

# Software Cost Estimation: A Tool for Object Oriented Console Applications

Ghazy Assassa, PhD  
Dept. of Computer Science,  
College of Computer and  
Information Sciences,  
King Saud University.  
ghazy@ccis.ksu.edu.sa

Hatim Aboalsamh, PhD  
Dept. of Computer Science,  
College of Computer and  
Information Sciences,  
King Saud University.  
hatim@ccis.ksu.edu.sa

Amel Al Hussan, MSc  
Computer Dept.,  
Riyadh Girls Colleges,  
Ministry of Education.  
aalhosan@yahoo.com

## Abstract

*Software cost estimation is an important activity during software development. There are many sophisticated parametric models for estimating the size, cost, and schedule of Object Oriented (OO) software projects. Several authors assert that a model's predictive accuracy can be improved by calibrating its default parameters to a specific environment.. The main aim of this paper is to improve the cost estimation accuracy of object oriented console applications at design phase. This paper suggests a variation on the existing formula by calibrating COCOMOII early design model for a specific organization environment using Object Oriented Function Point (OOF) as a size measure instead of standard Function Point (FP) that is used currently in COCOMOII model. This approach benefits from the model calibration and considers OO aspects such as inheritance, aggregation, association and polymorphism. Based on historical projects of an organization, linear regression is applied on the software projects data to relate LOC to software OOF. The obtained regression model is then used to estimate LOC as a step towards estimating effort and cost for new projects. The authors developed a software tool to implement the proposed estimation approach. Experiment with the cost estimation tool showed that it presents a user friendly automatic cost estimation for OO console applications and generates reasonable results.*

## Key words

Object Oriented, Software Cost Estimation, COCOMO, Calibration, Linear Regression.

## 1. Introduction

Recently, Object Oriented (OO) software engineering has emerged as a dominant practice. The growth of OO practices has required software developers and their managers to rethink the way they have been estimating the size, effort and cost of their development projects. To better understand and control these costs, the software organization often uses parametric cost models for software development cost and schedule estimation. However, the cost curacy of these models is poor when the default values embedded in the models are used; the accuracy can be improved by calibrating the parameters against organization's data [1] [2] [3][4]. Accurate estimation of size is vital at early stages of software development. At these stages, it is required to identify a size measure, such as Lines Of Code (LOC), that can be used to estimate effort directly. The problem with

LOC measure is that it is not available at early stages of development. COCOMOII Early Design Model solves this problem by using Function Point (FP) measure to estimate the size of project at this stage. However, FP measure is not suitable to estimate the size of OO projects, because some aspects of OO systems, e.g. inheritance, aggregation, association and polymorphism, are not included in the classical FP count [6].

In this paper, the suggested approach tries - with success - to improve the accuracy of software cost estimation at early design phase for OO software built as console applications. The proposed model is based on the calibration procedure [5] of the well-validated parametric model COCOMOII Early Design Model to adjust its default parameters to organization's specific environment. The suggested cost improvement is based on the count of Object Oriented Function Points (OOF) [7] as a size

measure in conjunction with calibrated COCOMOII. OOFP measure considers all basic concepts of OO systems and gives more accurate size estimation. A main advantage in defining OOFP is that the method could be automated, thus making it easy to recalculate OOFP throughout the life cycle of projects; consequently, supporting re-estimation.

This paper is organized as follows. Section 2 summarize the related work, section 3 describes our methodology and the proposed model and tool architecture for cost estimation. Section 4 discusses the results of applying the proposed model and tool. Finally, section 7 presents the conclusion and suggestions for future studies.

