ESTIMATION SOFTWARE PROJECT THROUGH BASIC FUNCTIONAL UNITS

Javier Alvarez *, Francisco Ortega**, Vicente Rodríguez**, Nieves Roqueñí**

**** Sociedad Regional de Recaudación. Principality of Asturias***

***** University of Oviedo. Project Engineering Area***

RESUMEN

Son numerosos los estudios, que demuestran el alto grado de fracaso que se produce en los proyectos de sistemas de información, debido al incumplimiento de los plazos y costes estimados inicialmente. En este documento, se tratará de buscar una causa que justifique estos fracasos y proponer una alternativa que solucione el problema o lo minimice.

Se analizarán algunos de los más famosos métodos de estimación sobre proyectos informáticos, examinando la problemática general de los mismos y proponiendo otra forma de realizar la estimación inicial del esfuerzo.

En concreto, se propondrá una variación sobre el método de puntos de función de Albrecht, en la que partiendo de una primera especificación funcional, se pueda realizar una estimación del esfuerzo necesario para desarrollar el proyecto, que será diferente en función de los lenguajes/tecnologías de programación que se vayan a utilizar, de forma que además de realizar una estimación del esfuerzo, se pueda contrastar cual es la tecnología y/o lenguaje de programación más adecuado para ese proyecto concreto y el coste adicional de usar otra tecnología.

SUMMARY

There are many surveys showing the high degree of failure that takes place in the projects of information systems, due to the breach of the terms and costs considered initially. In this work the cause that justifies these failures is identified and an alternative to solve the problem or diminish it is proposed.

Limitations of the most famous methods of estimation on computer science projects will be analyzed, examining the problematic general of them and proposing another form to make the initial estimation of the effort.

In particular, a variation of the method of Albrecht's function points is presented, in which starting off of one first functional specification, estimations of the effort necessary to develop the project can be made differently depending on the programming languages/technologies to be used. With this new approach, besides to make an estimation of the effort, it is possible to contrast the technology and/or programming language best fitted to a specific project and the additional cost of using another technology.

Estimation technique has been tested in software development projects of high cost (more than 600,000 €, with successful results.

1. SIZE AND EFFORT ESTIMATION.

In this section, we are going to analyze several of the most used and known methods of estimation, such as COCOMO, points of function and SLIM.

2.1 COCOMO method

It is very extensive the bibliography that studies the COCOMO method that was designed by Barry Boehm, as for example [IEE 1997], [Gra1994], [Dol 2000], [With 1997], etc.

Boehm carries out a division of the COCOMO model in three different levels of development:

- Basic model: static Model which calculates the effort and cost of the development of the software as function of the size of the program, expressed in lines of the code estimated (LDC).

- Intermediate model: The effort is a function of the size of the program and of a set of “Cost guides “that are a group of attributes of the product, of the hardware, of the personnel and of the project, which are subjectively evaluated.

- Advanced model: It is an extension of the intermediate model, there being evaluated the impact of the “guides of cost “in every phase of the process of creation of software.

In the same way, because of its application, Boehm classifies the projects in three types; organic, semifitted and fitted.

The “guides of cost " consist of 15 attributes valued at a scale that goes from " very down " to " extra high place " depending on the importance and grouped in four categories. Boehm establishes a series of parameters that determine the scale and gives the value for each of the six points that constitute the same one, each of the values are going to be named
"multiplier of the effort ", and of the product of all of them it is obtained the EAF (factor of adjustment of the effort). There shows the formula corresponding to the calculation of the effort in the intermediate model:

$$E = ab\ bd\ (KDSI)\ *EAF$$

2.2 Points of function method

Points of function method was developed by Allan Albrecht and in it, there are defined a series of user's functions which are; external income, external exits, external consultations, external and internal files of interface. Albrecht establishes three levels of complexity; down, middle and high, considering the above mentioned levels with a series of values [IFP 1994]. With this information it can be obtained the points of function not fitted:

$$\text{PFNA} = \sum_{i=1}^{15} (\text{n° of elements}_i)*\text{weight}_i$$

The following step, consists of considering 14 factors of adjustment of the system, establishing a value from 0 to 5 for each factor, according to the degree of scope that the factor for the system possesses. Once the above mentioned values are determined, it is possible to calculate the adjustment of technical complexity (ACT) by means of the following formula:

$$\text{ACT} = 0{,}65 + 0{,}01*\sum_{i=1}^{14} F_i$$

Finally, the definitive points of function are obtained of the product of the PFNA by the ACT.

