Evaluating Object Oriented Estimation Models

EDUARDO FERNÁNDEZ-MEDINA, MARIO PIATTINI
Departamento de Ingeniería informática.
Universidad Antonio de Nebrija.
C/ Pirineos 55, 28040, Madrid
Tfno: +34 91 452 11 00
Fax: +34 91 311 66 13
SPAIN

Abstract. In this paper some effort estimation methods for object oriented systems proposed in recent years are summarized. Moreover, we propose a general technique of effort estimation that is applicable to any paradigm of computer systems. Along these lines the huge importance of good estimation in the first stages of the computer systems life cycle is justified.

Key-words: Object Oriented Systems; Functional Size; Function Points; Estimation Methods; Measurement; Estimation; Cost; Effort; Complexity Metrics; Empirical Study.
CSCC'99 Proceedings: Pages 7201-7209

1 Introduction

Projects are often built with a cost higher than the 200% from the budget ([9]). Therefore, an estimation close enough to the size, the effort and the cost in the first phases of the life cycle is a fundamental factor in order to achieve a budget of the system before constructing it and preparing a temporary planning of the development activities ([5]). Thus, if the estimations are good, we can solve the historic problem of computer science, that is, delivering the systems in time, and without exceeding the budget ([3]). In ([10]) is mentioned the dangers of using wrong development cost effort estimation in the computer systems:

- When a cost estimation is low (infra-estimation) the directors can approve new projects which will require more resources (the resources are limited), so that it will fail the unitary contribution expected, and finally the estimations will be doubtful.
- Instead, when there are high cost estimations (overestimation) , the directors can refrain from developing new projects and thus have more benefits. This shows that some unsuitable cost estimations of an information system may have a harmful impact in the organization.

As the author points out in ([2]), when the development advances, the estimations should have a higher degree of accuracy, because they are building with part of the system already created.

Traditionally, the analysis of the system functionality has been used to estimate the effort, that is to say, it has analyzed the characteristics of the system from the point of view of what it offers the user ([4]). This can measure the complexity of certain external factors like departures, entries and user inquires, and with other internal factors such as the data stores and the interfaces with other systems (in Albrecht's Function Points), otherwise measuring the input and output transactions, and the processes (in Mark II's Function Point). Moreover it uses certain factors of the technology state and of other aspects of the future systems. Although these methods certainly are used successfully for structured developments, they do not seem entirely appropriate for object oriented systems. Thus, for example, in ([11]) the author thinks that "*it seems reasonable that in objects oriented systems, the estimate should be based on the analysis of an objects model, and not on the traditional criteria for calculations the function points*", because the function points had been created for other kinds of systems long ago, and perhaps they can not be adapted to the new technologies, and thus create a considerable error in such estimates.

In the following paragraphs, we give a summary of the present-day techniques of effort estimation for object oriented systems.

2 Estimation by analogy

This model is applied to the beginning of the developing life-cycle, when only the requirements of the system are available to us. The object oriented concepts have not been applied to the system so far, so that we can use this method in a project developed with any traditional or object-oriented methodology.

Since we have only the system requirements, the estimates will have a big margin of error, but it can help us know the wingspan of a new system.

In order to be able to apply this technique, we must have a spacious base of projects. This approach allow us to realise estimations of the size and effort when we have only the system requirements, but such an estimate will be only an aid, because we are based on unstable data like the system requirements.

As is indicated in ([2]), this approach consists of the following steps:

[1] The system is divided in several independent parts of relatively small size.

[2] We compare the divisions made with other already existent system divisions. We will compare certain parts of the new systems with parts of varied already existent systems. It is possible that we have other parts that are not covered by any of the existing systems. All the possible combination divisions are:
   a) Zone covered by an already existent system. This division will be compared to an already constructed system.
   b) Multiple covered zone. This area can be compared to certain parts of already created varied systems.
   c) Non-covered zone. There is no system that can help to estimate this zone.

[3] The size of the new system proposed is determined. The size will be estimated in function points, and the result will be the sum of the size estimation for each of the parts of the system.