## 2. Background and Related Studies

Estimating the effort, time, and cost of object-oriented software projects is a difficult task. Generating Reliable, repeatable, and accurate estimates can only be achieved through the use of appropriate estimating models that reflect the iterative and incremental nature of object technology. Several authors assert that a model's predictive accuracy can be improved by calibrating (adjusting) its default parameters to a specific environment [1][3][8]. In the work of [9], nine models were calibrated using three databases. The significance of this study is that results greatly improved with calibration. The study of [10] is significant because it analyzed the results of seven cost models to eight Ada specific programs; one of the models was COCOMO. The results of this study, like other studies, showed estimating accuracy improved with calibration. In 1997 COCOMO II Post-Architecture model was calibrated [5] for cost improvement. It is important in any cost estimation model to take into accounts the ability to estimate the size and effort in the early stage of the project development life cycle. Early estimates are essential when using the algorithmic cost estimation models. Allan Albrecht was the first to propose the original function point analysis [11]. Albrecht's function point is computed by counting the following software characteristics: external inputs and outputs, user interactions, external interfaces and files used by the system. Each of these is then individually assessed for complexity and given a weighting value, which varies from 3 (simple) to 15 (complex). Albrecht's function point has been widely used but it has some weakness. Thus, many kinds of function point, such as

TFPCP version [12], 3D Function Points version [13], Feature Points version[14] have been proposed. In recent years, OO technology has emerged, software organizations had to find out alternatives for estimating the size of their development projects [15]. To measure OO software, the main components to be considered are raw functionality of the software, communication among objects and inheritance [15]. Many researchers have proposed methods for adapting FP to object oriented software. Some researchers retain a focus on traditional function point as the output from a count. They relate OO concepts to FP elements; for example [16] considers each class as an internal file while messages sent across the system boundary are treated as transaction. Also [17] treats classes as files, and considers services delivered as transaction. Other researchers develop new measures, tailored to OO software but analogous to FPs. Object Point was proposed as a measure of size for OO software [18]. Predictive Object Point (POPs) were proposed [19] based on counts of classes and weighted methods per class, with adjustment for the average depth of the inheritance tree and the average number of children per class. Methods are weighted by considering their type and complexity, giving a number of POPs in a way analogous to FPs. OOFP was suggested as a measure of size for OO software [6]. Some aspects of OO system (e.g. inheritance, aggregation, association and polymorphism) are not included in the classical FP count. Nevertheless, they contribute to the final size of the system. If the objective of measuring functionality is to estimate the final size of an implementation of a system, and from that the effort and duration of software project, then these aspects should be taken into account. The work of [7] proposed a new Object Oriented Function Points (OOFP) counting procedure which considers all the basic aspects of OO systems. Several regression models were developed to relate LOC to software metrics computed at the design stage such as OOFP. Models with small number of independent variables were preferred for ease of use and interpretation and because the number of data points is not large. The study of [6] compared robust regression techniques with the more traditional least squares line fitting; it was concluded that the robust models do not gain much in explaining the data better. Accordingly, only simple linear regression model will be investigated in this paper.

This paper suggests a software cost estimation tool for OO projects written in C++ or Java console application with no automatic generation of code. The study satisfies the needs of software houses looking for calibrating COCOMOII Early Design model. The key aspect of the proposed cost estimation tool is its simplicity. It hides all complicated formulas from user's eyes and guides the user via easy set of wizards to feed the system with all historical data necessary to calibrate the cost estimation model and to estimate any new OO project.

### 3. Methodology

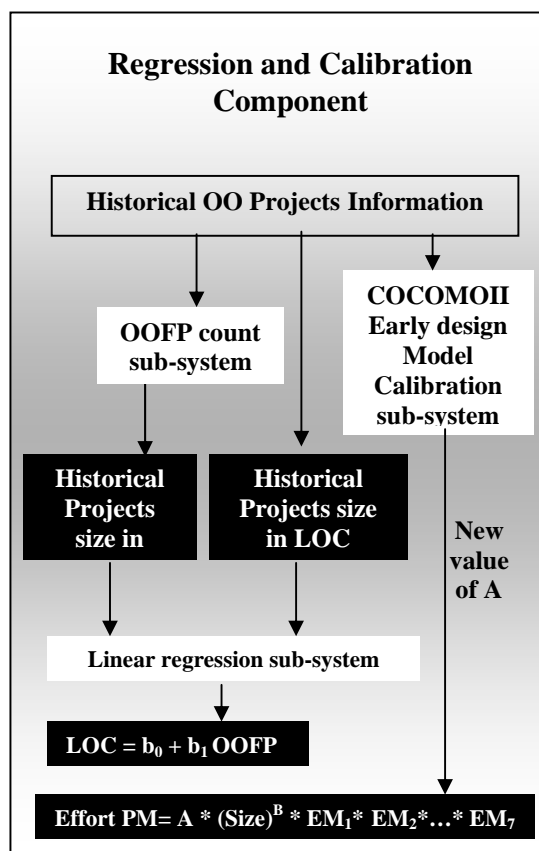


Figure 1.a: The regression and calibration tool component.

