Estimation Models based on Functional Profiles.

Alain Abran, Blanca Gil, Éric Lefebvre

École de Technologie Supérieure - Université du Québec, Montréal (Canada)

aabran@ele.etsmtl.ca, blanca.gil.1@ens.etsmtl.ca, elefebvre@ele.etsmtl.ca

Abstract:

As a software functional size method, Function Point Analysis (FPA) has been used in many organizations for measuring productivity and building estimation models [1]. FPA is based on a number of function types (external inputs, external outputs, external inquiries, internal logical files and external interface files), the set of which we refer to as a functional profile. It has been observed that a majority of projects within a sample are close to having an average functional profile, while some, of course, can be considered as outliers. This paper investigates the functional profiles within the ISBSG international repository, and whether or not productivity varies with such profiles. The results of the statistical analyses lead to distinct estimation models, depending on whether or not a project functional profile is within a reasonable range of the average functional profile for any particular sample.

Key words:

Function Points, functional profiles, software functional size, productivity, estimation, ISBSG.

1 Introduction.

In this paper, we introduce the concept of a software functional profile defined as the distribution of function types within software. The concept of a functional profile can be measured using existing functional size measurement methods by taking into account the types of base functional components (BFC – ISO 14143-1) defined in each functional size method. For instance, in the Function Point Analysis (FPA) method, a functional profile would include the relative distribution of the five function types for any particular project; that is, the percentages of external inputs, external outputs, external inquiries, internal logical files and external interface files (EI%, EO%, EQ%, ILF% and EIF%). Such a functional profile provides information about the distribution of functionality within specific software and permits comparison of its functional distribution with that of a sample of projects for which the average would be known, as well as its distribution across such an average, in terms of, for example, standard deviation, skewness and kurtosis.

Subsequent analysis for productivity comparison can then take this functional profile into account when building estimation models, in addition to other variables, such as the development language, the logical and physical architecture (core architecture, development platform, tools, etc.), and the project context (project origin, project type, development methodology, etc.).

The empirical analyses reported here document the "impact" of functional profiles within samples by programming language: the functional profile was selected as the controlled variable and, of course, the effort as the dependent variable.

The specific research questions investigated here are the following:

- Is there a relationship between the individual function types and project effort?
- Is there a relationship between the functional profile itself and project effort?
- Is there a better relationship with project effort if a project is close to the average functional profile of a sample?

This paper presents the methodology used to define the selection criteria and the results of statistical analyses of functional profiles using industrial projects documented in Release 8 (2003) of the project repository of the International Software Benchmarking Standards Group (ISBSG) [5]. The software used for the statistical analysis is the "LRGL-ISBSG" prototype developed jointly by GÉLOG-ETS and the University of Magdeburg [3]. The analysis also includes visual inspections of the graphical behavior of the controlled variable from the selected samples of projects. This graphical "behavior" included productivity models built using regression lines, confidence interval analysis and Kiviat diagrams.

2 Research context.

2.1 Function types in FPA.

The function types of FPA are defined as follows:

- An External Input (EI) is an elementary process which processes data or control information that comes from outside the application boundary [6].

- An External Output (EO) is an elementary process which sends data or control information outside the application boundary. The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information [6].
- An External Inquiry (EQ) is an elementary process which sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF or EIF [6].
- An Internal Logical File (ILF) is a user-identifiable group of logically related data or control information maintained within the boundary of the application [6].
- An External Interface File (EIF) is a user-identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application [6].

Unadjusted Function Points. The measurement of software follows "the global model of FPA [… that] includes two main parts: the Functional Size and the Adjustment Factor. On the one hand, the functional size is calculated from the counts of all the individual functions of the application measured, whereas the value for the adjustment factor is calculated by adding the values attributed to the general characteristics of the application as a whole. The first result represents the addition of the parts of the whole and is referred to in the literature as *Unadjusted Function Points*, whereas the result of the multiplication of the sum of the parts by the value of the adjustment factor gives the final count in *Function Points*, *Adjusted Size* or *Adjusted Function Points*" [1]. In this paper, we use the Unadjusted Function Points as Functional Size.

2.2 The LRGL-ISBSG prototype.

The LRGL-ISBSG prototype is a software estimation tool built jointly by ETS (Software Engineering Research Laboratory of the ETS) and the University of Magdeburg [3], with data provided by the International Software Benchmarking Standards Group (ISBSG) [5]. This software estimation tool prototype provides a white-box approach to the data analysis of this repository: it allows samples to be built using selection criteria, visual presentation of the data points meeting the criteria, the graphical representation of the regression model built using the sample and statistical information on the parameters of the estimation model derived from the statistical analysis.

2.3 Visualization techniques and Kiviat diagrams.

While it is easy to represent graphically with two-axis relationships across two or three variables, it is much harder to do so with a larger number of variables. For a greater number of variables, Kiviat diagrams are often used, with each variable being indicated by a radial axis. The outer- and inner-concentric circles indicate the upper and lower limits for each component respectively. Kiviat diagrams allow for the recognition of visual patterns within the interval selected (minimum and maximum), and it is up to the readers to pursue the analysis [7].

