project characteristics such as application domain, real time and storage constraints (characteristics referred to as attributes). Ideally, similar projects should have similar values on these attributes as well as having productivities that do not vary too much. For example, let's say that our project has a team size of 7. If the "similar" projects are those that have team sizes between 7 and 10, then this class of projects in our benchmarking database should not have productivities that vary, say, tenfold. Otherwise they are not similar enough since they represent projects that vary too greatly in their productivity. It then becomes prudent to try to use other variables that partitions the set of projects with team sizes between 7 and 10 into smaller subsets to reduce the variation in productivity.

The above discussion indicates that the attributes have to represent important variables that distinguish companies in terms of their productivities. But also the attributes have to be of importance for business decisions. For example, if my business domain is aerospace, I would not be interested in benchmarking my projects against projects in the computer games domain (see also [9]). Therefore, application domain would be an important variable to consider when identifying similar projects.

If one is using an already existing database (as in our case), then the potential attributes are predefined. The attributes that we use are presented later in this article.

There are many analytical techniques that can be used for identifying similar projects. An obvious one is cluster analysis [7] (also known as analogy in software engineering, e.g., [12]). However, this generally leads to clusters that are not optimal in terms of the variation in productivity values. The reason is simple: cluster analysis only considers the attributes and does not take into account the actual productivity values.

A class of data analysis techniques that build clusters taking into account the attributes and the productivity values are regression trees (see [4]). Another more common technique that can be used is least squares regression analysis (see [11]). We will present both later in this article.

3. Context of the Study

The database used in this study is the European Space Agency (ESA) multi-organization software project database. Since 1988, the ESA continuously collects historical project data on cost and productivity from different application domains. The data comes from European organizations, with applications from the aerospace, military, industrial, and business environment. Once a project questionnaire is filled out, each data supplier is contacted to ensure the validity and comparabiliy of the responses. Each data supplier regularly receives data analysis reports of the dataset (see [10]).

The version of the database we analysed consisted of 157 projects. The breakdown of projects by environment was, 36% space, 32% military, 22% industry, and 9% business projects. The variables that are taken into account in the analysis are listed Table 1. These are variables that potentially may have an impact on software project cost. The variables defined on an ordinal scale are some of the COCOMO factors (see [3] for further details). We

generated an additional variable called compression, because it was thought that it would have an impact on effort. It indicates the effort per unit of duration, i.e., the schedule pressure, and is defined as effort/duration. Our analysis is based on 101 projects from the space and military environment, in order to consider only comparable projects.

Variable	Description	Scale	Values / Range / Unit
ENV	Project environment	nominal	Space, Military
CATEGORY	Project's Category	nominal	On Board, Message Switching, Real Time, Ground Support Equipment, Simulators, Ground Control, Tools, Others
PROJTYPE	Type of SW Project	nominal	Customized Application, Partly Customized Application, Integration Project, Embedded Application, SW Product Development, Other
KLOC	New developed code	interval	1 KLOC=1000 Lines Of Code
EFFORT	Effort for SW project	interval	MM, where 1 MM=144 hours/month
DUR	Duration of SW project	interval	month
TEAM	Maximal team size on one stage of a project	interval	
LANG	Used programming language or combinations	nominal	ADA, Assembler, C, Coral, Fortran, Pascal, 4GL, Cobol, … and combinations
VIRT	virtual machine volatility	ordinal	2-5 (low-very high)
RELY2	required reliability	ordinal	1-5 (very low-very high)
TIME	execution time constraints	ordinal	3-6 (nominal-extra high)
STOR	main storage constraint	ordinal	3-6 (nominal-extra high)
MODP	use of modern programming practices	ordinal	1-5 (very low-very high)
TOOL	use of software tools	ordinal	1-5 (very low-very high)
LEXP	programming language experience	ordinal	1-4 (very low-high)

Table 1: Variables used from the ESA database.

