UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

EARLY ESTIMATION OF SOFTWARE SIZE IN OBJECT-ORIENTED ENVIRONMENTS
A CASE STUDY IN A CMM LEVEL 3 SOFTWARE FIRM

Marco Ronchetti, Giancarlo Succi, Witold Pedrycz, and Barbara Russo

January 2004

Technical Report # DIT-04-007

Early estimation of software size in object-oriented environments a case study in a CMM level 3 software firm

Marco Ronchetti

Dipartimento di Informatica e Telecomunicazioni
Università di Trento, Trento, Italy

Giancarlo Succi

Center for Applied Software Engineering
Free University of Bolzano-Bozen, Italy

Witold Pedrycz

Department of Electrical and Computer Engineering
University of Calgary, Calgary, Alberta

Barbara Russo

Center for Applied Software Engineering
Free University of Bolzano-Bozen, Italy

Abstract

Early estimation of the size of a software product is extremely important. In this paper we analyze two software packages developed by a CMM level 3 software firm. We study if any property of analysis objects can be used to infer the size of the final code in an object-oriented environment. In both cases we find the number of methods well correlated with software size, in the sense that the correlation with the final size is high ($r > 0.77$) and significant at the level .05. Inferential statistics guarantee that the results of this study are also applicable outside the scope of the two projects.

1 INTRODUCTION

An accurate estimation of the time and the effort required to develop a software product is a prerequisite for effective management of the software development process. Several time and effort estimation techniques exist, such as those proposed by Boehm (1996), Putnam (1992), and Albrecht (1983).

Most of these techniques either define a link between external features of the product and the size or the time/effort to develop, or focus on finding a relation between the size of the product and the time/effort to develop.

Our work tries to take advantage of a specificity of object-oriented development: the same "language" is used throughout the software lifecycle. We use objects to represent requirements, analysis documents, design documents, and code.

A huge effort has been placed in the last few years on object-oriented metrics and their use to predict effort, quality, and productivity. Among the most significant works, Li and Henry have targeted the number of changes in a component (1993), Basili, Briand, and Melo the faults present in an objects (1996), Chidamber, Darcy, and Kemerer the productivity, the rework effort, and the design effort (1998). All these three works use the Chidamber and Kemerer metric suit (1994), often referred to as "CK metrics."

The long-term goal of our study is to determine if any property of analysis objects is highly and significantly correlated with the time/effort to develop the product, so that early in the lifecycle managers and developers can have reliable estimates of the required effort to complete the product.

This paper presents an experiment in which we have collected and analyzed data from 2 real projects of a CMM level 3, ISO 9000 software firm to determine whether there are metrics of analysis objects that can predict the size of the final product.

Our experiment has two main limitations. First, the analysis metrics used are those supplied by the OMT-based object modeling tool in use by the firm; it would have been very interesting to use all the CK metrics, unfortunately they were not available. Second, we focus on the size of the final product and not on the time or the effort to develop it; this is simply because we only have the information of the size.

The uniqueness of our work is that we focus on analysis objects. (Li and Henry, 1993) did not clarify whether its measures came from design or from code. (Basili et al.,1996) dealt with code. (Chidamber et al., 1998) took into consideration design objects only for one of the three data-sets they used; in the other two cases they dealt with code. Moreover, we deal with industrial data from a CMM level 3, ISO 9000 company; (Basili et al., 1996) analyzed students projects, (Li and Henry, 1993) and (Chidamber et al., 1998) do not specify the maturity of the firms they dealt with.

We find that a very simple measure, the number of methods of an analysis object, is well and significantly correlated with the size of the final product in both our data sets.

This paper is organized as follows. Section 2 describes the set of metrics we have used. Section 3 details the environment where we have collected the measures and the process applied to collect the measures. Section 4 presents the results that we have obtained. Section 5 draws some conclusions.

2 BACKGROUND

As mentioned, several metrics exist for Object-Oriented analysis documents. Several books include excellent surveys and discussions, such as (Whitmire, 1997; Henderson-Sellers, 1997; De Champeaux, 1997).

In this paper we will use the following class metrics: external complexity and internal complexity (Moreau and Dominick, 1990), depth of inheritance tree, number of methods, number of children (Chidamber and Kemerer, 1994), and number of attributes (De Champeaux, 1997).

We adopted this set of metrics since it was computed automatically by the design tool in use. The designers of the systems supplied the metrics to us at the end of the project without any direct intervention during the projects or any change in the standard software development practices. The developers of the project were not informed in advance of the subsequent data analysis. The

selection of a different set of metrics would have compromised the feasibility and the internal validity of our experiment.