In order to implement the suggested approach, a tool was developed with software architecture having two components: the regression and calibration component, and the deployment component. The regression and calibration component represents the model building step where regression and calibration of a set of

historical software projects data are carried out to establish the parameters of the model. The deployment component uses the established model for cost estimation of new projects.

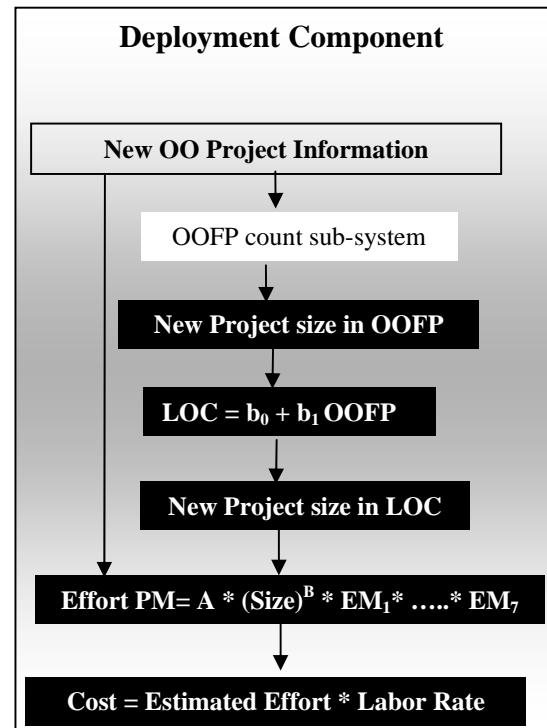


Figure 1.b: The deployment tool component.

Figures 1.a and 1.b show the two components with details as explained hereafter.

#### 3.1. The regression and calibration component

The goal of this component is to establish the parameters of the model, namely A, b<sub>0</sub>, and b<sub>1</sub> via the use of measurements of historical projects. This component includes three subsystems: COCOMOII early design model calibration subsystem, Object Oriented Function Point (OOFP) subsystem, and linear regression subsystem. Details are given below on each subsystem.

### 3.1.1 COCOMOII early design model calibration subsystem

The objective of this subsystem is to determine the parameter A that reflects the organization specific environment and its capability and maturity level. Historical measurements data should provide actual effort and size and the 17 effort multipliers defined in COCOMOII model:

$$\text{Effort PM} = A * (\text{Size})^B * EM_1 * EM_2 * \dots * EM_{17}$$

In order to calibrate (tailor) the above COCOMOII for a specific organization, we need measurements data of at least 10 completed projects [5]. The calibration procedure [5] is adapted with some changes as follows:

- 1- Enter historical data for n completed object oriented software project;
- 2- Calculate the B exponent for each historical project form the equation (this is different from previous studies which set the B exponent to a fixed value of 1.153):

$$B = 1.01 + 0.01 * \sum \text{Exp}_{1..5}$$

where  $\text{Exp}_{1..5}$  are the five scale factors in COCOMOII. Model;

- 3- Calculate the value of Q for each historical project form the new equation:

$$Q_i = (\text{Size})^B * \sum_{1..n} \text{EMs}/n;$$

- 4- Calculate the suitable value A form the equation:

$$A = (\sum \text{PM}_i Q_i) / (\sum Q_i^2).$$

### 3.1.2 Object Oriented Function Point (OOF) subsystem

The goal of this subsystem is to automate the counting process of OOF which is an adaptation of the traditional function points. The counting procedure described in [7] is used to estimate the size of OO consol applications projects. Full details on OOF counting are given in [21] and are summarized in the following algorithms.