Estimation of the size =
Sum of the estimated size of each part covered by only one existent system +
Sum of the estimated size of each part covered by different existent systems +
Sum of the estimated size of each part covered by no existent system

We are going to see how we can realize these estimates:

a) Zone covered : It estimates with the function points amount of the already existent system corresponding to the percentage of the covered zone.
b) Multiple covered zone: We estimate the size realizing the average of the sizes estimated by each existent system.
c) Non-covered zone: This estimate must be realized only when has been estimate the zone covered. If the covered percentage is P%, the non-covered is Q% (P+Q=100) and Z is the amount of function points estimated for the covered zone, the resulting to estimate the covered zone will be (Z/P)*Q.

Consider the example of figure 1, where a new system can be covered for other systems that already have been constructed, and its size in function points is known.

- Estimate of the zone covered (solely) by A.
  10% of A → 1000 FP.

- Estimate of the zone covered (solely) by B.
  20% of B → 1500 FP.

- Estimate of the zone covered by C.
  30% of B → 1500 FP.

- Estimate of the multiple covered zone.
  10% of A → 1000 FP.
  20% of B → 1500 FP.
  2500 FP / 2 → 1250 FP

- Estimate of the zone covered by no system.
  65% of the system supposes 1000 + 1500 + 1500 + 1250 = 5250 FP.
  Then 100% supposes 8077 FP.
  Then, the non-covered part, the 35% supposes 2827 FP.

- Estimate of the complete system.
  It is the sum of the estimates for the covered and non-covered zones: 3450+1150 = 4600 FP.

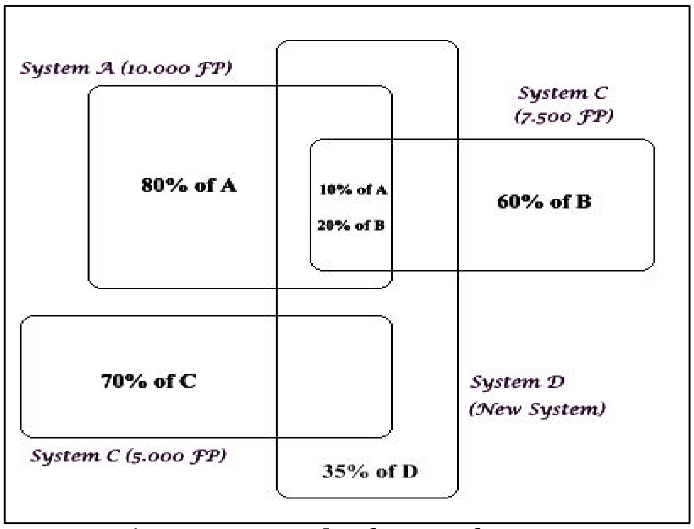

Figure 1. Example of parts of a system.

After the size of the new system has been estimated in function points, we must express the size in terms of development effort and cost. This can be done through techniques of regression that relate the size in function points with effort and size with development cost.

In ([2]), the author proposes a modification of this technique, that is, instead of estimating the size in function points, estimating directly the effort or the development cost. We can do it considering, instead of the function points of the already existent systems, the effort that has been necessary to construct it. So we would realize the same process as before, and we will obtain the result in effort units.

2.1. Strong and weak points of the method of estimation by analogy

This estimation technique of function points has the following weaknesses: The requirements must be very complete and well specified, the comparison between a system and parts of other systems is very ambiguous and the technique does not contemplating the reusing.

Among the more favourable factors, we show the following: It obtains estimates of the functional size of the system very soon, and with very little cost and realizing the divisions, and comparing with other systems divisions already created, we can see what parts of the system can be reused.

3 Estimation through "Object Points"

Traditionally, the way for system sizes have been estimated through analysis of the functionality that the system contributes to the user and through empirical estimations of the source lines of code of the final product. Although the function point analysis can be appropriate, techniques of estimation created explicitly for the object oriented paradigm should be used ([11]).

Regarding the source lines of code, it is necessary to mention that although it is a very utiliced estimating method, it has certain problems with the estimates realized. One of the most important problems of the source lines of code is cited in ([1]): «*The software involves a huge percentage of the fixed cost that is not depending on the code. The more powerful the programming language is the less SLOC are needed, but the requirements, specification, documentation and other elements tend to have fixed costs. This mean that the SLOC as measures of systems size is incorrect because we are measuring the global size in function of the only size of a phase of the development, and besides that is very unstable*». This problem is bigger with object oriented systems, as the phase of encoding has less weight in the development because systems dedicate more effort to the analysis and design phases, with the purpose of obtaining most robust systems with lesser maintenance cost. In ([12]) is confirmed this, denoting that the measure of the code lines to measure the size of object oriented systems is not appropriate, because there exists a displacement of personnel, that in traditional systems was dedicated to encoding work, whereas the object oriented systems are devoted to analysis tasks. Therefore, as in ([11]) is indicated, the orientation to objects has obliged developers to readjust their thought in diverse aspects, and particularly, they have had to change the form of measuring and the form of estimating certain characteristics of these systems.