The formal definitions of the adopted metrics follow.

- **External Complexity** of a class C - *EC(C)*:

$$EC(C) = \sum_{i=1}^{n} AC(A_i)$$

  Where n is the number of associations of the class, A_i is each association of the class, and *AC(A)* is:

$$AC(A) = \frac{Arity(A) - 1}{2} + 0.1 * Q + 0.1 * \#\, Attributes(A)$$

  Where Q is 1 if the association is qualified, 0 otherwise.

- **Internal Complexity** of a class C - *IC(C)*:

$$IC(C) = \frac{\sum_{i=1}^{n} MC(M_i)}{n} + 0.2 * n$$

  Where n is the number of methods of the class, M_i is each method, and *MC(M)* is:

$$MC(M) = 1 + 0.1 * \#\, parameters(M)$$

- **Depth of the Inheritance Tree** of a class C *–DIT(C)*: maximum length of the path from the class to the root of the inheritance tree
- **Number of Methods** of a class C *–NoM(C)*, **Number of Children** of a class C *–NoC(C)*, and **Number of Attributes** of a class C *–NoA(C)* are self-defined

For the sake of precision, NoM is used by both (Basili et al., 1996) and (Chidamber et al., 1998) under the name "Weighted Method Count", WMC, assuming constant unitary weight for each method. We prefer the simpler and more intuitive name.

The scales of EC and IC are ratio. The scales of DIT, NoM, NoC, and NoA are absolute. Therefore, we can use them all as independent variables in parametric linear regressions.

3 DESCRIPTION OF THE WORKING ENVIRONMENT

The data come from a European software company, certified CMM level 3 and ISO 9000. The company operates in the domain of telecommunications; it employs around 300 people; it has a defined software development process based on object orientation and reuse. The software development process is iterative and includes in each iteration separate activities for object-oriented analysis, Object-Oriented design, and Object-Oriented programming. The analysis and design language is OMT (Rumbaugh et al., 1990). The programming language is C++.

Most of the software engineers have a MSc degree either in Electrical Engineering or in Computer Science. They have 2 to 7 years of programming experience. When they are hired, they are exposed to a significant amount of training –about 3 months full time, so that once they start working, they have a good understanding of the processes, languages, and tools in use.

The projects under analysis are part of a large product line in the network management domain. The people involved in the project are a representative sample of the employees in the firm.

As mentioned, during the development phases the developers were not aware that their work would be analyzed later. The data collection occurred as part of the standard post-mortem analysis. Thus, it did not create any artifact in the development processes.

Altogether, we think that the collected data refer to valid samples of the products of the firm.

4 ANALYSIS OF THE RESULTS

A summary of our approach follows.

1. We provide a general description of the statistical variables: descriptive statistics and boxplot.
2. We run a parametric correlation between our analysis metrics and LOC; in this way we identify the factor with the overall highest linear influence, number of methods.
3. We identified a linear parametric model and we verify that it is sound, that is, the properties of normality, homoscedasticity, independence of error, and linearity hold. A limited presence of autocorrelation of the error is identified in the two data-sets; the other three properties are satisfied enough.
4. We study whether the model can be extended to a more general population.
5. Since the hypothesis of independence of error is not fully, we investigate what would happen if the four properties did not hold. We run the non parametric Spearman's rank correlation (Thomas, 1997) and we still find the same kinds of relations.

The sizes of the samples are not large enough to support multivariate analysis.

4.1 General description of the statistical variables

Table 1 and Table 2 contain the descriptive statistics for the considered variables in Project 1 and Project 2 respectively. Project 1 contains 12 analysis classes and Project 2 contains 11 analysis classes. Figure 1 and Figure 2 contain the box-plots of the same variables. LOC is the total lines of code for implementing an analysis class; an analysis class can result in several code classes; LOC is measured by counting the number of semicolons. The other variables are the ones defined in Section 2.