#### OO Counting Procedure

Files and transactional functions of non-object approach represent, respectively, classes and methods of OO software. A logical file is a collection of data elements that are visible to all

methods of a class, and transactional functions are the methods in a class. OOF calculation includes logical files and transactional functions counting as explained in the following algorithms.

#### Algorithm for OO logical files counting:

For each software:

- 1- Consider the OO design and extract: number of classes, number of simple data type, number of complex data type, number of inherited simple data type, and number of inherited complex data type;
- 2- Compute Data Element Type (DETs) and Reference Element Data Type (RETs) as:  
DETs = number of simple data type + number of inherited simple data type, and  
RETs = 1 + number of complex data type + number of inherited complex data type
- 3- Compute the complexity of classes (as low, average or high) according to their DETs and RETs count from IFPUG table [3];
- 4- Compute the UOOF for the software classes according to their classes complexity;
- 5- Compute the Value Adjusted Factor (VAF) by evaluating the fourteen General System Characteristics (GSCs), on a scale from 0 to 5, to determine the Degree of Influence (DI) for each of the GSC;
- 6- Add the DI for all fourteen GSCs to produce the total degree of influence (TDI);
- 7- Use the application TDI in the following equation to compute the VAF:  $\text{VAF} = 0.65 + \text{TDI} * 0.01$ ;
- 8- Use the VAF to get the OOF for each class:  $\text{OOF} = \text{VAF} * \text{UOOF}$ .

#### Algorithm for OO transactional functions counting:

For each class:

- 1- Compute the DETs and RETs for each method in a class;
- 2- Compute method complexity (as low, average or high) according to their DETs and RETs count from Traditional FP Counting Procedure (TFPCP) complexity matrix for External Input (EIs);
- 3- Compute UOOF for each method according to method's complexity;
- 4- Compute the VAF and then the OOF for each method, as done above for classes.

#### OOF of the whole software:

Accumulate all computed OOF for all classes and methods to get OOF of the software.

### 3.1.3 Linear regression subsystem

The simple linear regression model [20] is used to relate computed OOF to actual LOC for all historical projects. The purpose of the simple linear regression is to compute the two parameters  $b_0$  and  $b_1$  of the equation:

$$\text{Estimated LOC} = b_0 + b_1 * \text{OOF}$$

The algorithm of getting  $b_0$  and  $b_1$  is shown below.

- 1- Count OOF for  $n$  historical projects [7] [21];
- 2- For  $n$  historical projects, read actual LOC and calculated OOF;
- 3- Compute  $b_1$  from the equation:

$$b_1 = \frac{\sum(\text{LOC}_i * \text{OOF}_i) - (\sum \text{LOC}_i) * (\sum \text{OOF}_i) / n}{\sum(\text{LOC}_i^2) - (\sum(\text{LOC}_i)^2 / n)}$$

- 4- Compute the average of OOF:

$$\text{AVG of OOF} = \sum \text{OOF}_i / n;$$

- 5- Compute the average of LOC:

$$\text{AVG of LOC} = \sum \text{LOC}_i / n;$$

- 6- Compute  $b_0$  from the equation:  
 $b_0 = (\text{AVG of OOF}) - b_1 * (\text{AVG of LOC})$ .

### 3.2. The deployment component

The goal of this component is to estimate the effort and cost of new projects via the use of the established model as discussed in the regression and calibration component. The following algorithm shows the main calculation steps of effort and cost for a new project.