2.3 SLIM method

The SLIM model was proposed by Putnam [Put 1978] being based on his database of projects. The method rests on the theoretical model of Rayleigh-Norden to obtain a " equation of the software " that connects the expected number of lines of code with the effort and the time of development.

The method is based on a constant of productivity of the process, including a set of factors that affect the organization, such as the project management, a right functional specification, quality of the design codification and tests, the experience of the members of the group, etc.

The low values of the constant are joined together with basic environments, with inadequate tools, or to many complexity in the product, nevertheless, the high values mean good environments, experienced personnel or to products of low complexity and easy to understand [Pre 1992]. The final formula is this

$$L = C*K^{1/3}*t_d^{4/3}$$

2. METHOD VARIANT POINTS OF FUNCTION

As we could see in the main methods of estimation, it exists a series of factors that affect the calculation of it, that is to say, the estimation must be corrected depending on factors that depend on the own characteristics of the project. On the other hand, the usest elementary measure of size is the line of code.

In view of the speed with which they change and develope the programming languages, it is necessary that the basic unit of size used in the estimation is different to the line of code, since every time, they are more the API's, bookshops or classes that the manufacturers provide to the developers with the tools of programming, which they make very difficult to compare the lines of code of one or another language when it comes to estimating the size, therefore, a unit of measure more generic is now necessary that could be parametered depending on the used language of programming.

The above mentioned basic unit, must be able to be defined likewise in the previous study of the estimation, therefore, must be able to be deduced from an analysis of requirements and a functional previous specification.

Finally, the above mentioned basic unit, must be able to turn into the corresponding units of effort for a specific language of programming, for it, it will be necessary to store the information of previous projects, but it must be classified by programming languages of programming and / or development technologies. This implies a greater precision that with the code lines, since it will only be used the historical information of projects with an equal or similar technology.

2.1 Basic functional unit

Taking into account the need to look for a basic unit of functional measure, which can be easily turned into a unit of suitable effort to carry out an estimation, it is proposed the use of the "module", understanding by modules; a series of functional patterns which are repeated in the vast majority of development projects of software regardless of the technology or of programming language which is used.

Below it is proposed a typical classification of modules of a software development; Screens of entry of information, screens of exit of information, interactive processes, batch processes, on-line and off-line reports.

It can happen that due to the use of a specific technology, there are used a series of specific modules, which are not considered in the basic cases or which are specializations of the same ones. For example, in an web application, it is common to find pages or screens in which the results of a search are shown, being able to surf to the following screen of results or specifically go to one through a link. This type of screens might be considered to be a module called "Paginadores", which in this case would be important to distinguish of the
" Screens of exit ", since though it might be considered to be a specialization of the same ones, they would imply a different development and since in a web application

they would often happen, it would be significant for the estimation to distinguish this singularity.

Therefore, the list of modules, might be extended or diminish depending on the technology of development that is used, but the important thing is that the set of the information that they represent is the appropriate, that is to say, it is no use that the modules are very concrete if this information cannot be obtained in a previous study of the project. The same thing happens if too generic modules and the above mentioned information are used, though it could be obtained in the previous study,it is so wide and ambiguous that it is not possible to turn it into a measure of estimation.

Once it is decided the list of significant modules for the application that is going to be carried out, it is necessary to know the number of modules of every type that are going to be carried out. For it, is basic that this data are collected during the phase of interviews with the participants of the project. This study would allow to know in a first approximation, how should be the user's interface of the application, as well as the processes related to the above mentioned interface and a degree of complexity of the same ones.

This type of information should be classified depending on the defined modules for this project and with regard to a degree of complexity that might be of three levels (low, middle and high), so that at the end of the study, for each type of module, there would be have a number of modules of low complexity, another one of middle complexity and another of high complexity.

2.2 Estimation based on functional modules

In order to get the specific effort value in days / men, it is necessary to break down its modules in other specific units of the language or technology used, being necessary a table of conversion of (functional) modules to the respective units of the language for every possible technology.