Name	Min.	Max.	Mean	Std. Dev.
LOC	29	618	260.17	226.73
EC	1.00	6.00	1.87	1.48
IC	1.60	6.90	3.21	1.70
DIT	0	1	0.42	0.51
NoM	2	27	8.50	7.61
NoC	0	3	0.41	0.99
NoA	0	14	6.25	4.58

Table 1: Descriptive statistics for Project 1

Name	Min.	Max.	Mean	Std. Dev.
LOC	3	714	308.36	263.30
EC	.50	4.00	0.95	1.11
IC	1.40	7.20	4.09	2.01
DIT	0	2	1.09	0.54
NoM	2	30	14.00	10.27
NoC	0	8	1.00	2.49
NoA	0	21	5.82	5.84

Table 2: Descriptive statistics for Project 2

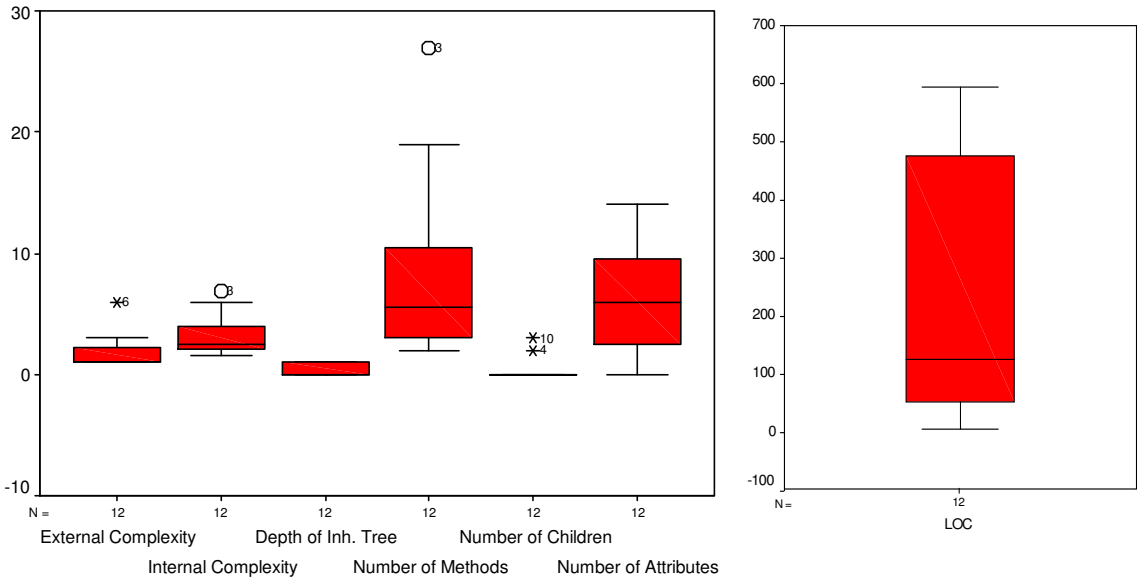


Figure 1: Box-plots for Project 1

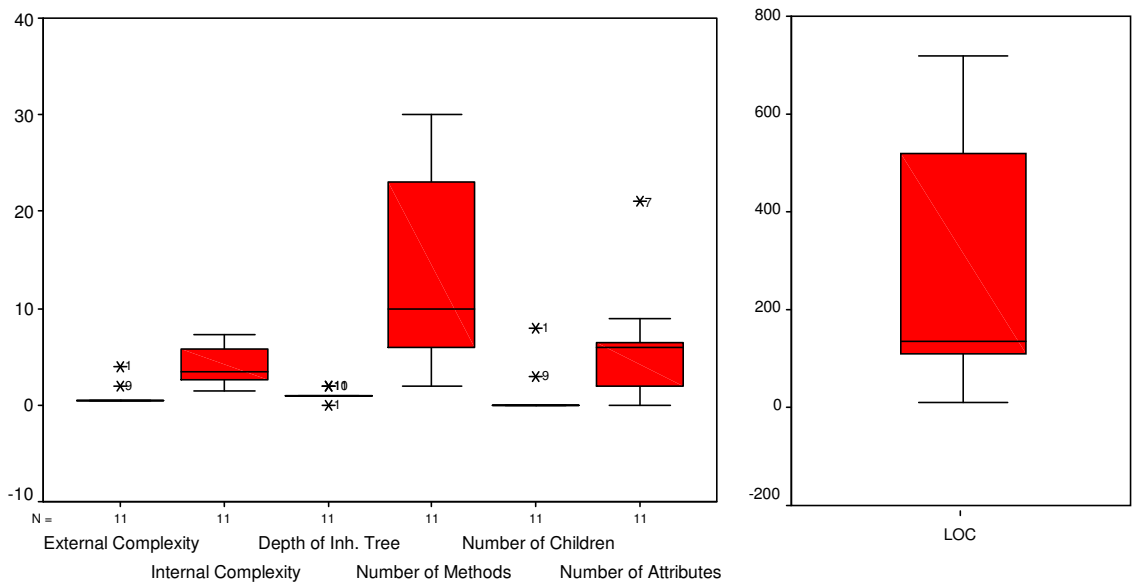


Figure 2: Box-plots for Project 2

As already noticed by (Chidamber et al., 1998), DIT and NoC tend to assume very low values.

