А.А.Горовик, М.В.Лазарева
Ферганский филиал Ташкентского университета информационных технологий, г.Фергана
УДК 658.52

ОЦЕНКА СЛОЖНОСТИ И ВРЕМЕНИ РАЗРАБОТКИ ПРОГРАММ ПРИ ОПРЕДЕЛЕННЫХ ТРЕБОВАНИЯХ КАЧЕСТВА ОСНОВАННАЯ НА ГЕНЕТИЧЕСКОМ ПОДХОДЕ

Аннотация. Основной проблемой современных методов оценки трудозатрат является сложность их адаптации к каждому конкретному проекту. В статье предложен принципиально новый, неклассический генетический подход к этой проблеме, главные принципы которого — обеспечение качественной и количественный оценки трудозатратности разработки программных проектов.

Ключевые слова: оценка трудоемкости, модель COCOMO, техника PERT, точки свойств, объектные точки, оценка по аналогии, принцип Паркинсона, генетический алгоритм оценки, оценка объема программного кода.

ESTIMATION OF THE DIFFICULTY AND TIME OF PROGRAM DEVELOPMENT FOR SPECIFIC QUALITY REQUIREMENTS BASED ON THE GENETIC APPROACH

Abstract. The main problem of modern methods of evaluation of labor is the difficulty of adapting them to each specific project. This paper proposes is a fundamentally new, non-classical genetic approach to the problem, the main principles of which is providing qualitative and quantitative assessment of labor-intensive software development projects.

Keywords: assessment of labor input, COCOMO model, the PERT technique, the object point, the assessment by analogy, the principle of Parkinson's, the genetic algorithm evaluation, assessment code volume.

The exact calculation of the resources necessary to implement this software product with the specified quality requirements is one of the main problems in the field of project management. However, with such a calculation, it becomes difficult to take into account a huge number of factors that affect the software life cycle. Find

out the hidden patterns and relationships between these parameters, as well as make an analytical calculation in the vast majority of cases is not possible. In any software project, you have to balance between the cost, time, quality and volume of the functionality being realized. As a result, to date, many companies face serious problems in the event of incorrect calculations of the required timing:

- while underestimating - unforeseen expenditure of additional funds, dissatisfied in customer defaults on time, "sleepless nights" of employees, the complexity of the "avalanche" of management about the poor quality of the final product, underdeveloped system functions, and so on.;
- the revaluation - a useless race course resources involved in the project, from the Kazakh customer from the contract with the data by the conditions (due to possible loss of jobs is), etc.

In today's market of large software systems, losses can amount to millions of dollars.

Accurate estimates of the costs of production of software are important to both the developers and the customer (the client). They can be used in contract negotiations, planning, monitoring, etc. Thus, there is a real need to develop methods and tools that allow the manager to assess the required time and human resources based on all available project characteristics: the history of previous similar projects, experience and Employee productivity, company specifics, etc. In addition, it is necessary to recalculate and refine the time and resources already at the development stage of the system, taking into account the current trends observed during the project implementation. This will help the manager to timely detect deviations from the established schedule and take appropriate measures in project management.

A brief overview of the main concepts and problems of the area under consideration

The immediate efforts of producers (their work) provide the bulk of the cost of software development, and as a rule, methods for estimating the necessary funds (as a consequence, and time) focus on this aspect and give estimates in person-months that can subsequently be converted into duration Project or cost.

In practice, there are often three problems that are of fundamental importance:

1. Which model Assessments choose?

2. Some metrics on the Use Call size (number of lines of code (LOC - Lines Of Code), «functional point» (FP - Function Points) or "Points properties » (feature points))?

3. What can be considered a good assessment?

Choosing an evaluation model

A widely used method of estimating labor costs is an expert evaluation. However, this approach is fraught with many problems:

- The grounds for obtaining an assessment are not explicit;
- difficult to find of highly qualified experts for each new about the project;
- connection between the size of the system and the labor dozatratami nonlinear. Efforts by WHO will melt exponentially with the increase in volume. Therefore, expert evaluation is obtained adequate only if the current and previous projects of approximately the same size;
- management policy aimed at reducing costs, as a rule, casts doubt about the real experience of the previous projects and makes the share of "blind optimism".