Now, i am going to describe metrics called "object points" that combine some of the characteristics of the metric oriented to functionality and oriented to source lines of code:

- Like lines of code, the object points have large correlations with the effort, and their calculation can be automated from the product after the project has finished.
- Like the function points, the object points incorporate an understanding of the product behaviour, and they offer the appraisal of the system from the user's point of view.

3.1. Foundation of the objects points metric

The first thing to see how we can estimate the object oriented system size will be to examine the metrics that are available, and observe if they are useful in some aspects. We can see in figure 2 the relationship between the form of obtaining function points (Mark II) and the form of obtaining object points.

The object points combine several metrics that have already been analyzed and proclaimed in the existing bibliography, like ([7]) or ([6]). The metrics that interest us must be estimated early in the development. The only ones that carry out that property are those that analyze the structure of the classes and the structure of the associations between them. The measures that in ([11]) are considered are the following: Number of top level classes (TLD), Weighted methods per Class (WMC), Average depth of class in hierarchy tree (DIT) and Average number of children per class (NOC).

We can see an example of calculation of those metrics in Figure 3.

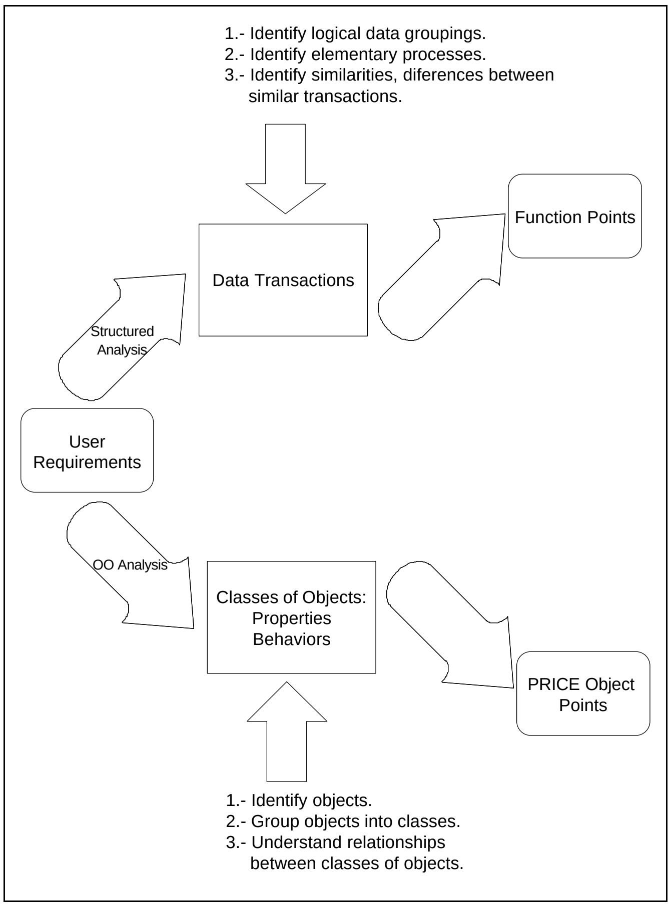
1.- Identify logical data groupings.
2.- Identify elementary processes.
3.- Identify similarities, diferences between similar transactions.
Function Points
Data Transactions
Structured Analysis
User Requirements
OO Analysis
Classes of Objects: Properties Behaviors
PRICE Object Points
1.- Identify objects.
2.- Group objects into classes.
3.- Understand relationships between classes of objects.

Figure 2. Procurement of Function Points and Object Points

Now it is necessary to establish a procedure to assign complexity to the methods. In order to do so, we can distinguish the following kinds of methods: Constructors, destructors, modifiers, selectors and iterators.