4.2 Parametric correlations

The scale of LOC is ratio (Fenton and Pfleeger, 1997), therefore we can use it both in parametric correlations and as dependent and independent variable in parametric regressions.

The parametric correlations identify the attributes of the analysis objects that have the highest linear influence on the size of the code. Table 3 contains the results of the parametric correlations. A "*" indicates a correlation significant at the 0.05 level.

Project	EC	IC	DIT	NoM	NoC	NoA
1	-0.365	0.447	-0.318	0.771*	-0.389	0.502
2	-0.481	0.827*	0.208	0.800*	-0.473	0.232

Table 3 : Parametric correlations between LOC and attributes of the analisis objects

In both cases the number of methods of the analysis objects are well correlated with the size of the resulting system; they explain more than 59% of code size and the correlation is statistically significant at the level 0.05.

We proceed in our exploration trying to identify a model of the kind:

$$LOC = a \times NoM + b$$

The Internal Complexity is also well correlated with code size, but only for Project 2. Therefore, we drop it from this initial exploration phase.

4.3 Development of the linear model and verification of its validity

Table 4 and Table 5 contain the features of the linear models for Project 1 and 2 respectively. Figure 3 presents the regression line LOC vs. NoM for Project 1 and Project 2.

	Coefficients	Std. Error	Significance
(Constant)	***b*** = 64.813	67.128	0.357
NoM	***a*** = 22.983	5.996	0.003

Table 4: Linear Model for Project 1

	Coefficient	Std. Error	Significance
(Constant)	***b*** = 21.257	87.702	0.814
NoM	***a*** = 20.508	5.134	0.003

Table 5: Linear Model for Project 2

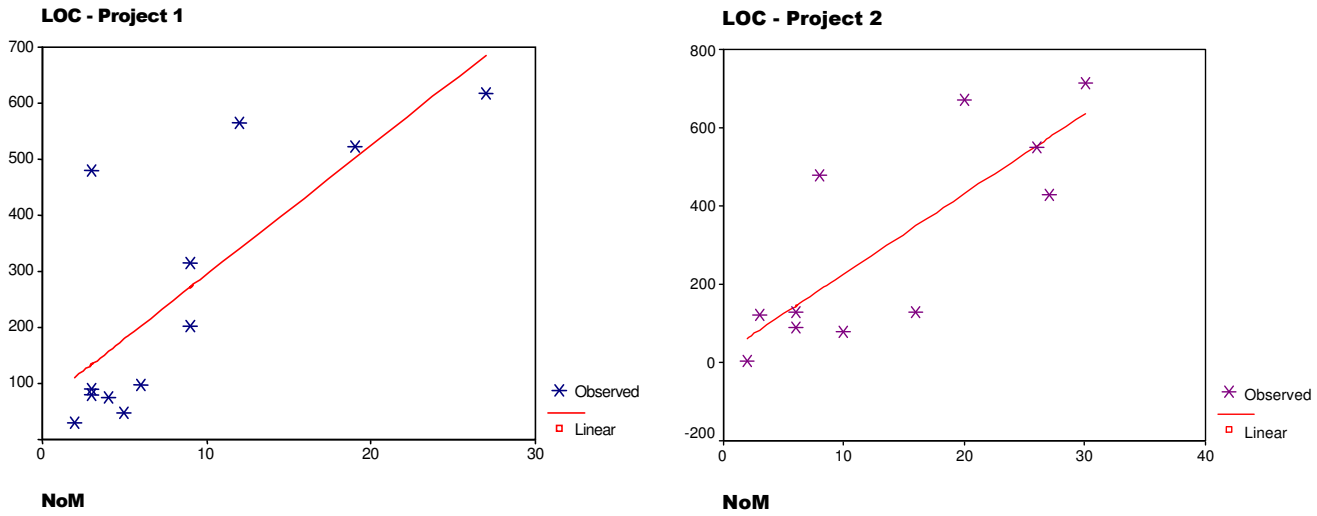


Figure 3: Regression Lines

The two proposed linear models follow.

(Project 1) ***LOC = 22.983 × NoM + 64.813***

(Project 2) ***LOC = 20.508 × NoM + 21.257***

Notice that for both datasets the two regression coefficients are close (22.983 and 20.508). The two constants are very different (64.813 and 21.257).