2.4 Analysis criteria of estimation models.

In linear regression models, the coefficient of determination (R^2) describes the percentage of variability explained by the predictive variable. This coefficient has a value between 0 and 1; when the coefficient is close to 1, it indicates that the variability in the response to the predictive variable can be explained by the model (i.e. there is a strong relationship between the independent and dependent variables [2]). The coefficient of correlation (r) does not measure a causality relation between the X and Y variables: a coefficient close to 1 does not mean that one variable implies the other, it simply expresses the fact that the two variables vary in the same direction [4].

3 Methodology.

After the study of the descriptive statistics of the available ISBSG repository, the second step is a discussion of the selection criteria for the preparation of the samples to be employed to construct the models. Subsequently, using the LRGL-ISBSG prototype, a series of productivity models is constructed. Then, we set the confidence intervals and represent these models graphically; an analysis of the resulting profiles was carried out using the Kiviat technique. This paper concludes by presenting a set of results.

3.1 Data selection criteria for the preparation of samples by programming language.

The criteria for the selection of the projects to be included in this analysis are presented below:

- Project sizes must have been measured with the same size standard (here, the IFPUG standard).
- While in the ISBSG repository, all projects have a total functional size, but not all have details by function type. This criterion therefore verifies that a project has data for at least one function type.

- The next criterion verifies the consistency of the details by function type, that the summation of these detailed function type sizes corresponds to the total size, as recorded in the ISBSG repository (S types = FP). If there is inconsistency, then such a project is not selected for analysis.
- According to [5], a quality label "A" basically means that the data provided have a high integrity quality, "B" that they are high-quality data, but with some factors which could affect their credibility, "C" that some significant data were not provided, so it is not possible to assess the integrity of the data, and "D" due to one factor or a combination of factors, little credibility should be given to the data provided. For this analysis, only projects of quality A and B were selected.
- Finally, samples with significantly less than 30 projects by programming language were discarded: such samples are too small for statistical analysis.

The projects that met these criteria are presented in Table 1. In summary, of the 2027 projects from Release 8 of the ISBSG repository, 236 met all the criteria and were distributed as follows within the following programming languages (Table 1, right-hand column): COBOL (136 projects), NATURAL (67 projects) and C (33 projects). For each of the other programming languages, not enough projects met the criteria for relevant statistical analysis.

Language		Projects based on IFPUG standard	Function type details available	Function type details consistent with Total size	Projects with high quality data (Data quality = A or B)
1	**COBOL**	286	144	144	136
2	**NATURAL**	79	68	68	67
3	**C**	138	33	33	33
					Total = 236

Table 1: Selection criteria, and the number of projects meeting the criteria.

4 Initial analysis of the samples.

4.1 Regression models with functional size as the independent variable.

Initial models were built for each sample in Table 1 (right-hand column) using FPA size as the independent variable and the reported effort as the dependent variable. The next step was to identify outliers using graphical analysis, and rebuild regression models without the outliers identified.

In the COBOL sample, there were the following:

- One significant outlier in terms of size: a project with close to 14,000 FP, while all of the other projects had fewer than 5,000 FP.

- Six outliers in terms of effort: projects with effort between 35,000 and 60,000 hours, while most other projects had fewer than 30,000 hours.

The graphical representation of the COBOL sample, with and without outliers, is presented in Figure 1. Figure 1 and Table 2 show the regression line of 136 COBOL projects and the regression line of 129 remaining projects after elimination of the "outliers".

For NATURAL and C languages (Figures 2 and 3), the selected samples have a strong correlation between FPA size and effort. For these two languages, no obvious outlier was identified, either in size or in effort. Their regression lines and their associated data are shown in Figures 2 and 3 and in Tables 3 and 4.

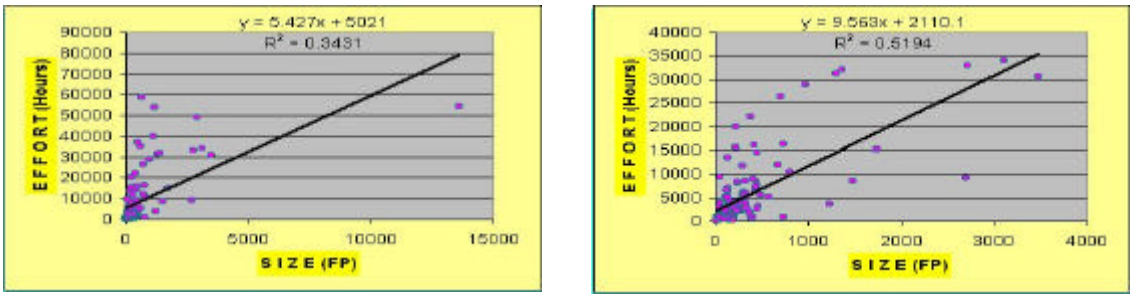


Figure 1: COBOL: regression lines with and without “outliers”.