- 1- Enter all OO software properties: number of classes, number of simple and complex data in each class, number of inherited simple and complex data in each class, number of reference parameters in the methods, number of single valued and multi valued association between methods;
- 2- Compute the size of the project in OOF;
- 3- Estimate the size of the project in LOC using the inference regression model:

$$\text{LOC} = b_0 + b_1 * \text{OOF};$$

- 4- Enter the seven Effort Multipliers  $EM_1$ - $EM_7$  for the new project;
- 5- Enter the five exponent factors for the new project (Exp1..5)

- 6- Compute the exponent  $B$  for the new project from the equation:

$$B = 1.01 + 0.01 * \sum \text{Exp}_{1..5}$$

- 7- Estimate the effort of the new project using COCOMOII early design calibrated model:

$$\text{Effort PM} = A * (\text{Size})^B * EM_1 * EM_2 * \dots * EM_{17}$$

- 8- Estimate the direct labor cost of the new project:

$$\text{Cost} = \text{Estimated PM} * \text{Labor Rate of the organization.}$$

## 4. Results

Measurements (cost drivers) were collected for ten completed OO software projects from EHE software house, for which both an OO design model (classes and methods) and the final code were available. All ten projects were developed in the same environment, using the same language, namely, C++ console application. Company profile information included labor rate, work hours in person month (PM) and the maturity level. The tool estimation accuracy is expected to be affected by the number of historical data used to calibrate the model, by the design details available on project classes and methods and by the availability of the effort Multipliers (EMs). Table 1 summarizes the results of calibration using ten historical projects; this led to the determination of the parameter  $A$  that reflects the organization specific environment. Table 2 shows the results of linear regression for the same ten historical projects, leading to the determination of the two parameter  $b_0$  and  $b_1$ .

To determine the accuracy of proposed model, the proposed software tool was used to estimate the size and effort of the available ten historical software projects. Table 3 shows a comparison between the estimated and actual results for the sizes and efforts of the used ten projects; the model parameters were those determined in tables 1 and 2, namely,  $A=2.24$ ,  $b_0=0.237911$ , and  $b_1=0.006415$ . The results suggest that the size difference % has a mean of 37% and a variance of 41%, while the effort PM difference % has a mean of 69% and a variance of 99%. Examination of table 3 shows that projects 4 and 10 yield abnormal data points. When the tool was run after eliminating projects 4 and 10 from

the input data, better results were obtained as displayed on table 4 with model parameters  $A=3.95$ ,  $b_0=0.08292$  and  $b_1=0.00959$ . Table 4 suggests that the size difference % has a mean of

23% and a variance of 31%, while the effort PM difference % has a mean of 71% and a variance of 48%.

Measurements	Projects P1 .. P10											
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10		
Exp1	2.48	1.24	2.48	6.2	2.48	3.72	4.96	4.96	1.24	3.72		
Exp2	3.04	4.05	4.05	4.05	3.04	3.04	2.03	2.03	3.04	4.05		
Exp3	5.65	4.24	4.24	1.41	5.65	5.65	7.07	7.07	4.24	7.07		
Exp4	0	1.1	0	1.1	1.1	0	0	0	0	0		
Exp5	3.12	3.12	3.12	3.12	3.12	3.12	3.12	3.12	3.12	3.12		
B	1.153	1.148	1.149	1.169	1.164	1.135	1.182	1.132	1.126	1.19		
Actual KLOC	0.864	0.248	0.605	1.08	0.322	0.324	0.158	0.6	0.65	0.96		
EM1	1	1	1	1.1	1	1	0.92	0.92	.92	1		
EM2	1.17	1	1.17	1.17	1.17	1	1	1	1	1.17		
EM3	1	0.9	0.9	0.9	1	1.14	0.9	0.9	0.9	0.9		
EM4	1.23	1	1	1.11	1.11	1	0.81	0.81	0.81	0.81		
EM5	1	1.07	1	1.15	1	1	0.95	0.95	1.07	1.24		
EM6	1	1	1.11	1.11	1	1	1	1	1	1		
EM7	1	1	1.05	1.05	1	1	1	1	1	1		
EM8	0.87	0.87	0.87	1.15	0.87	0.87	0.87	0.87	0.87	0.87		
EM9	0.85	0.85	0.85	1	1	0.85	0.85	0.85	0.85	0.85		
EM10	0.88	0.88	0.88	1	0.88	0.88	0.88	0.88	0.88	0.88		
EM11	0.81	0.81	0.81	0.9	0.9	0.81	0.81	0.81	0.81	0.81		
EM12	0.88	0.88	0.88	1	0.88	0.88	0.88	0.88	0.88	0.88		
EM13	0.85	0.85	0.85	1	0.91	0.85	0.85	0.85	0.85	0.85		
EM14	0.91	0.91	0.91	1	0.91	0.91	0.91	0.91	0.91	0.91		
EM15	1.09	1.17	1.09	1.09	1.17	1.09	1.09	1.09	1.09	1.09		
EM16	0.8	0.8	0.8	0.8	0.86	0.8	0.8	0.8	0.8	0.8		
EM17	1	1	1.14	1	1	1	1	1	1	1		
n1..17 EM	0.45	0.323	0.438	1.555	0.656	0.357	0.199	0.199	0.225	0.331	EM1+.....+EM10/n	<b>0.473</b>
Qi	0.4	0.096	0.266	0.518	0.127	0.127	0.053	0.259	0.291	0.451		
(Qi)2	0.16	0.009	0.071	0.268	0.016	0.016	0.003	0.067	0.085	0.203	$(Q1)^2+.....+(Q10)^2$	<b>0.898</b>
effort (PMi)	0.750	1.500	0.750	1.000	0.500	0.5	0.25	0.75	1	0.5		
Qi * PMi	0.3	0.143	0.199	0.518	0.063	0.064	0.013	0.194	0.291	0.225	Q1*PM1+..+Q10*PM10	<b>2.012</b>
A	<b>2.240</b>											