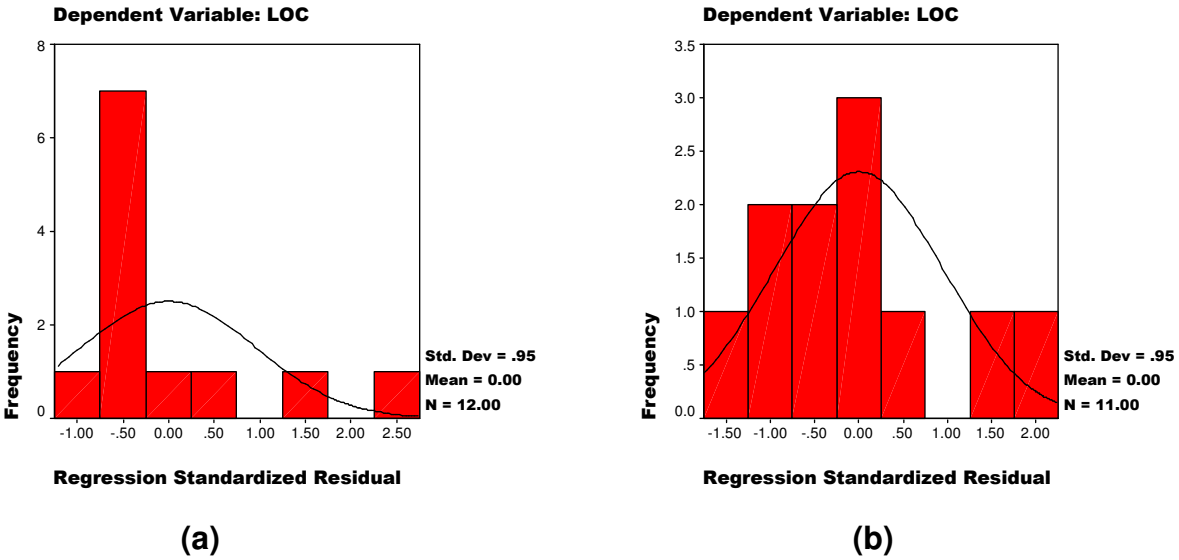
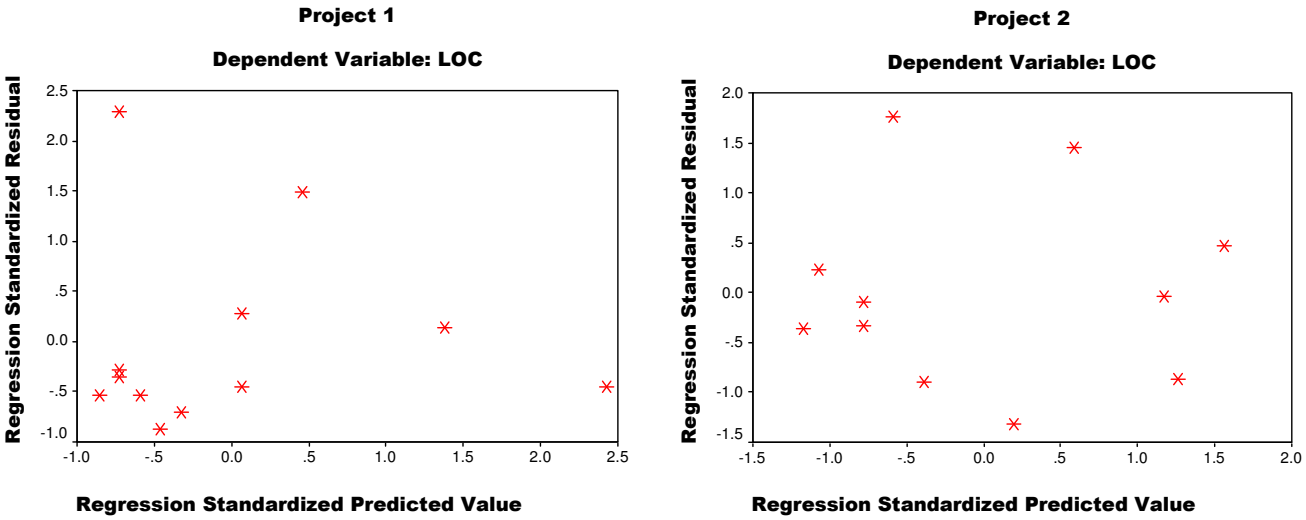


Figure 4: Normality of residuals

To validate the model we need to verify normality, homoscedasticity, independence of error, and linearity (Tryfos, 1998).