4. Regression Tree Models

A regression tree is a collection of rules of the form: "if (x and y and ...) then z", displayed in the form of a binary tree[2] [4]. Regression trees classify instances (in our case software projects) with respect to a certain variable (in our case productivity). Each node of a tree

[2] The tree does not have to be a binary tree. However, the technique that we use for the analysis presented in this article produces binary trees.

specifies a condition based on one of the project variables (e.g., RELY2) we have selected. Each branch corresponds to possible values of this variable. A project can be classified by starting at the root node of the tree and selecting a branch to go to based on the project's specific variable value (e.g., RELY2 = HIGH). Moving down the tree branch corresponding to the value of the variable, a new node is reached and the procedure above can be repeated until a terminal node / leaf is reached. For each terminal node and based on the projects it contains, the mean, median, and quartile values are computed for productivity. These statistics can be used for benchmarking purposes. One can, for example, determine whether a project's productivity is significantly below the node median value. More precisely, one may determine in which quartile of the node distribution the new project lies. If the project lies in the first or last quartile, then it is significantly lower/higher than expected and the reason why this happened should be investigated. The organization can then gather lessons learned from projects and build a corporate experience base over time [1].

Regression trees have several main advantages. They are easy to use and interpret. They can deal with many variables of different types (discrete and continuous). They can be easily used for benchmarking of a given new project, since the past projects in the corresponding terminal node can be used as a baseline of comparison.

Building a regression tree involves recursively splitting the data set until a stopping criterion is satisfied [4]. The splitting criterion used is the split, in a given node, which most successfully separates / distinguishes the node productivity values. A terminal node is reached when a minimal number of observations is left in a node, or the homogeneity regarding productivity within a node is very high. In our case the minimal number of observations in a node was set to 10 (see [13][14]). Productivity for a node can be predicted by using the mean or median value of the observations in the node. One advantage of regression trees is that several trees can be generated and used for estimation: for a given project to estimate, select the tree where the terminal node to which belongs the project shows minimum productivity variance.

4.1 Results

As a dependent variable, i.e., the variable used to benchmark projects, we have used productivity. Independent variables, i.e., the influential factors on productivity, were team size, project category, programming language used, compression factor, and 7 of the COCOMO factors (for further details about COCOMO factors see [3]).

We did not use effort as a dependent variable with regression trees, because this would mean that we have to account for the effect of size (KLOC) as an independent variable. However, this would have complicated the interpretation of the trees, which is one of their major advantages.

During the course of the analysis, we have generated many alternative regression trees. The trees were built considering all the variables mentioned in Table 1.

In order to select the best model, the evaluation criterion we used to assess the trees' accuracy was the mean magnitude of relative error over all tree nodes (see Equation 1).

$$MMRE = \frac{1}{no\,of\,observations} \bullet \sum_{i=1}^{no\,of\,observations} \left| \frac{actual\,value_i - predicted\,value_i}{actual\,value_i} \right|$$

Equation 1: Mean magnitude of relative error.

The best regression tree in terms of the accuracy regarding productivity is presented in Figure 1. In this tree the three variables, team size (TEAM), project category (CATEGORY),

and use of software tools (TOOL) appear to have the most significant influence on productivity.

The total number of considered projects is 56. This is because, if a missing value for a variable occurs, the whole observation is ignored for building the tree. The tree is to be interpreted in the following way:

- On the first level of the tree, projects are first split according to their team size. If the team size is lower or equal to 7 persons, these projects are split further according to their category. For 29 projects the team size is lower or equal to 7 persons; 27 projects have a team size greater than 7 persons.
- Following the left branch: Projects falling in the categories "on board systems", or "simulators" have a mean productivity of 0.35 KLOC/MM. There are 10 projects out of the 29 projects.
- Similarly, projects falling in all other categories (see Table 1), have a predicted productivity of 0.58 KLOC/MM. This holds for 19 projects.
- Projects with more than 7 team members and where tool usage is between low and nominal have a predicted productivity of 0.09 KLOC/MM. This is the case for 16 projects.
- Projects with a team larger than 7 persons and high-very high usage of tools have a predicted productivity of 0.217 KLOC/MM. This holds for 11 projects.