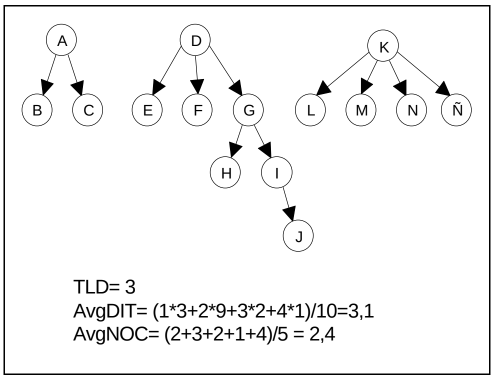

Figure 3. Ejample of calculation of metrics.

We will distinguish three kinds of categories in function of the complexity: low, medium and high. Thus we can assign complexity ranks for each kind of method (constructors, destructors, modifiers, selectors and iterators) and so calculate the WMC based on that complexity. Also it is necessary to establish some rules to calculate the parameters number and properties implicated in the method.

Once we have arrived at this point, the Object Point metric is formed in function of the preceding metrics. For the calculation of the object points is important to take into account that the principal metric is WMC.

The way to calculate the object points is the following:

$$\text{NumberClass= TLC + TLC*((1+NOC) * DIT)}^{1,01} + \text{ABS(NOC - DIT)}^{0,1}$$
UnadjustedOP= NumberClase*WMC
OPAdjusted= f(NOC,DIT)
OPs=UnadjustedPO*(1+AdjustedOP)

We can see that the object point adjust is in function of two metrics. An experiment achieves the next index:

f(NOC,DIT)

NOC / DIT	0-2	3-5	5-
0	0	8	12
1-4	5	10	15
4-	14	17	20

The problem is that this technique is too new, and there is not sufficient data we can take as reference.

3.2. Limitations of the objects points

There are certain aspects of much importance that in ([11]) are not considering, or are not addressing in an ‘appropriate’ manner in the calculation technique. These aspects are the following: The reuse of already created components is not considered and the calculation technique leans completely upon one index that can mark the estimation, and that can make the estimations unstable.

One possible solution for the first problem can consist in the employment of the metrics in a separate way, that is to say, to obtain some metrics TLC1, WMC1, DIT1 and NOC1 for all the non-reuse components, and other metrics TLC2, WMC2, DIT2 and NOC2 for all the reused classes. Afterwards, we can obtain OP1 and OP2 like this:

OP=OP1 + r*OP2.

For the total calculation, we must bear in mind a) the object points associated with the non-reused classes, and b) the object points associated with the reused classes multiplied by a factor r (r<1), because these classes will have analysis, design, maintenance, and integration costs, but they do not have encoding cost. The factor r would be calculated empirically from a previously built historic base of projects with reused classes and non-reused classes.

In the calculation of the object points we use a function which depending of the worth of NOC and DIT, returns us a complexity factor that will be multiplied by the unadjusted object points. These values have been obtained by some empirical proofs and are in principle applicable to any environment. A feasible solution consists in applying a multilineal regression method with which we obtain the best values for the index. For this, it is necessary to have a big data base, so that those values can be reliable.

4 Estimating of Effort for Object Oriented Systems

In ([12]) is affirmed it is necessary to have new metrics to measure object oriented systems aspects, and presents new size and complexity measures to evaluate and predict the effort of development. This author assumes that the complexity and the size are significantly related to the effort also in the object oriented systems. In order to analyze the complexity and the size we consider three levels: methods, classes and system. For each level, he defines metrics according to other low level metrics.

4.1. Levels in the method metrics

An improvement refereed to traditional metrics is also consider the method interface complexity/size, namely, the complexity or size of the parameter list of the methods. The reason for which it adds complexity or size aspect to the methods, is that they can be effective to evaluate the adoption and reuse cost, and to predict the implementation cost. The method complexity is defined in this way: $MC_m=W_{MICm}\ MIC_m + W_m m$

Where: W_{MICm} and W_m are weights that depend on the metrics adopted, MIC_m is the complexity of the method interface and m is the metric adopted by the method evaluation. It can be one of the already known, like the MacCabe's cyclomatic complexity (m=Mc), of the number of source lines of code (m=SLOC) of Halstead's metric (m=Ha) of Nesi and Querci's metric (m=MS).
The weights are obtained by applying measures through the validation process that present different values in different phases of the life cycle (evaluating a set of similar projects). MIC_m is a forecast factor, as it contributes information when we only have a method prototype. MIC_m is estimated to take into account the sum of the complexity and size of each method parameter.