Normality. We verify the normality by visual inspections. Figure 4 presents the histograms of the residuals for Project 1 (a) and 2 (b). The distribution of residuals is somewhat normal for both projects.

Figure 5: Scatterplot of residuals



Homoscedasticity. We verify the homoscedasticity by virtual inspection of the scatter-plots of residual (Figure 5). In both scatter-plots we do not see any major pattern.

Independence of Errors. The Durbin Watson coefficient is 1.087 for Project 1 and 2.477 for Project 2. This highlights the presence of a limited auto-correlation of errors (Tryfos, 1998).

Linearity. Both projects present a significant linear relation between NoM and LOC.

4.4 Extension of the model to the underlying populations

Since both correlations are statistically significant, we can extend the models to the underlying populations. The overall models for the populations have the form:

$$LOC = a \times NoM + b$$

We always refer to populations, since we have no indication whether the two datasets come from the same population. On the contrary, we cannot claim that the reference population is all the programs written by the software firm: the coefficient ***a*** is almost the same in both projects, but ***b*** is quite different.

This might imply that we could derive a fairly stable, organization-wide value for ***a***, while each project needs to calibrate its own value of ***b***.

4.5 Non parametric correlations

Our conclusions depend on our arbitrary decisions of what to consider as acceptable:

- the values of the Durbin Watson coefficients
- the shape of the histograms of the residuals
- the randomness of the scatter-plots of the residuals

(Briand et al, 1996) claims with strong evidences that linear parametric models are in general applicable to any software engineering data.

However, if we rejected the approach of (Briand et al., 1996) and any one of the three decisions above, we could not conclude with parametric methods. We had to resort to non-parametric correlations.

Table 6 contains the values of the Spearman's rank correlations.

Project	EC	IC	DIT	NoM	NoC	NoA
1	-0.445	0.540	-0.367	0.765*	-0.344	0.534
2	-0.608	0.843*	0.349	0.779*	-0.608*	0.230

Table 6 : Spearman's rank correlations

The Spearman's rank correlation still identifies the same relations: there are significant and high non-parametric correlations between LOC and NoM in both projects. Also, we notice in Project 2 significant and high non-parametric correlations between LOC and IC and between LOC and NoC.

However, on the basis of non-parametric methods we could not develop a linear parametric model, like the ones previously presented.

5 DISCUSSION

It would be interesting to abstract general conclusions out of these data. We can make conjectures about the possibilities of identifying similar correlations in similar firms. However, as already mentioned, on the basis of the available data we cannot define any general rule.

In addition, it is worth evidencing that the effective corporate software development process in place in the analyzed firm may be one of the causes of the high correlations between a property of the analysis objects and the final sizes of the systems. In this paper we deal with a mature firm at level 3 of the Capability Maturity Model. Such maturity implies that the process in place is fairly stable and more predictable.

As mentioned several times, it would have been much better to analyse the correlation with effort or calendar time instead of size. Here we are assuming that the size in LOC of the project is correlated with the effort to develop it. The argument is similar to that of (Li and Henry, 1993) where the number of changes in a class is considered a proxy for maintenance effort. We are fully aware that this is far from optimal under at least two perspectives:

(a) as also (Fenton and Pfleeger, 1997) affirms, it is by far better to perform direct comparisons (in our case between the 6 considered metrics and effort or time) rather than using proxies (in our case the size as measured by LOC),

(b) several researchers have objected strongly on the use of LOC as a size measure.

The scarcity of industrial data on the object of this research is a severe problem.

6 COMPARISON WITH SIMILAR STUDIES

It is interesting to compare this investigation to three very important studies in the field: (Li and Henry, 1993), (Basili et al., 1996), and (Chidamber et al., 1998). Table 7 provides a synopsis of their main features using an extension of the tabular approach provided in (Basili et al., 1996).

Study	Li and Henry, 1993	Basili et al., 1996	Chidamber et al., 1998	This study
Origin of data	Software firm	Students	Software firm	Software firm at level 3 of CMM
Lifecycle phase of the independent variables	Design or code	Code	One dataset of design, two datasets of code	Analysis
Reference prog. language	Classic-ADA	C++	C++	C++
Collection of the independent vars	Automatic	Automatic	Automatic	Automatic
Dependant variables	Number of lines changed during maintenance as proxy for maintenance effort	Number of faults	Productivity, rework effort, design effort	Size as proxy for development effort
Collection of the dependant variables	Part of the regular process	Ad-hoc	Part of the regular process	Part of the regular process
Prediction system	Class 3	Class 3	Class 3	Class 1
Investigative technique	Survey	Experiment	Survey	Survey
Statistical analysis	Parametric linear regression	Logistic regression	Stepwise linear regression including dummy variables	Parametric and non parametric correlation; linear regression
Conclusions	The CK metrics but CBO influence significantly the dependant var.	NoM, DIT, CBO, RFC influences significantly the dependant var.	High values of CBO and LCOM influences significantly the dependant vars.	NoM influences significantly the dependant var.
Other considerations	N/A	The cross-correlations of the CK metrics are low.	DIT and NoC assume low values. CBO, NoM, and RFC are highly correlated; the other correlations between metrics are low.	DIT and NoC assume low values; the cross-correlation between the three CK metrics used are low.