For example, supposing that after a previous study, for a type of module, the following resultas are obtained:

Module	**High complexity**	**Middle complexity**	**Low complexity**
"Screens of entry of information "	2	5	3

According to the technology that is in use, it can happen that the module breaks down of different forms in components of the used language, for example, if the application is programmed in Visual Basic, with a part client and bookshops of access to information distributed in the servers, it might be:

Module	Language components
"Screens of entry of information "	1 Visual Basic Form 1 Dynamic bookshop (data entry)

But for example, if a web technology is being used, the decomposition might be:

Module	Language components
"Screens of entry of information "	1 ASP 1 JavaScript 1 XML 1 PL-SQL

Finally, the stored historical information, based on the results obtained in previous projects, it would have the entries as a language components instead of as a level of modules.

An example of a modules - efforts parameter table, but with entries as for language components it might be the following one:

COMPONENT	COMPLEXITY		
ANALYSIS	**LOW**	**MIDDLE**	**HIGH**
ASP	2	4	6
XML	1	1.5	2
PL-SQL	2	6	9
...	...	...	...
DEVELOPMENT	**LOW**	**MIDDLE**	**HIGH**
ASP	1	3	6
XML	1	2	4
PL-SQL	3	6	9
...	...	...	...

If what is wanted to obtain is a general estimation of the effort, it would be necessary to add the days of effort in all the phases; analysis, development, tests and documentation for all the language components, so that the previous table would reduce to an entry for each component and three effort values according to the complexity. In case it is wanted to estimate the effort in one of the phases, it would take directly the value of the table.

The necessary effort for the development of each module would be obtained adding the effort of every component of that it consists of, taking into account the degree of complexity, in this way three possible values of effort would be obtained for the accomplishment of each module, depending on its complexity.

If M (i) is the number of modules of each type and complexity, Y(i) the effort for each type of module and complexity and n the number of different types of modules multiplied by three possible complexities, the total effort would be:

$$E = F * \sum_{i=1}^{n} M_i * Y_i$$

A multiplier F factor is included, that would be useful to fit the result of the calculation according to the general characteristics of the project in question. The above mentioned multiplier factor, might be calculated according to the experience in previous projects, or it might resort to some of the most known factors of the methods of estimation, like for example, the " adjustment of technical complexity " (ACT) of the points method of function or the " factor of adjustment of the effort " of the method COCOMO, which are operandos that "correct" the estimation according to a series of factors and which adapt the general estimation to the own characteristics of the project. In this case, the use of the ACT is considered to be appropriate.

Once the effort has been calculated, it might repeat the calculation with other programming technologies, using the tables of conversion of modules - components of language, checking thus the different estimations of the effort depending on the technology that will be used.

BIBLIOGRAPHY

[Con 1997] McConnell Steve (1996): *Rapid development*. Trad Esp.: *Desarrollo y gestión de proyectos informáticos.* Ed Microsoft Press (1997).

[Dol 2000] Dolado Cosín, José Javier y Fernández Sanz, Luis (2000): *Medición para la gestión en la Ingeniería del Software*. Ed Ra-Ma (2000).pp 209-223.

[Gra 1994] Granja Álvarez, Juan Carlos y Barranco García, Manuel J. (1994): *Proyectos informáticos*. Ed. Universidad Nacional de Educación a Distancia (Centro asociado “Andrés de Vandelvira” Jaen).

[IEE 1997] Richard H. Thayer (1997): *Software Engineering Project Management*. Ed. Los Alamitos, CA: IEEE Computer Society Press.

[IFP 1994] IFPUG Internacional Function Point User Gloup: *Function Point counting rules 4.0* (1994).

[Pre 1992] Pressman, Roger S. (1992): *Software Engineering. A Practitioner's Approach. Trad. Esp.: Ingeniería del Software. Un enfoque práctico (3ª edicion)*. Ed. Mac Graw Hill (1993).

[Put 1998] Putnam, L.H.: *"A general empirical solution to the macro software sizing and estimating problem", IEEE transactions on software engineering, vol. 4* (1978) pp 345-361.

CORRESPONDENCE

José Javier Álvarez Larraceleta
Sociedad Regional de Recaudación. Principality of Asturias
Telf. 985104272
E-mail: larra@navegalia.com