Over the past three decades, many different models of quantifying labor costs have been developed.

They range from models based on empirical data (for example, the model of "COCOMO" by Bohm [1]) to purely analytic ones. Empirical models use the data from previous projects to evaluate the current one (by analyzing patterns observed in previous projects). On the other hand, analytical models are based on global assumptions about the relationship of various parameters, such as the speed with which the developer fixes defects and their number at a certain point in time. Each model has its advantages and disadvantages, but the key factor in its consideration is, of course, accuracy.

Selecting the size metric

Most models (both empirical and analytical) are based on the use of various metrics of the size of the system being developed (LOC, FP, etc.). The accuracy of the evaluation of labor costs directly depends on the accuracy of the size estimate.

Regardless of the size metric chosen, it is not possible to accurately determine it in advance, so it is necessary to refine it (and accordingly perform a general reassessment of labor costs) already directly in the development process. The effect of this factor is noticeably reduced by using a sufficiently rigorous development methodology and detailed elaboration of the system requirements. Due to the special importance of this issue for solving the initial problem, it will be considered in more detail in the section "Comparison of Software Size Metrics".

Criteria for good evaluation

According to Royce [2], a good estimate of software production costs should be:

- understood and supported by the Smart rum project and development team;
- Approved by all stakeholders as feasible;
- based on a clear model with credible bases, as well as data on such a project (with similar business processes, technologies, environment, people and the requirements);
- as defined in detail what the key risk areas are clear, and The probability of success objectively evaluated.

Comparing Software Size Metrics

The size of software is the most important factor determining the complexity of software implementation. Next, the author will describe five software size metrics used in practice. The number of lines of source code and function points are the most popular metrics from the five considered.

Number of lines of code

LOC (Lines Of Code) - the number of non-empty lines of source text, excluding comments [4]. Despite the fact that this metric essentially depends on the chosen programming language, it still remains the most used software size metric. The exact number of LOCs can only be obtained after the project has been completed. Therefore, estimating the size of the program code prior to its creation is not much simpler than estimating real labor, for example, in man-months.

A typical method of implementation of this assessment uses a combination of expert assessments technique called PERT [6], is as follows: suppose there are n experts, each i-th expert expresses three assumptions about the final size: L i - lower

bound on the size; H i - - the size of the upper bound; M i - the most likely size. Then the size of S can be computed as the accuracy of the assessment can be significantly improved by applying PERT not to the project as a whole, but to its individual components. In this case, a general estimate of the size can be obtained as the sum of "local" estimates.

$$\boldsymbol{S = \frac{1}{n}\sum_{i=1}^{0}\frac{L_i+H_i+4M_i}{6}}, \quad (1)$$

The Halstead Metrics

M. Halsted proposed such metrics as code length and volume [7]. The length of the code is defined as

$$N = N\,1 + N\,2, \quad (2)$$

where N 1 - the total number of occurrences of the operators in the program; 2 N - total number of operands. The amount of code corresponds to the amount of memory required to store the program, and is calculated by the formula:

$$V = N \log (n\,1 + n\,2), \quad (3)$$

where n1 - number of different operators; n2 - number of different operand appearing in a program.

Obviously, it is usually more difficult to estimate the total number of operators and their operands before the end of the project than to evaluate the LOC, so a lot of comments were made on the metrics proposed by Halstead. Support for this approach has been steadily declining in recent years.

Function points

The most successful replacement of the number of lines of code to become functional size measurement point (function points), first proposed by IBM employee A. Albrecht in 1979 [8]. The application of functional points is based on the evaluation of the volume of the functionality being realized by studying the requirements, so that the evaluation of the required labor can be performed at the earliest stages of the project and will be further refined along the life cycle, and the explicit relationship between the requirements for the system being created and the resulting estimate allows The customer to understand what he is paying for, and what the change in the initial task will result in.