Table 7: Comparison of studies on the use of Object Oriented Metrics

The meanings of DIT, NoC, and NoM have been described previously. For the definitions of CBO – Coupling Between Objects, LCOM –Lack of Cohesion between Methods, and RFC –Response For a Class, the reader can refer to (Chidamber et al., 1998) or the earlier (Chidamber and Kemerer, 1994).

Origin of data. Three out of four studies deal with industrial data; however, as previously mentioned; only this study addresses a mature software organization.

Lifecycle phase of the independent variables. The lifecycle phase of the independent variables varies from code (Basili et al., 1996) to design (Chidamber et al., 1998) to analysis -this study; it is unclear whether Li and Henry used the design objects or the code objects to extract their data.

Reference programming language. In (Li and Henry, 1993) the reference programming language is a proprietary extension of ADA with Object Oriented features. In the other three cases, it is C++.

Collection of the independent variables. In all four cases the collection of the independent data has been performed automatically.

Dependant variables. The dependent variables are most interesting in the studies by Basili and colleagues and by Chidamber and colleagues: they tracked properties of the final products -number of faults, and of the overall development processes –productivity, design and rework effort. Li and Henry use the number of lines changed as a proxy for the maintenance effort, while we use the size in lines of code as a proxy for the development effort.

Collection of the dependant variables. In all the studies but (Basili et al., 1996), the collection of the dependents variables was part of the regular development process, therefore it was not affected by the empirical investigation. Basili and colleagues asked a group of independent reviewers to test the produced code.

Prediction system. According to the definition of (Fenton and Pfleeger, 1997), the three earlier studies belong to Class 3 –internal product attributes predict process attributes. This study belongs to Class 1 –early internal product attributes predict later internal product attributes.

Investigative technique. Again according to the definition of (Fenton and Pfleeger, 1997), the investigative techniques are "survey" for all the studies but (Basili et al., 1996); they are "retrospective studies of a situation." (Basili et al., 1996) is an "experiment," that is, "an investigation of an activity, where key factors are identified and manipulated to document their effects." (The citations are from pages 118 and 119 of the cited book.)

Statistical analysis. The four studies use different statistical analysis tools. (Li and Henry, 1993) uses standard multivariate linear regression and do not question the validity of their models in deep. (Basili et al., 1996) uses logistic regression. (Chidamber et al., 1998) uses stepwise linear regression, and dummy variables to model the effects of different developers; also, it performs an in-depth analysis of the satisfaction of the criteria for establishing a linear model.

This study uses linear parametric correlation and linear regression, discusses the validity of the resulting model, and details an alternative procedure based on non-parametric correlation.

We could not obtain the information of who developed each task, so we could not use dummy variables to represent the different developers skills.

We did not apply stepwise linear regression since (a) given the limited size of our sample, multivariate analysis was not feasible and (b) we were looking for one independent variable that

would have provided the best prediction over the two projects. Analysing the correlation coefficients gave us a global indication of behavior rather than just one variable per project.

As a proof of concept we checked the results of stepwise linear regression (Tryfos, 1998). We obtained the same result for Project 1: stepwise regression identified a model with one single independent variable, NoM. A slightly different result was found for Project 2: still one single variable was identified, but IC was preferred over NoM in the last step of the stepwise regression. Since our goal is to find the single most influential variable across the two projects, the use pf stepwise regression leads to the same results.

Conclusions. All the four studies conclude that the independent variables, or a subset of them, influence significantly the dependent variables.

Other considerations. Both (Chidamber et al. 1998) and (Basili et al., 1996) did not find any significant correlation between DIT, NoM, and NoC. Table 8 and Table 9 presents the parametric correlations between these variables -the only CK variables considered here. In general, they are not highly, nor significantly correlated: only in Project 2 we find a significant correlation between NoC and DIT.