Then, there are some method complexity metrics in function of the basic metric than we use. Thus, the metric MC_m is more complete than a simple size or complexity metric. Obviously, these metrics can be useful also to obtain better non-object oriented system evaluations. The defined metrics are MC_{Mc}, MC_{SLOC}, MC_{Ha} and MC_{MS}.

4.2. Levels in the class metrics

Using the MCabe, Halstead and SLOC metrics, the definition of the complexity or size for each class in function of the methods that it has is immediate. With traditional systems (non-object oriented) this metric is defined like this:

$$CM_m = \sum_{i}^{NM} m(i)$$

Where NM is the ethod number by class and m is the base metric that we use.
This way, we define the class metrics: CM_{Mc}, CM_{Ha} y CM_{SLOC}.In the literature, it has been demonstrated that CM_m is not appropriate to evaluate object oriented projects, since it does not consider object oriented aspects. In function of MC_m it has defined the next metric: Class method complexity:

$$CMC_m = \sum_{i}^{NM} MC_m(i)$$

This metric solely contemplates the complexity of the class methods. It also can be expressed like this:

$$CMC_m = W_{MICm}\sum_{i}^{NM} MIC_m(i) + W_m\sum_{i}^{NM} m(i)$$

This metric is able to obtain estimations before the class implementation is available. This is due to the existence of the MIC_m metrics. The problem is that these metrics do not contemplate object oriented aspects yet, and we must consider both attributes and methods, both inherited and locally defined. These factors must be considered in order to know the cost or gain of the inheritance.
The class complexity is defined like this: CC=CCL+CCI
Where CCL corresponds to class local complexity, and CCI corresponds to class inheritance complexity, and through these metrics we must express the attributes and methods complexity, both local level and inherited. Thus, class complexity can by expressed in this way: $CC= w_{CACL}CACL + w_{CMCL}CMCL + w_{CACI}CACI + w_{CMCI}CMCI$
Where CACL corresponds to class complexity by local attributes, CMCL corresponds to class complexity by

local methods, CACI corresponds to class complexity by inherited attributes and CMCI corresponds to the class complexity by inherited methods.These weights will be obtained through a multilinear regression analysis where we know class effort. In normal situations, the class complexity weights or coefficient by inherited attributes will be negative, since inheritance of attributes mean less system complexity or size, and then less effort.
We can express CACL and CMCL like this:

$$CACL=\sum_{i}^{NAL} AC_i \quad CMCL=\sum_{i}^{NML} MC(i)$$

Where NAL and NML are the local attribute number and local method number respectively. It is able to define in an analogous way CACI and CMCI. If we generalize for any of the already known metrics, we can have the following: $CC_m = w_{CACLm}CACL_m + w_{CMCLm}CMCL_m + w_{CACIm}CACI_m + w_{CMCIm}CMCI_m$
It has defined a set of metrics that will have some weights in function of the base kind of metric that we utilize. The defined metrics are: CC_{Mc}, CC_{Ha}, CC_{LOC} y CC_{MS}.

4.3. Level in the system metrics

We can consider several levels within the system: a set of classes organized in one or varied class trees, a set of procedures and functions in C++, a set of global definitions of types, structures, etc. and a set of variable declarations.
In the same way, we can define metrics for each level. The defined metrics are: NCL (number of system classes), NSF (number of system functions/procedures), NGD (number of global definitions), MGV (number of global variables), Etc.

The system complexity is defined in the following way:

$$SC_m=w_{CCm}\sum_{i}^{NCL} CC_m(i) + w_{FCm}\sum_{i}^{NSF} FC_m(i) + w_{GDCm}\sum_{i}^{NGD} GDC_m(i)+w_{GVCm}\sum_{i}^{NGV} GVC_m(i)$$

Where FC_m, GDC_m y GVC_m are complexity/size metrics that measure non-object oriented aspects, namely, functional aspect, global definitions and variables declarations respectively. Then, SC_m is precise equation of the system complexity from all the aspects, functional, data, object oriented relationships, contrarily to the reverse that the traditional metrics.