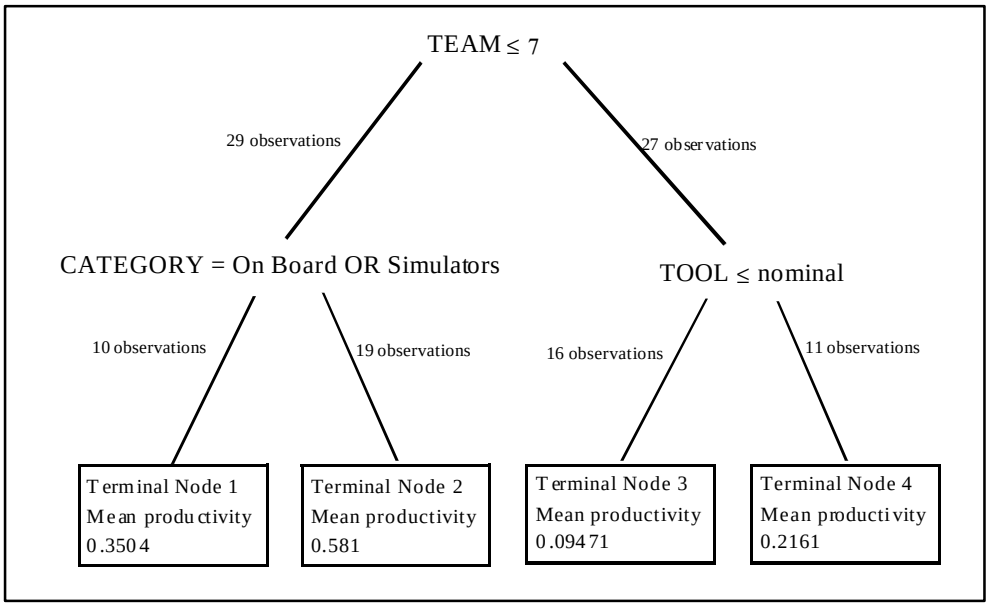

Figure 1: Regression tree model.

This tree intuitively makes sense. Projects with a smaller team size have higher productivity than projects with larger teams. If tool usage is higher than nominal, then productivity is also higher. The usage of tools is more significant for larger projects.

We numbered the terminal nodes from left to right, starting with 1. For each terminal node, the mean, and the within node MMRE for effort and productivity are given in Table 2.

The overall MMRE values for effort and productivity for the tree are 0.567 and 1.00 respectively. This is the best result we obtained for regression trees. It has to be considered, that there are large variations in the productivity MMRE's across the terminal nodes. Therefore, the accuracy/usability of regression trees depends strongly on the terminal node a project lies in. Therefore, choosing a tree or a regression model (see next section) for benchmarking will also be determined by the terminal nodes' MMRE that are relevant to a given project to be benchmarked.

	Terminal Node 1	Terminal Node 2	Terminal Node 3	Terminal Node 4
Mean productivity	0.35	0.581	0.095	0.216
MMRE (effort)	0.74	0.45	0.65	0.49
MMRE (productivity)	1.56	0.99	0.88	0.68

Table 2: Terminal node information for regression tree.