Total: 136	Min	Mean	Max
Team Size	1.0	8.8	53.0
Months	1.0	8.8	44.0
Size (FP)	10.0	450	13,580
R^2	0.34		
A	5.4		
B	5021		

Total: 129	Min	Mean	Max
Team Size	1.0	8.8	53.0
Months	1.0	8.2	36.0
Size (FP)	10.0	363	3476
R^2	0.52		
A	9.6		
B	2110.1		

Table 2: COBOL: samples and regression coefficients with & without “outliers”

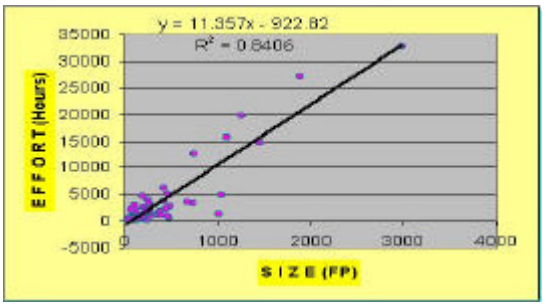


Figure 2: NATURAL regression line.

Total: 67	MIN	Mean	MAX
Team Size	1.0	5.3	20.0
Months	2.0	7.4	48.0
Size (FP)	25	363.3	2983
R^2	0.84		
A	11.4		
B	-922.8		

Table 3: NATURAL sample and regression coefficients

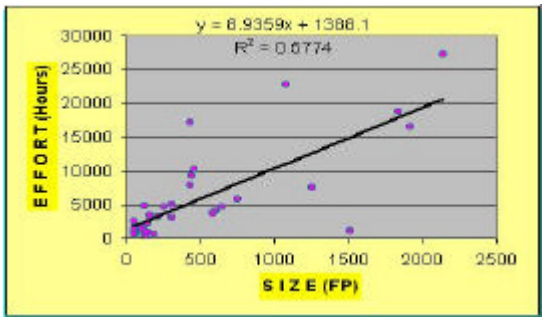


Figure 3: C regression line.

Total: 33	MIN	Mean	MAX
Team Size	1.0	6.2	23.0
Months	2.0	9.1	26.0
Size (FP)	50	507.1	2138
R^2	0.58		
A	8.9		
B	1388.1		

Table 4: C sample and regression coefficients

4.2 Comparison of samples by programming language.

- The samples for the three programming languages have basically the same size intervals (from 10 to 2400 FP, with a single point around 3,000 FP for the NATURAL sample, and four around 3,000 for COBOL).
- Similarly, for the effort intervals, most projects are in the 10 to 30,000 hour range in the regression models.
- C language has the lowest variable cost (e.g. the slope of the model = a coefficient) at 8.9 hours/FP, followed by COBOL at 9.6 hours/FP and NATURAL at 11.4 hours/FP.
- The relationship of effort to size is fair for both C and COBOL, with an R^2 in the 0.50 to 0.60 range, while it has a much larger relationship for NATURAL with an R^2 at 0.84. This is also illustrated in the graph in Figure 2, where the data points of the NATURAL sample are aggregated much more closely to the regression model than for the other two samples with more dispersion.

5 Analysis of functional profiles.

5.1 Descriptive analysis of functional profiles – Kiviat graphs.

The next sets of tables and figures present the distribution of the functional types, or functional profiles, within each sample by programming language.

To determine the functional profile of each sample, we calculated the weighted average (in percentage) for each function type by programming language. This average is based on the calculation of the distribution of function types for each project and on the average distribution across all projects. This way of calculating the averages ensures that they are not unduly influenced by larger projects. A functional profile of a sample is then the set of functional distributions within the sample (Table 5): for instance, the functional profile of the NATURAL sample is: EI = 39%, EO = 28%, EQ = 16%, ILF = 15% and EIF only 2%.

Language	Average Weighted EI	Average Weighted EO	Average Weighted EQ	Average Weighted ILF	Average Weighted EIF	Total
COBOL	**28%**	**28%**	13%	21%	10%	100%
NATURAL	**39%**	28%	16%	15%	2%	100%
C	**36%**	23%	16%	21%	4%	100%

Table 5: Distribution of weighted function types.

Of course, it cannot be expected that the functional profiles of all projects will be close to the sample average; some will, while others could be quite different. This paper investigates next whether or not taking into account the closeness of a project functional profile to the average of the sample can improve the quality of the productivity and estimation models derived from their samples.

5.2 Descriptive analysis of the average functional profiles of the samples.

On the basis of previous work carried out in [1], a threshold of 30% was selected for identifying closeness to the average functional profile and to build distinct sub-samples within each programming language; for instance, all projects with function type profiles within a range of + or – 30% of the average distribution were grouped into one subsample, and the others outside that range in another. For language C, according to the analysis carried out, a + or – 20% range outside the average distribution was sufficient for building two distinct subsets: one which would include about 80% of the projects in the sample, and another about 20%.

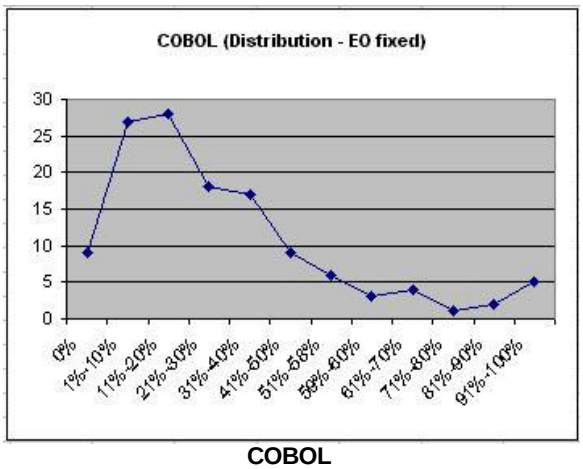
The intent is that the 20% sample should include the projects that would be fairly far apart from the average functional profile.