	DIT	**NoM**	**NoC**
DIT	1		
NoM	-0.081	1	
NoC	-0.369	-0.186	1

Table 8: Correlation of the Analysis Metrics in Project 1

	DIT	**NoM**	**NoC**
DIT	1		
NoM	-0.360	1	
NoC	-0.670*	-0.469	1

Table 9: Correlation of the Analysis Metrics in Project 2

We have already mentioned that DIT and NoC assume in general low values like in (Chidamber et al., 1998). DIT and NoC had very low values also in (Henry and Li, 1993) –see their tabulations, even if this fact is not evidenced in that paper.

Altogether, all the four studies provide an interesting unique perspective on the practical and effective use of object oriented metrics. It would be interesting to apply meta-analytic techniques to them to try to obtain general conclusions (Hunter and Schmidt, 1990).

7 CONCLUSIONS

In this paper we analyzed two projects from a software company working in the telecommunication domain in northern Italy to determine if any metrics of the analysis objects could predict the final size of the system.

In both projects we found that the number of methods of the analysis objects were highly and significantly linearly correlated with the overall size of the projects.

Moreover, we found that if we did not consider satisfied the requirements to validate a linear model, we could still find a high and significant non parametric correlation between number of methods and lines of code.

However, we were not able to define a unique population for the two data-sets; that is, linear models seemed to hold within each project but not across different projects even within the same company.

Still we were limited to size of the projects as measured by lines of code. It would be extremely interesting to know whether what we found for size applies also to effort and calendar time.

Much more research needs to be done in this field. The availability of industrial data is a prerequisite to any significant step further.

ACKNOWLEDGMENTS

We thank Eliseo Mambella, Angela Lo Surdo, Giuseppe La Commare, and Cristina Zenatti for their precious help in this research, William E. Curry and Eric Liu for reviewing diligently the draft of this paper.

The second author thanks the University of Calgary, the Government of Alberta, the Canadian National Science and Engineering Research Council, and the European Commission for partially supporting this research.

REFERENCES

Albrecht, A.J., and J.E. Gaffney, Jr.(1983) "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, **9**(6)

Basili, V.R., L.C. Briand, and W.L. Melo (1996) "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, **22**(10)

Boehm, B., B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby (1995) "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering*, **1**(1)

Briand, L., K. El Emam, and S. Morasca (1996) "On the Application of Measurement Theory in Software Engineering" *Empirical Software Engineering*, **1**(1)

Chidamber, S.R., and C.F. Kemerer (1994) "A Metrics Suite for Object-Oriented Design," *IEEE Transactions on Software Engineering*, **20**(6)

Chidamber, S.R., D.P. Darcy, and C.F. Kemerer (1998) "Managerial Use of Object-Oriented Software: An Explanatory Analysis," *IEEE Transactions on Software Engineering*, **24**(8)

De Champeaux, D. (1997) *Object-Oriented Development Process and Metrics*, Prentice Hall, Inc.

Fenton, N., S.L. Pfleeger (1997) *Software Metrics: A Rigorous and Practical Approach*, PWS Publishing Co.

Henderson-Sellers, B. (1996) *Object Oriented Metrics: Measures of Complexity***,** Prentice Hall, Inc.

Hunter, J.E., and F.L. Schmidt (1990) *Methods of Meta-Analysis*, Sage Publications, Inc.

Li, W., and S. Henry (1993) "Object Oriented Metrics That Predict Maintainability," *Journal of Systems and Software*, **23**(2)

Moreau, D.R., and W.D. Dominick (1990) “A Programming Environment Evaluation Methodology for Object-Oriented Systems: Part I - The Methodology,” *Journal of Object-Oriented Programming*, **3**(1)

Moreau, D.R., and W.D. Dominick (1990) “A Programming Environment Evaluation Methodology for Object-Oriented Systems: Part II - Test Case Application,” *Journal of Object-Oriented Programming*, **3**(3)

Putnam, L.H. and W. Myers. (1992) *Measures for Excellence: Reliable Software on Time, within Budget*. New Jersey: Prentice-Hall

Rumbaugh, J., M. Blaha, W. Lorensen, F. Eddy, W. Premerlani (1990) *Object-Oriented Modeling and Design*. Prentice Hall, Inc.

Thomas, R. (1997) *Quantitative Methods for Business Studies*, Prentice Hall, Inc

Tryfos, P. (1998) *Methods for Business Analysis and Forecasting: Text and Cases*, John Wiley & Sons, Inc.

Whitmire S.A. (1997) *Object Oriented Design Measurement*, John Wiley & Sons, Inc.