4.4. Limitations of the estimation system

A set of general metrics is offered, namely, equations that are in function of some weights, and these weights depend on the base metrics that we use and of the environment, that is to say, of the set of projects used to realize the multilinear regression analysis. Because of this, every time we want to use this prediction system, we must obtain a value for all the weights realizing a regression analysis with data of the specific environment.

5 Effort Parametric Estimation: An Object Oriented Model

This technique is oriented to the estimating of the development effort for traditional systems transforming the analysis documentation in an object oriented model, expressed in natural language. Therefore, this technique is easily extensible to systems whose requirements analysis has already been constructed with an object oriented methodology.
In ([8]) it is demonstrated that the development effort can be forecast during the conceptualization of a system using parametric analysis. Using an abstract object oriented description of software, it is possible to develop an empirical model of the software, that is useful for much phases of the development. The parametrical model has operations and interfaces that are extracted from a high-level description of the system. Several functional formularies of the estimations were developed and evaluated harmoniously.
Of all the models utilized, it was found that with simple models of regression, using the object variable transformed exponentially, behaved much better than any others.

5.1. Introduction

The objective of the author in ([8]) is to examine the possibility of identifying cost estimate parameters from an abstract description of information systems. An estimation model based on an abstract description of the systems can be used to plan the develop and control process. An abstract model derived from an object oriented frame was treated to identify system characteristics that are able to have a direct relation with the development effort. The variables considered were only those measurable in the first stages of the development.

5.2. Methodology

The process consists in the use of regression analysis to discover empirical relation between systems characteristic and their costs. If the relation can be demonstrated, then it is possible to use those characteristics as parameters in a cost estimation model.
A lot of the present-day estimation techniques are non-parametric, and require the estimation of the number of lines of code to predict the time required to complete the project. Then, if we must estimate first the lines of code

and then use this estimation to calculate the effort, we are charging with a high degree of error. The parametric models are less frequent, and the more important example of parametric model used for the estimating of the development effort is the function points analysis (FPA).

5.3. An abstract model of software development effort

This method attempts to extract effort estimation metrics in the first phases of the life-cycle. The principal difference between this model and the function points model is that the first one is most abstract than the function point model.
An object oriented model of a system is composed of objects, interfaces and operations between those objects. These components are abstractions of real world objects, operations and interfaces.
It seems reasonable that the development effort to create a system is in function of the objects, operations and interfaces number that compose the system: **Man hours = f(objects, operations, interfaces).**
The main advantages of the employment of this approach of object oriented effort are:

- An object-oriented model is an objective representation of the proposal system.
- An object-oriented model promotes a better understanding of the system, by using a methodology for articulating system functions and requirements.
- The model can be built in different levels of abstraction. When the system is in the conceptual stage, it can be viewed at a high level of abstraction. As development progresses, the systems are expressed with increasingly lower levels of abstraction, and the cost analysis can be refined.
- The model does not suggest a particular implementation of the system.
- It may be possible to integrate an object-oriented cost estimation approach with advanced system design and development object oriented methodologies.

5.4. Election of the kind of model

There is a theory that indicates that this produces a deseconomy of scale in the software development effort. This means that when the number of objects is raising, the effort is raising with a bigger proportional factor. This is because, if the project size is increasing, then the time necessary to interface, integration and proofs will be bigger to.
Therefore, although the first focus will consist in the employment of a non-linear model, we will use a linear analysis to begin with.

5.4.1. Results for linear model

The linear effort model in function of the objects number, the operations number and the interfaces number is the following:

$$\mathbf{Hours = B_0 + B_1*OBJ + B_2*OP + B_3*INT + e}$$

Where: Hours = Duration of the programming task, OBJ = Number of distinct objects identified with the programming task, OP = Number of distinct operations identified with the programming task, INT = Number of distinct interfaces identified with the programming task and e = random error associated with the model.
When the regression analysis was made, the results obtained with this model were satisfactory. We also analyzed a linear model with only the objects number:

$$\mathbf{Hours = B_0 + B_1*OBJ + e}.$$

By chance, the results obtained were better with the reduced model than with the complete model.

5.4.2. Results for nonlinear model

There are three no linear models, logarithmic, semi-logarithmic and exponential. The better results were obtained from the exponential model, which is formulated with the following equation: $\mathbf{Hours = B_0 + B_1*e^{OBJ} + error.}$

The results confirmed the theory of the existence of a scale deseconomy in the effort estimation in function of the objects number. This indicates that the more appropriate model is a non-linear model.