So, using this approach, the criteria selected for each sample by programming language is the function type with the largest percentage: from Table 5, this criterion gives the EI for both NATURAL and C, and we chose the EO function type for COBOL in order to analyze a different function type.

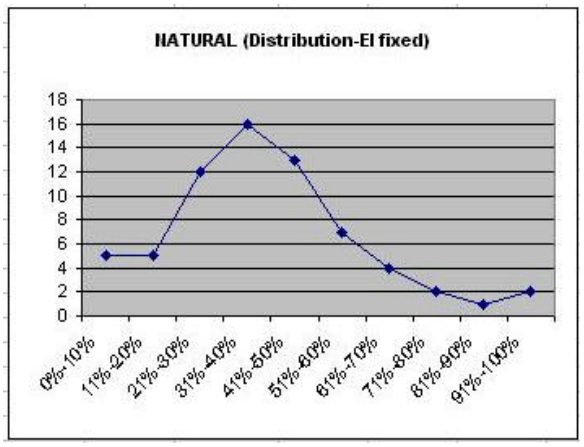
Then, the projects within + or – 30% (+ or – 20% for C) of the average distribution of that largest function type were identified and grouped within two subsamples (within this interval and outside it). Table 6 shows those subsamples generated using the largest function type, and Figure 4 presents the distribution graphics.

Interval of average functional size distribution	COBOL (EO fixed)	NATURAL (EI fixed)	Interval of average functional size distribution	C
-30% and -1%	73	17	-20% and -1%	12
0% and 30%	32	40	0% and 20%	14
-100% and -31%	9	5	-100% and -20%	3
31% and 100%	15	5	20% and 100%	4
TOTAL	**129**	**67**	**TOTAL**	**33**

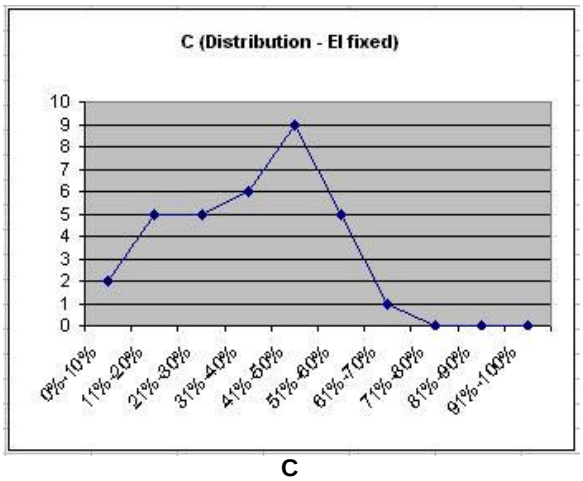
Table 6: Average functional size distribution – data.



COBOL



NATURAL



C

Figure 4: Average functional size distribution – graphs.

This average distribution is presented next in Figure 5, and corresponds to the average functional profile of the projects within each of the three samples.