4.2 Using Regression Trees for Benchmarking

The information contained in a regression tree (Figure 1) and additional information about the terminal nodes (Table 3) can be used for benchmarking in the following way. Based on the independent variables available, the projects in this terminal node are comparable, in terms of productivity, to the completed project you want to assess. Therefore, we can consider its productivity distribution as a comparison baseline. Thus, the productivity range within each node can be partitioned into 4 productivity levels (Table 3), i.e., intervals according to the 25% quartile value, the median, and the 75% quartile value. For example, in the terminal node 2 in the tree (Table 3), the 25% quartile value is 0.3602, meaning that 25% of the projects in node 2 have a productivity equal to or below this value. Similarly, the 75% quartile value in node 2 is 0.8058, meaning that 25% of the projects have a productivity value higher than 0.8058. For a given project to be assessed:

- Follow the appropriate path in the regression tree (Figure 1) until a terminal node is reached, e.g., terminal node 2.
- Calculate the actual project productivity (size/effort), e.g., 0.86.
- Compare the productivity value of the completed project with the quartile and median values. Determine to which productivity level the project belongs. If the project lies within a low (≤25% quartile) or high (>75% quartile) productivity level, then reasons for such large differences from the comparison baseline should be investigated.

Productivity Levels	Interpretation	Terminal Node 1	Terminal Node 2	Terminal Node 3	Terminal Node 4
Level 1	Very low Productivity	≤ 0.1166	≤ 0.3602	≤ 0.0388	≤ 0.0937
Level 2	Low Productivity	≤ 0.2068	≤ 0.5	≤ 0.0615	≤ 0.2351
Level 3	High Productivity	> 0.2068	> 0.5	> 0.0615	> 0.2351
Level 4	Very high Productivity	> 0.6071	> 0.8058	> 0.1049	> 0.2842

Table 3: Productivity levels, ranges, and their interpretations for regression tree.

Table 3 summarizes the possible ranges of productivity for a project to be assessed. An actual productivity value of a completed project can be easily assessed using this benchmark table.

5. Multivariate Least Squares Regression Models

The multivariate regression model that we built uses effort as the dependent variable. This is different from the regression tree models that were presented in Section 4. The effect of size on effort can be modelled using least-squares regression analysis (see [2][11] for details of ordinary least squares regression analysis).

Using effort as a dependent variable with ordinary regression models still allows one to perform productivity benchmarking because we model effort for a given size. This will be

illustrated below. The advantage of using effort as a dependent variable is that non-linear relationships between size and effort can be easily modelled. Productivity defined as a ratio of size over effort inherently assumes proportionality between effort and size.

We present the best regression model in this section in terms of goodness of fit. The model explains a large percentage of the variation in effort (92%), and therefore it can be considered good for benchmarking purposes.

The main advantages of least squares regression models are that they are commonly used and therefore there is familiarity with them, and when their assumptions are met, they can provide valid and rather accurate results. However, compared to regression trees, least squares regression produces models that are not as easy to understand and not as intuitive. This is especially true when there are many variables and many interactions amongst them.

5.1 Results

Our regression model was built considering the influential factors on effort: project category, new developed code, compression factor, team size, programming language, and the 7 COCOMO factors.

In this model we have two types of variables: continuous and discrete. For continuous variables (i.e., new developed code, compression, and team size) we took their natural logarithms in order to linearize their relationship with effort, thus allowing the use of least squares estimates for their coefficients. For the discrete variables (e.g., the COCOMO factors) we created dummy variables (i.e., variables taking 2 possible values, usually 0 and 1). The dummy variable approach for modelling discrete variables in regression equations is explained in [2][11].

A stepwise process was performed to extract those variables that have a significant (p-value < 0.05) influence on effort. All non-significant variables were excluded from the model. The model reported in Equation 2, turned out to be the best one.

(1) $\ln(EFFORT) = a \bullet \ln(KLOC) + b \bullet \ln(COMPRESSION) + c \bullet \ln(TEAM) + d \bullet VIRT_D + f \bullet STOR_D + Interpect$

(2) $EFFORT = KLOC^{a} \bullet COMPRESSION^{b} \bullet TEAM^{c} \bullet e^{d \bullet VIRT_D} \bullet e^{f \bullet STOR_D} \bullet e^{Intercept}$

Equation 2: Linearized (1) and corresponding exponential (2) regression model.