Table 1: Results of model calibration for ten historical software projects.

(GSCs)	Projects P1 .. P10									
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
communications	0	0	0	0	0	0	0	0	0	0
DBprocessing	0	0	0	0	0	0	0	0	0	0
Performance	1	2	1	1	3	3	2	2	1	2
configuration	2	1	2	2	3	1	1	1	1	2
Transaction	3	0	2	2	3	2	1	1	0	4
OLdataentry	5	3	4	3	4	3	5	5	4	5
efficiency	2	3	2	4	4	5	4	4	3	3
OLupdate	5	4	4	4	4	4	4	4	3	5
Complex processing	1	4	1	4	4	4	4	4	3	4
Reusability	2	3	2	4	4	3	3	3	4	4
Installation	2	1	2	2	2	4	1	1	4	3
Operational	1	0	2	1	0	3	1	1	4	2
MultipleSites	0	0	0	0	0	0	0	0	0	0
Changes	0	0	0	0	0	1	1	2	2	2
total degree of influence (TDI).	24	21	22	27	31	33	27	28	29	36
VAF = (TDI * 0.01) + 0.65.	0.89	0.86	0.87	0.92	0.96	0.98	0.92	0.93	0.94	1.01
Total UOOFPI	91	27	33	69	33	24	25	59	65	143
OOFPI = VAF * UOOFPI	80.99	23.22	28.7	63.5	31.68	23.5	23	54.9	61.1	144.4
OOFPI <sup>2</sup>	6559	539.2	824.26	4030	1004	553	529	3011	3733	20860
Actual KLOCi	0.86	0.25	0.61	1.08	0.32	0.32	0.16	0.6	0.65	0.96
KLOCi * OOFPI	69.98	5.759	17.4	68.6	10.2	7.62	3.634	32.9	39.72	138.7
$b1 = \frac{\sum(KLOC_i \cdot OOFPI_i) - (\sum KLOC_i)(\sum OOFPI_i)/n}{\sum(OOFPI_i^2) - [\sum(OOFPI_i)]^2/n}$										0.006415
$b0 = (AVG \text{ of } LOC) - b1 (AVG \text{ of } OOFPI)$										0.237911

Table 2: Results of linear regression for ten historical software projects.