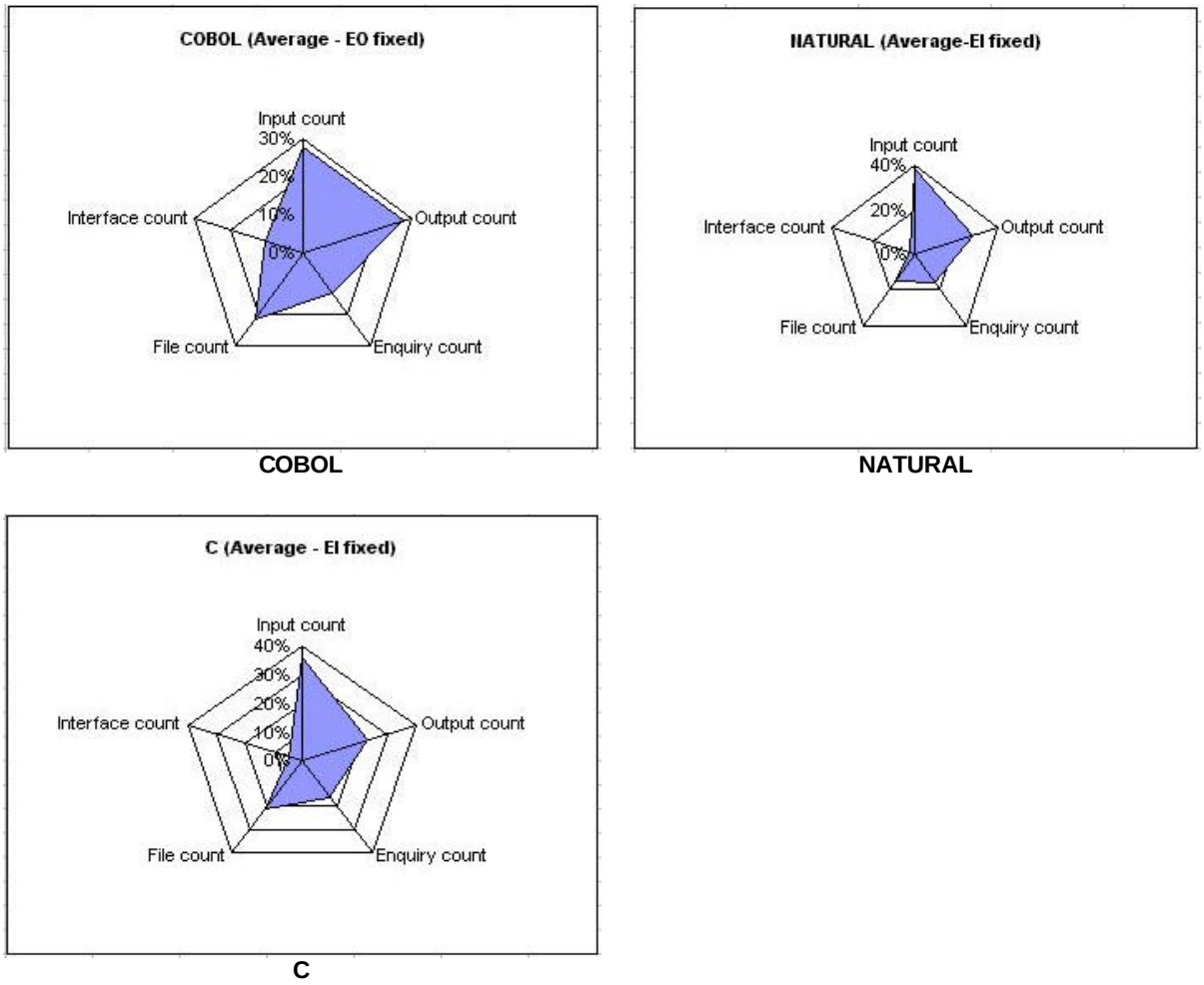


Figure 5: Kiviat diagrams of average functional profiles.

Of course, it cannot be expected that all projects will be close to such average profiles. For the purpose of this study, there was interest in knowing whether or not projects which were not significantly close to this average functional profile had a different relationship to effort.

Figure 6 presents the Kiviat diagrams within the + or – 30% (+ or – 20% for C) across the average distribution of function type, while Table 7 presents the number of projects within the +/– 30% (+ or – 20% for C) intervals, and those outside them. Comparing the Kiviat graphs of the functional profile (Figure 6) and those of the diagrams of average functional profile (Figure 5) for each language, it can be observed that the functional profile of all the projects is similar to the functional profile of the projects within the selected ranges. The functional profiles of the projects outside the selected ranges show the variation of the largest function type for each language (EO for COBOL and EI for NATURAL and C).