5.5. Conclusions

The results of this study show that it is possible to base forecasts of programming effort on a very abstract object oriented model of the system. The exponential model exhibits the best fit and accuracy of all the models.

5.6. Limitations

It is necessary to have precaution in generalizing this results to other data sets. The models developed in this study would probably require further re-estimation in order to be used in other settings and for larger projects.

6 Method of effort estimation based in «Effort Point»

The technique presented in this paper is based on the thought of the author of ([11]) and in the method of parametric estimation offered in ([8]). This is a general technique that tries to estimate the development effort or

cost in object oriented systems using a general formula adaptable to any particular environment.
The idea is not to estimate the objects points (or function points) to transform them in effort or cost (commiting two errors, in the moment of calculating the object or function points, and next when we estimate the development effort), but that we will estimate directly the data that really interests us, that is to say, the development effort. This way, the concept of functional size disappears as such, ceasing to be used as an unit of measure function points or object points, and proceeding to measure the system size as the effort necessary to realize it, that is, a concept highly correlated with the function points (although on some study cases ([4]) have not been able to confirm the existence of such a correlation).

In this case, the measure of estimation will be the effort points, that consist of each unity of people-month of work in the development of a system. The formula to expound will relate all the outstanding metrics discussed in the paragraph 3, but further testing is needed to retain the better model. There are a lot of models, linear and nonlinear. A linear model can be the next: $\text{Effort}_i = \mathbf{A}*\text{NumberClass}_i + \mathbf{B}*\text{WMC}_i + \mathbf{C}*\text{NOC}_i + \mathbf{D}*\text{DIT}_i + \text{error}_i$

This is a model that supposes a linear relationship between the metrics utilized and the effort realized to construct an information system. In the formula there appear a series of unknown variables, that is necessary to know (A, B, C, D and error_i). The form of knowing these unknown variables consists in applying a resolution method of equation systems where there is an equation by each project already created, otherwise in applying directly a multilinear regression analysis, where we obtain the correlation index, and moreover the variance of the variables.
We can also consider the theory exposed in ([8]) about the deseconomy of scale, then we must consider several nonlinear models like these:
$\text{Eff}_i = \mathbf{A}*\text{NClass}_i^{D} + \text{WMC}_i^{E} \ (\mathbf{B}*\text{Noc}_i + \mathbf{C}*\text{DIT}_i)^{F} + \text{err}_i$
$\text{Eff}_i = \mathbf{A}*\text{NClass}_i * \text{WMC}_i * (\text{Noc}_i*\text{DIT}_i)^{D} + \text{err}_i$
In principle, we need various models to realize some proofs, and thereafter adapt the model that produces less error in the estimations, and therefore, with the model that produces better correlation between the metrics and the real effort. As the equation system can not be linear, we can utilize some technique of approximation (like the genetic algorithms or the blind search), in such a way that we obtain the unknown variables by minimizing the sum of the error charged in all the equations.

Until now, we have the deduced mystery variables, and we can apply the formula to the data of the new system, to obtain a nearness of the system effort. Besides, by using the error charged in the equations, we can have a nearness of the maximum error that we charge with ours estimation. As soon as we finish the construction of our system we already know the necessary real effort. Then, those data will form part of the equation system, and will permit further refinement of the calculation of the unknown variables for the estimation of the effort of future projects.

Example

We suppose that we have a data base with relative information about 6 already realized projects, and we know both effort and the characteristic parameters of the object oriented paradigm, and now we are beginning to construct a new system. We would like to obtain an estimation of the effort, and therefore of the cost of the new system.