Project #	Calculated OOFPI	Actual Size in KLOC	Estimated Size in KLOC	Difference %	Exponent B	EM1	EM2	EM3	EM4	EM5	EM6	EM7	Actual PM	Estimated PM	PM Difference %
P1	80.99	0.864	0.757	12%	1.1529	1	1	1	0.83	0.87	0.87	1	0.75	1.021	36%
P2	23.22	0.248	0.387	56%	1.1475	1	1.07	1	0.83	0.87	0.87	1	1.5	0.506	66%
P3	28.71	0.605	0.422	30%	1.1489	1	1	1.29	0.83	0.87	0.87	1.1	0.75	0.741	1%
P4	63.48	1.08	0.645	40%	1.1688	1.33	1.15	1.29	1	1	1	1	1	2.647	165%
P5	31.68	0.322	0.441	37%	1.1639	1	1	1	0.83	0.87	1	1	0.5	0.624	25%
P6	23.52	0.324	0.389	20%	1.068	1	1	1	0.83	0.87	0.87	1	0.5	0.513	3%
P7	23	0.158	0.385	144%	1.1818	0.83	0.95	1	0.83	0.87	0.87	1	0.25	0.360	44%
P8	54.87	0.6	0.590	2%	1.067	0.83	0.95	1	0.83	0.87	0.87	1	0.75	0.632	16%
P9	61.1	0.65	0.630	3%	1.1264	0.83	1.07	1	0.83	0.87	0.87	1	1	0.742	26%
P10	144.43	0.96	1.164	21%	1.1896	1	1.24	1	0.83	0.87	0.87	1	0.5	2.091	318%

Table 3: Comparison between the estimated and actual results for the sizes and efforts of ten historical software projects.

Project #	Calculated OOF	Actual Size in KLOC	Estimated Size in KLOC	Size Difference %	Exponent B	EM1	EM2	EM3	EM4	EM5	EM6	EM7	Actual PM	Estimated PM	PM Difference %
P1	80.99	0.864	0.859	1%	1.1529	1	1	1	0.83	0.87	0.87	1	0.75	2.086	178%
P2	23.22	0.248	0.306	23%	1.1475	1	1.07	1	0.83	0.87	0.87	1	1.5	0.682	55%
P3	28.71	0.605	0.358	41%	1.1489	1	1	1.29	0.83	0.87	0.87	1.1	0.75	1.083	44%
P5	31.68	0.322	0.387	20%	1.1639	1	1	1	0.83	0.87	1	1	0.5	0.945	89%
P6	23.52	0.324	0.308	5%	1.1653	1	1	1	0.83	0.87	0.87	1	0.5	0.631	26%
P7	23	0.158	0.303	92%	1.1818	0.83	0.95	1	0.83	0.87	0.87	1	0.25	0.478	91%
P8	54.87	0.6	0.609	1%	1.1818	0.83	0.95	1	0.83	0.87	0.87	1	0.75	1.090	45%
P9	61.1	0.65	0.669	3%	1.1264	0.83	1.07	1	0.83	0.87	0.87	1	1	1.402	40%

Table 4: Comparison between the estimated and actual results for the sizes and efforts of eight historical software projects.

Some comments on the proposed cost estimation tool are due:

- The calibration is affected by the number of historical software projects. As the number of historical software projects increases the model accuracy increases; that is because the parameters A, b0 and b1 would be statistically based on a larger population. A minimum of 10 well documented historical projects are to be used [5].
- The estimated effort is affected by the estimated values of the seven effort multipliers (EM1..EM7). We have collected the seventeen effort multipliers (EM1..EM17) for the historical software projects, but some of the seven effort multipliers combine more than one of the seventeen EMs.
- When collected measurements are abnormal, the calibration values and regression model parameters are unfavorably affected. Abnormal data should be excluded.
- The accuracy and completeness of data collected for calibration has a direct impact on the overall model. As the OO design becomes more stable with more information on the involved classes and methods, re-estimation could be repeatedly carried out to improve the accuracy of estimated effort and cost.

## 5. Conclusion

The proposed cost estimation model combines the benefits of different models to construct a new cost estimation model for OO console applications projects. The model takes into

consideration the software OO aspects, such as inheritance, aggregation, association and polymorphism. The model uses Object Oriented Function Point (OOF) as a size measure and employs a linear regression to relate computed OOFs and actual size in LOC. The parameters of the model were determined via calibration and regression of ten historical OO projects from a specific software house, for which both the OO design model and the final code, in C++ console application, were available. Once the parameters of the model are determined, the transformation formula is used to estimate the size in LOC of new OO software at design phase.