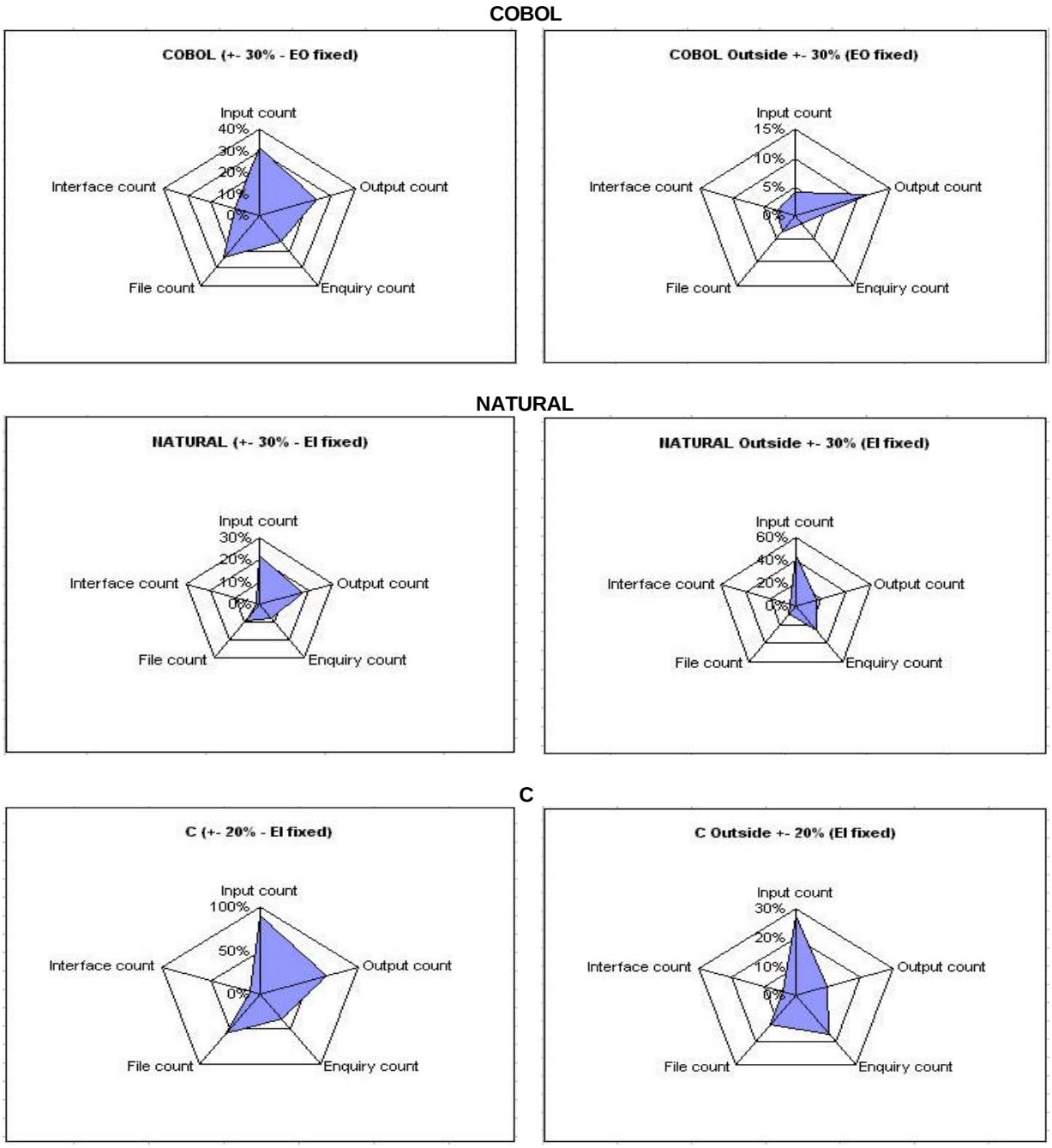


Figure 6: Functional profiles.

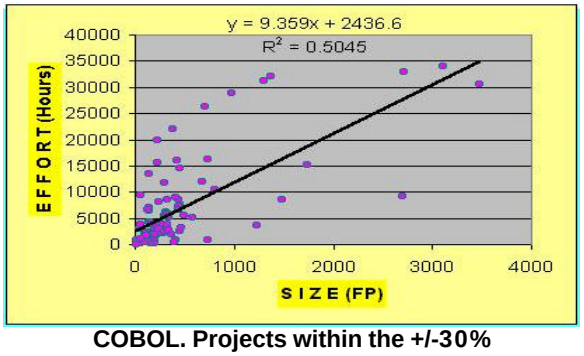
Language	Projects within the +/- 30% intervals	Projects outside the +/- 30% intervals	Total
COBOL	105 (81%)	24 (19%)	129 (100%)
NATURAL	57 (85%)	10 (15%)	67 (100%)
Language	**Projects within the +/- 20% intervals**	**Projects outside the +/- 20% intervals**	**Total**
C	26 (79%)	7 (21%)	33 (100%)

Table 7: Number of projects.

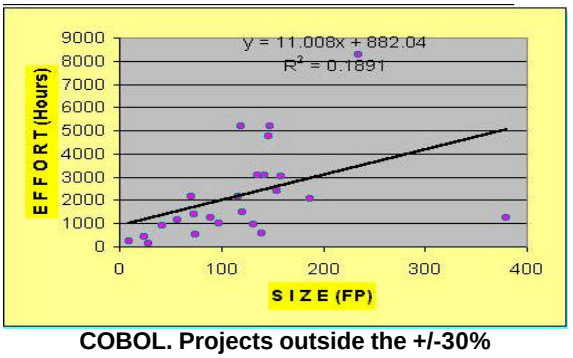
Figure 7 presents the data sets on two axes, and it can be observed that the projects outside of the average functional profiles are relatively small (maximum functional size from 400 FP for COBOL to 600 FP for C) in comparison to the overall data sets (maximum functional size of 2,200 to 4,000 FP, for the samples excluding outliers).

6 Models built with subsamples by functional profile.

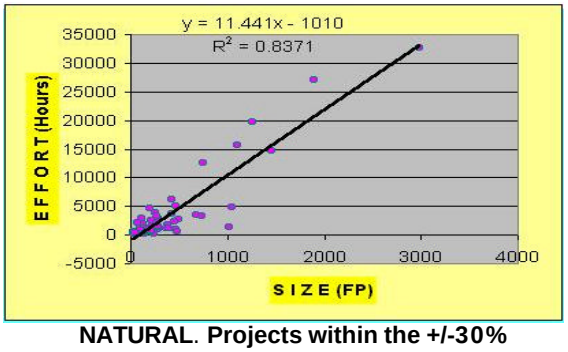
The next set of regression models was built using the subsets defined above; that is, for the projects with a distribution of function types within the + or – 30% (+ or – 20% for C) of the average distribution for that largest function type, and the others with a distribution more than 30% (or 20% for C) from the average functional profiles.