Equation 2 gives the model specification of the linearized and the corresponding exponential model. Table 4 provides information about the regression coefficients and the intercept. Table 5 summarizes the model's accuracy results, and gives the number of projects considered to build the model.

Parameter	Estimate	p-value
a	0.286	0.0004
b	0.379	0.0066
c	0.689	< 0.0001
d	0.455	0.0123
f	0.653	0.0084
Intercept	1.586	<0.0001

Table 4: Parameter estimates for regression model.

MMRE (effort)	0.343
MMRE (productivity)	0.344
R^2 / R^2 (adjusted)	0.92 / 0.91
Number of Observations	47

Table 5: Information about regression model.

5.2 Using Multivariate Regression Models for Benchmarking

The usage of the regression models for the purpose of benchmarking is explained in this section. The presented approach is based on the distribution of residuals, in contrast to the approach based on the productivity distribution presented for the regression trees (Section 4.2). The residual is the difference between actual effort and predicted effort. Given that our regression models have a rather good fit to the data, the predicted effort estimates the effort expected for a project with similar characteristics to the ESA database. Therefore, the residual represents the "distance" between the effort obtained on the real project being benchmarked and the effort for a project with similar characteristics to the ESA database. If the real project has a larger effort value than the predicted value (i.e., a positive residual), then it means that the real project is less productive than the previous similar ESA projects. If the real project has a smaller effort value than the predicted value (i.e., a negative residual), it means that the real project is more productive than previous similar ESA projects.

Just using the sign of the residuals (i.e., just positive or negative residuals) is a little crude. Therefore, we can improve on that by splitting the residual distribution into 4 intervals corresponding to the 4 quartiles.

For a given project to be benchmarked regarding productivity, the distribution of residuals can be used in the following way. Partition the distribution of the residuals according to the 25%, 50% (median), and 75% quartile values. Thus, the effort range can be partitioned into 4 levels (i.e., intervals) according to the quartile values (see Table 6). For example, the 25% quartile value is -0.4116, meaning that for 25% of the projects the residual values are equal to or below this value. Similarly, the 75% quartile value for regression model 1 is 0.4316, meaning that 25% of the projects have a residual value higher than this value. Benchmarking can be performed in the following way:

- Calculate the predicted effort for the project to be benchmarked using Equation 2 (linearized model equation).
- Calculate the residual effort value. The residual is the actual value - predicted value.
- Determine the productivity level using the benchmark table (Table 6).

Productivity Levels	**Interpretation**	**Residual Ranges**
Level 1	very low productivity	> 0.288
Level 2	low productivity	> 0.016
Level 3	high productivity	$\leq$ 0.016
Level 4	very high productivity	$\leq$ -0.346

Table 6: Residual ranges and productivity levels for linearized regression model.

6. Comparison of the Models

In this paper we have presented two models for benchmarking productivity. To choose from these models for benchmarking the following criteria must be considered: accuracy of the model, availability of project data required for applying a model, and ease of use of a model.

It turned out that the least squares regression model is the most accurate in general. Therefore, as a starting point one would use the least squares regression models. However, if data for the variables that are required to use these models are missing, then it may not be possible to use the most accurate model. Missing variables is a problem that can occur in practice. Under these conditions, the next most accurate model ought to be selected.

The general procedure is to try using least-squares regression model which is, in general, more accurate. However, if one of the trees' terminal nodes corresponding to the project to be benchmarked shows a smaller MMRE (effort or productivity) than the least squares regression model, it should be considered as the favorite alternative. This will not often be the case, however.

Despite a general poorer accuracy, the regression tree models are more intuitive and easy to use for benchmarking purposes. This criterion can also be taken into account when deciding on the particular model to use. In some cases, it may be worthwhile to use a less accurate model simply because it is easier to use. Trees can be refined by experts who can, for example, add decomposition levels to the tree based on their own experience. Thus, a model which is both data and expert driven can be derived. This remains, however, a research issue to be investigated.