An automated calibration and estimation tool was developed to handle the calibration and regression processes. The key aspect of the proposed cost estimation tool is its simplicity. It hides all complicated formulas from user's eyes and guides the user via a set of easy wizards to feed the system with all historical data to calibrate the cost estimation model and to estimate any new OO project. Experimentation with the developed tool shows that it yields reasonable estimation. We have plans to extend the model and the tool to incorporate, in addition to the currently available OO console applications, windows applications including visual components and automatic generated code.

## References

- [1] Kemerer, Chris F. May 1987. "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, pp. 416-429.

- [2] Van Genuchten, Michiel and Hans Koolen. 1991. "On the Use of Software Cost Models." *Information and Management* 21: 37-44.
- [3] Andolfi, M. August, 1996. "A Multi-criteria Methodology for the Evaluation of Software Costs Estimation Models and Tools." CSELT Technical Reports 24: 643-659.
- [4] Boehm, Barry W. *Software Engineering Economics*. Englewood Cliffs, N.J., Prentice-Hall (1981).
- [5] Bernheisel, Wayne A. 1997. "Calibration and Validation of the COCOMO II.1997.0 Cost/Schedule Estimating Model to the Space and Missile Systems Center Database." Unpublished masters thesis. Dayton, OH, Air Force Institute of Technology.
- [6] Caldiera, G., G. Antoniol, R. Fiutem, C. Lokan," Definition and Experimental Evaluation of Function Points for Object-Oriented Systems",IEEE, 5th. International Symposium on Software Metrics, March 1998, pp. 167
- [7] Ram D. J. and S. V. G. K. Raju, "Object Oriented Design Function Points", IEEE, The First Asia-Pacific Conference on Quality Software (APAQS'00), October 2000, pp. 121
- [8] Jones, C., "Software Cost Estimation in 2002", *CrossTalk The Journal of defense Software Engineering*, Jun 2002, pp. 4-8.
- [9] Thibodeau, Robert. *An Evaluation of Software Cost Estimating Models*. Huntsville AL: General Research Corporation, 1981.
- [10] IIT Research Institute. *Test Case Study: Evaluating the Cost of Ada Software Development*. Langham MD: IIT Research Institute, 1989.
- [11] Albrecht, A.J., "Measuring Application Development Productivity", *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, Oct. 1979, pp. 83-92.
- [12] IFPUG. *Function Point Counting Practices Manual Release 4.1*. International Function Point Users Group, Westerville, Ohio, 1999.
- [13] Fioravanti, F., and Nesi, P., "Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object-Oriented Systems", *IEEE Transactions On Software Engineering*, Vol. 27, No. 12, December 2001
- [14] Jones, C., "Applied Software Measurement", McGraw-Hill(1996).
- [15] Teologlou, G., "Measuring object oriented software with predictive object points". In 10th Conference on European Software Control and Metrics, May 1999. Available at <http://www.escom.co.uk/publications>.
- [16] Whitmire, S., "Applying Function Point to Object Oriented Software Models", 1993, In : *Software Engineering Productivity Handbook*, pp. 229-224, McGraw-Hill.
- [17] Schooneveldt, M., "Measuring the size of Object Oriented Systems", In: *Proce-2nd Australian Conferences on Software Metrics*, Australian Software Metrics Association, 1995.
- [18] Sneed, H., "Estimating the development costs of object oriented software", in: *Proceeding of 7th European software control and metrics conference*, Wilmslow, UK., 1996.
- [19] Mehler, H. and A. Minkiewicz, "Estimating Size for object oriented software", 1997, in: *Proceeding of ASM'97 Application in Software measurement*, Berlin.
- [20] Chapter - 13: Simple Linear Regression at [www.ilstu.edu/~achoudh/MQM100\\_Simple\\_Linear\\_Regression.pdf](http://www.ilstu.edu/~achoudh/MQM100_Simple_Linear_Regression.pdf)
- [21] Al-Hossan, A.A., "Cost Estimation Model for Object Oriented Software Console Applications". MSc Project, Department of Computer Science, College of Computer and Information Sciences, King Saud University, 2005.