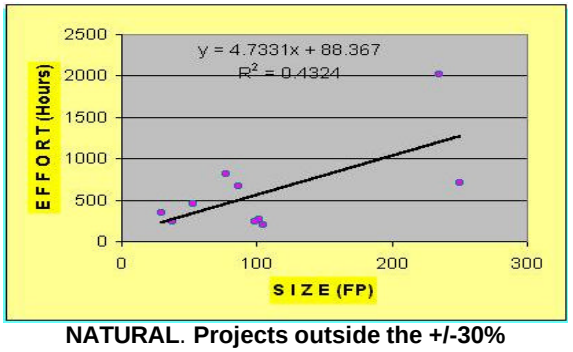
COBOL. Projects within the +/-30%



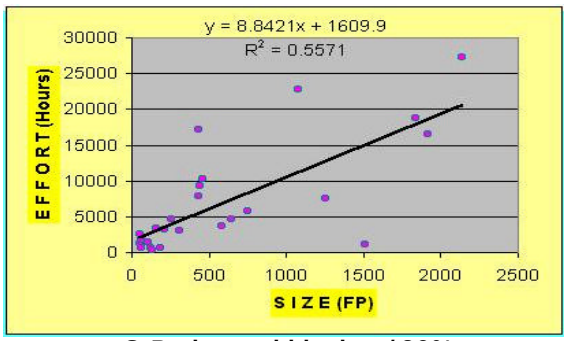
COBOL. Projects outside the +/-30%



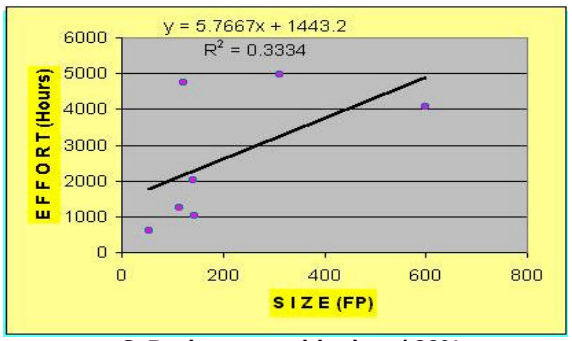
NATURAL. Projects within the +/-30%



NATURAL. Projects outside the +/-30%



C. Projects within the +/-20%



C. Projects outside the +/-20%

Figure 7: Regression models of functional profiles within and outside the interval across averages.

		All projects	Projects within +/-30%	Projects outside +/-30%
COBOL (129 projects)	R^2	0.52	0.51	0.19
	A	9.6	9.4	11
	B	2110.1	2436.6	882
		All projects	Projects within +/-30%	Projects outside +/-30%
NATURAL (67 projects)	R^2	0.84	0.84	0.43
	A	11.4	11.4	4.7
	B	-922.8	-1010	88.4
		All projects	Projects within +/-20%	Projects outside +/-20%
C (33 projects)	R^2	0.58	0.56	0.33
	A	8.9	8.8	5.8
	B	1388.1	1609.9	1443.2

Table 8: Regression models of functional profiles within and outside the interval across averages.

It can be observed from Figure 7 and Table 8 that:

- For the samples within the selected range of average functional profiles (30% for COBOL and NATURAL and 20% for C), the regression coefficients A and B did not deviate much from the coefficients for all the projects in the entire samples.
- For the samples outside of the selected range of average functional profiles, one or both of the regression coefficients vary considerably from the coefficients for all the projects in the entire samples.

These models of the data projects outside of the average functional profiles must therefore be preferred for estimation purposes over the generic models of all projects independently of their functional profiles.

The above relationship between size and effort is derived from completed projects; it is to be noted however, that the information to determine a functional profile can be obtained very early in the development cycle. For estimation purposes at the beginning of a project, the functional profile can be identified however from the start of the project, and therefore it provides right away the required information to determine which estimation model is the most suitable: the functional model within or outside the average functional profile.

7 Analysis of individual function types.

Two types of analysis were conducted, first to study the relationship between sizes by function type (external inputs, external outputs, external queries, internal logical files and external interface files) with respect to the total functional size, and then with respect to project total effort.

7.1 Analysis by individual function type.

For this analysis, the individual size by function type is taken as the independent variable, and the total functional size as the dependent variable. The results of the regression models on these variables are presented in Table 9. It can be observed that, for the three samples, the relationship with respect to total functional size is very strong (R^2 greater than 0.70) for both external input (EI) and external output (EO) function types. For the other three function types, the relationships to total size are different across programming languages, with an R^2 from a low of 0.30 for external interfaces (EIF) in the COBOL sample to 0.65 for the external queries (EQ) in the NATURAL sample.

7.2 Analysis with effort.

For this analysis, the individual size by function type is taken as the independent variable, and the effort as the dependent variable. The results of the regression models on these variables are also presented in Table 9. The relationships are not as strong with project effort, but still of interest with an R^2 over 0.50 for at least one function type in each sample by programming language, but a different function type in each sample. The relationship is stronger in the NATURAL sample for both the EO and EQ function types.

Also of interest is to compare the slopes (e.g. the 'A' coefficient in Table 9) in order to analyze the influence of function types in the productivity of the samples. Table 9 shows the results obtained from the regression analyses.

Function type – independent variable	With respect to functional size			With respect to effort		
	R^2	A	B	R^2	A	B
COBOL (129 projects)						
External Inputs - EI	0.81	1.74	139.63	0.39	15.91	3534.82
External Outputs - EO	0.79	4.24	16.36	0.42	41.09	2219.79
External Inquiries - EQ	0.58	5.10	123.89	0.58	67.29	2429.26
Internal logical files - ILF	0.51	3.22	132.57	0.25	29.81	3448.93
External interface files - EIF	0.30	4.12	222.84	0.08	28.04	4627.16
NATURAL (67 projects)						
External Inputs - EI	0.72	3.22	-34.27	0.53	34.22	-1020.75
External Outputs - EO	0.86	1.60	156.10	0.73	18.23	843.27
External Inquiries - EQ	0.65	5.56	66.48	0.64	68.52	-452.82
Internal logical files - ILF	0.42	4.58	130.97	0.39	54.98	411.90
External interface files - EIF	0.55	18.75	245.23	0.39	196.51	1965.81
C (33 projects)						
External Inputs - EI	0.76	2.58	93.03	0.34	20.19	2673.45
External Outputs - EO	0.77	2.74	129.40	0.57	27.82	2081.04
External Inquiries - EQ	0.42	2.35	318.71	0.19	18.78	4415.94
Internal logical files - ILF	0.63	3.28	165.48	0.34	28.50	2953.33
External interface files - EIF	0.32	5.71	368.78	0.36	71.35	4191.97

Table 9: Estimation models by function type and its coefficients.