7. Conclusions

We have presented two different types of multivariate models that can be used for productivity benchmarking. The models were developed using data in the European Space Agency (ESA) project cost database. Multivariate least squares regression has been used, although this technique is a priori more adequate to build models for prediction purposes. We have described the best obtained multivariate regression model. Using regression tree analysis, we provided results for the best obtained regression tree in detail. We have also presented procedures that can be followed for interpreting the benchmarks. These procedures identify four levels of productivity, depending on the extent to which the productivity value of a given project is better/worse than similar previous projects in the ESA database.

Despite the fact that we focused on external benchmarking, it is important to note that an identical modeling approach can be used for internal benchmarking in an organization. Future work will consist of using more sophisticated techniques in an attempt to build more accurate benchmark models. In terms of research activities, future steps will involve the use of a novel technique called Optimized Set Reduction (OSR) [5], [6], which has been designed to produce more accurate benchmark models, but for which tool support was so far not fully available. The goal of OSR is to provide models which are both accurate in their predictions and interpretable, so that the multivariate patterns observed in the data can be discussed with experts.

8. Acknowledgements

We are grateful to Benjamin Schreiber and Uffe Mortensen, from the ESA/ESTEC center, for giving us access to the ESA project database. We would also like to thank Wolfgang Belau and Norbert Schielow, from DASA in Bremen, for all the interesting discussions regarding project benchmarking and their support in this study.

9. References

[1] K.-D. Althoff, F. Bomarius, C. Tautz, "Case-Based Construction of Knowledge Systems for Realizing Learning Software Organizations", submitted to the European Conference on Artificial

Intelligence, ECAI-98, Workshop on Building, Maintaining, and Using Organizational Memories (OM-98), Brighton, UK.
[2] W.D. Berry, S. Feldmann, "Multiple Regression in Practice", Series: Quantitative Application in the Social Sciences, 50, SAGE University Paper, 1985.
[3] B. Boehm, "Software Engineering Economics", Prentice Hall, 1981.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, "Classification and Regression Trees," Wadsworth & Books/Cole Advanced Books & Software, 1984.
[5] L. Briand, V. Basili, W. Thomas, "A pattern recognition approach for software engineering data analysis", IEEE Transactions on Software Engineering, Vol. 18, Number 11, pp. 931-942, November 1992.
[6] L. Briand, V. Basili, C. Hetmanski, "Developing interpretable models with optimized set reduction for identifying high-risk software components", IEEE Transactions on Software Engineering, Vol. 19, Number 11, pp. 1028-1044, November, 1993.
[7] B. Everitt, "Cluster Analysis", Edward Arnold, 1993.
[8] H. Lyman "Test Scores and What They Mean", Prentice-Hall, 1963.
[9] K. Maxwell, L. van Wassenhove, S. Dutta, "Benchmarking: The Data Contribution Dilemma", Proceedings of the 1997 European Software Control and Metrics Conference, Berlin, Germany, May, 1997.
[10] K. Maxwell, L. van Wassenhove, S. Dutta, "Software Development Productivity of European Space, Military, and Industrial Applications", IEEE Transactions on Software Engineering, Vol. 22, No. 10, October 1996.
[11] L.D. Schroeder, D.L. Sojoquist, P.E. Stephan, "Understanding Regression Analysis - An introductory Guide", Series: Quantitative Applications in the Social Sciences, 57, SAGE University Paper, 1986.
[12] M. Shepperd, C. Schofield, "Estimating Software Project Effort Using Analogies", IEEE Transactions on Software Engineering, Vol. 23, Number 12, pp. 736-43, November, 1997.
[13] D. Steinberg, P. Colla, CART, "Classification and Regression Trees, Tree Structured Nonparametric Data Analysis", Interface Documentation, Salford Systems, 1995.
[14] Mathsoft Inc., "S-Plus, User's Guide", Version 4.0, Data Products Division, Mathsoft, Inc., Seattle, Washington, August 1997.