Number Project	**Effort**	**Number Class**	**WMC**	**NOC**	**DIT**
1	690	34	6	0	3
2	730	35	4	2	2
3	495	22	8	0	2
4	1820	87	5	1	6
5	850	42	2	1	3
6	765	37	6	2	2
7	**¿¿¿???**	58	10	2	3

If we consider a linear model, we can build an equation system with a model like this:
$\text{Eff}_i = \mathbf{A}*\text{NClass}_i + \mathbf{B}*\text{WMC}_i + \mathbf{C}*\text{Noc}_i + \mathbf{D}*\text{DIT}_i + \text{err}_i$
In this case, the equation systems is the next:

690	=	34A	+	6B	+	0C	+	3D	+	Error_1
730	=	35A	+	4B	+	2C	+	2D	+	Error_2
495	=	22A	+	8B	+	0C	+	2D	+	Error_3
1820	=	87A	+	5B	+	1C	+	6D	+	Error_4
850	=	42A	+	2B	+	1C	+	3D	+	Error_5
765	=	37A	+	6B	+	2C	+	2D	+	Error_6

The first step will consist in obtaining a solution (or a nearness to the solution) of this equation system, and then build an estimation model. For example, if the better solution to this system consists the following values: A=20, B=5, C=0.5 and D=0.5, we can appreciate that the values of the errors is the following:

$\lvert\text{Error}_1\rvert=21.5$	$\lvert\text{Error}_2\rvert=8$	$\lvert\text{Error}_3\rvert=14$
$\lvert\text{Error}_4\rvert=51.5$	$\lvert\text{Error}_5\rvert=2$	$\lvert\text{Error}_6\rvert=7$

Then, we would obtain two important things:
a) An estimation model: $\text{Effort}_i = 20*\text{NumberClass}_i + 5*\text{WMC}_i + 0.5*\text{Noc}_i + 0.5*\text{DIT}_i + \text{error}_i$
b) An estimation of the maximum error that are able to be charged with.

With those data we can already realize the estimation of the project that we are constructing. We are substituting the project data in the forecast model: Effort = 20*58 + 5*10 + 0.5*2 + 0.5*3 + error = 1212,5 + error.

Therefore, we can think that the effort will be approximately 1212.5 and with an error margin between –50 and +50.

7 Conclusions

In this paper an overview of several effort estimation methods for object oriented technology has been presented. Nowadays, the principal effort estimation method is FPA, but the trend is to extend this method or create new ones that will make suitable the new characteristic of the new development methodologies.

Further investigation is necessary to create new methods of estimation for the new technologies and to validate recent estimation methods that are not consolidated so far. We must carefully collect data from OO projects in order to improve and calibrate these estimation methods.

8 Bibliography

[1] Bolk, F. (1998) Function Point Analysis and Data Warehousing. *Proceeding of the ORACUG User Group Conference*. Viena, 21-24 April 1998.

[2] Catherwood, B. AND Gupta, M. (1998) Life-Cicle Estimation for Object Oriented Systems. Prediction by Analogy. *Proceeding of the FESMA98 Conference*, 6-8 May 1998.

[3] Crosstalk, (1998) The Journal of Defense Software Engineering. Software Metrics Capability Evaluation, Metodology and Implementation. *Http://stscols.hill.af.mil/crosstalk/1996/jan/metrics.html*. May 1998.

[4] Dolado, J.J. (1997). A study of the Relationships among Albrecht and Mark II Function Points, Lines of Code 4GL and Effort. J. *Systems Software*, 1997, nº 37, p 161-73. New York: Elsevier Science Inc.

[5] Fenton, N.E., Iizuka, Y. and Whitty, R.W. (1995). *Software Quality Assurance and Measurement: A Worldwide Perspective*. London: International Thomson Computer Press.

[6] Hateras Software, inc. (1998) OOMetrics. *Http://www.hatteras.com/metr_des.htm*. May 1998.

[7] Henderson-Sellers, B., (1996). Object-Oriented Metrics: Mesures of Complexity, *Prentice-Hall*.

[8] Jenson, R. And Bartley, J. (1991). Parametric Estimation of Programming Effort: An Object-Oriented Model. *J. Systems Software*. 1991, 15, pp. 107-114.

[9] Kemerer, C. (1993). Reliability of Function Point measurement. *Communications of the ACM* . February 1993, Vol 36, Nº 2, pp. 85-97.

[10] Leader, A. y Prasad, J.(1992). Nine Managemente Guidelines for Better Cost Estimating. *Comunications of the ACM*. February 1992. Vol 35, Nº 2. PPs 51-59.

[11] Minkiewicz, A. (1998) Estimating Size for Object-Oriented Software. *Http://www.pricesystems.com/foresight/arlepops.htm* . June 1998.

[12] Nesi, P. And Querci, T. (1998). Effort estimation and prediction of object-oriented systems. *The Journal of Systems and Software*, 42, pp. 89-102.