8 Summary and next steps.

This paper pointed out the following research questions:

- Is there a relationship between the individual function types and project effort?
- Is there a relationship between the functional profile itself and project effort?
- Is there a better relationship with project effort f a project is close to the average functional profile of a sample?

These research questions have been investigated using the ISBSG 2003 projects repository. It is important to remind the reader that the majority of the projects in the ISBSG repository are MIS applications. The results of the statistical analyses led to distinct estimation models.

From the analysis of the contribution of the individual function types and their impact on project effort:

- According to the results of each function type with respect to the functional size shown in Table 9, we can see that all function types behaved as expected (as each function type grows, the total functional size also grows), but we noted some weak relationships (e.g. have lower regression coefficients):

The external interfaces files (EIF) (R^2 = 0.30) for COBOL projects; the internal logical files (ILF) (R^2 = 0.42) for NATURAL, and the external inquiries and external interfaces files (EQ, EIF) (R^2 = 0.42 and R^2 = 0.32) for C projects. We corroborated for these samples the important contribution of external inputs (EI) and external outputs (EO) to the size estimation models and the small contribution of external interfaces files (EIF) to the dependent variable of functional size.

- According to the results of each function type with respect to the effort shown in Table 9, we can see that the highest regression coefficient for COBOL projects is associated with external inquiries (EQ) (R^2 = 0.58), for NATURAL projects with external outputs (EO) and external inquiries (EQ) (R^2 = 0.73 and R^2 = 0.64, respectively) and for C projects with external outputs (EO) (R^2 = 0.57). For all others, the relationship of size increase to effort increase is relatively weak.

Depending on whether or not a project functional profile is within a reasonable range of the average functional profile for any particular sample, the following results were observed:

- According to the distribution of the weighted function types (Table 5), the external inputs (EI) and the external outputs (EO) in these ISBSG samples are the function types which make the biggest contribution to the project functional size.
- The use of the range of +/– 30% for COBOL and NATURAL projects led to a "natural" selection of a proportion of approximately 80% and 20% of the total number of projects (Table 7). For C projects, in order to have the same proportion (80/20), the range was decreased to +/– 20% because the range of 30% included almost all the projects.
- The projects within the range selected away from an average profile (+/– 30% for COBOL and NATURAL and +/–20% for C – using a selection made on the one function type with the greatest contribution to functional size) lead to regression models very similar to the models for the full sets of projects (Figure 7 and Table 8).
- For those samples of projects selected outside the ranges selected of the average functional profiles, the regression models are different and specific to these samples. These models of the data projects outside of the average functional profiles must therefore be preferred for estimation purposes over the generic models of all projects independently of their functional profiles.

For estimation purposes at the beginning of a project, the functional profile can be identified from the start of the project, and therefore it provides right away the required information to determine which estimation model is the most suitable. In summary, for estimation purposes, the functional profile of a project can be used to improve its estimation by guiding the selection of the estimation model to be used (e.g. the one within the average or the one outside of it).

Is important to note that these results cannot be considered definitive; such studies should be replicated with different and larger data sets. We can conclude, however, that these analyses from diverse points of view (criteria of data selection of the samples, outliers identification, visualization techniques, functional profile, linear regression) provides us with better analytical tools for project estimation based on information that is available very early in the development life cycle.

9 References.

1. Abran, Alain; “Analyse du processus de mesure des points de fonction,” PhD Thesis; École Polytechnique de Montréal; Montreal; 1994.
2. Abran, Alain; Silva, Ilionar; Primera, Laura; “Field studies using functional size measurement in building estimation models for software maintenance,” Journal of Software Maintenance and Evolution: Research and Practice; Num. 14; pages 31-64; 2002.
3. A. Abran, R. Braungarten, R. Dumke, "Web-based Support for White Box Software Estimation," 13th International Workshop on Software Measurement – IWSM 2003, Montréal (Canada), Springer-Verlag, Sept. 23-25, 2003, pp. 218-231.
4. Amzallag E.; Piccioli N.; “Introduction à la Statistique,” Ed. Hermann; Paris 1978.
5. International Software Benchmarking Standards Group; CD R8 Dated - Field Descriptions; http://www.isbsg.org/html/index2.html
6. ISO/IEC 20926:2003 Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual. International Organization for Standards (ISO), Geneva, 2003.
7. Swanson, Gregory; Globus, Lee; “Software Metrics Visualization For a Graphical Programming Environment,” EE-Evaluation Engineering; Online:

   http://www.evaluationengineering.com/archive/articles/0301soft.htm