The author will briefly review the basic principles of this method. The total number of functional points of the program depends on the number of elementary processes of five types (Figure 1):

1. incoming transaction (External inputs (El)) - Receive data from user;
2. outgoing transaction (External outputs (EO)) - Transmit data user;
3. user interaction (External inquiries (EQ)) - Interactive Dialogues from user (requiring from Him any - any actions);
4. files internal logic (Internal logical files (ILF)) - files (logical Groups information), used in Within these systems interact;
5. files external interactions (External Interface files (EIF)) - Participate in External Interactions from Other SIS topics.

In this terminology, a transaction is an elementary, indivisible, closed process that is important to the user and transfers the product from one consistent state to another.

Each of the five types is assigned one of three levels of complexity (1 = simple, 2 = medium, 3 = complex), and each pair (type, difficulty level) is assigned a weight that is the number of unaligned function points (UFP) From 3 (for a simple incoming transaction) to 15 (for complex internal files). The total size estimate in UFP is calculated as (4):

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} N_{ij} W_{ij},$$

where N i W ij Respectively, the number and weight of elements of a system of class / with complexity /.

For example, if the system easy entry 2 (W ij = 3) 2 complex output (W ij = 7) and a complicated interior file 1 (W ij = 15), whereas UFP = 2x3 + 2x7 + 1x15 = 35.

This is the number of function points can be directly used to estimate the cost / labor intensity and refined with the help of alignment factor (VAF), which is calculated on the basis of the characteristics

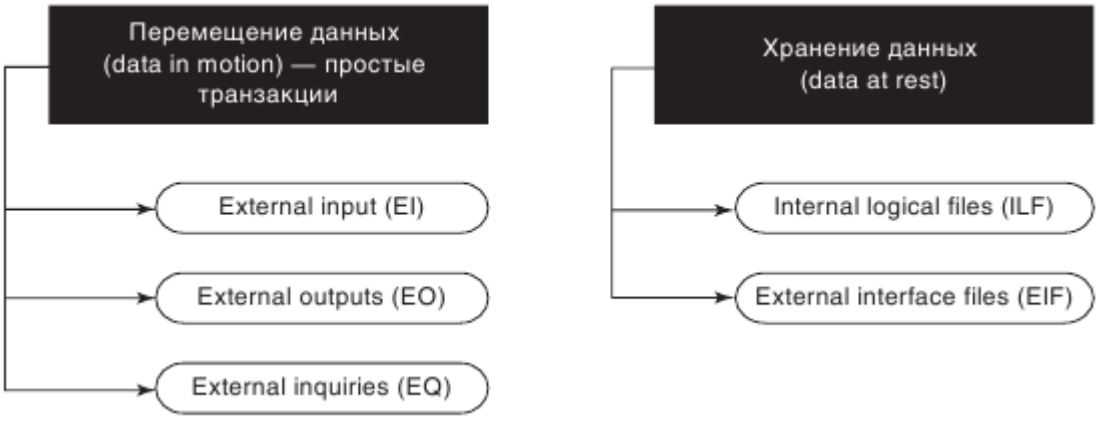

Figure 1. Types of elementary processes used in the FP method

Total project, such as: degree of distribution of processing and storage of data, system performance requirements, security requirements, etc. In this case, the final estimate of the size in the aligned function points is calculated as

$$AFP = (UFP + CFP) \bullet VAF, \quad (5)$$

Where CFP - additional functional points, which will be required, for example, to install or migrate data. The general scheme of the evaluation procedure is shown in Fig. 2.

The number of UFP and the number of lines of code (LOC) are connected linearly:

$$LOC = \text{and} \times UFP + b. \quad (6)$$

The parameters a and b can be obtained by linear regression on the basis of available data on the project.

1. Basic model of COCOMO. Project Size S Measured at LOC (KLOC), and spent labor - in person-months. Created on the basis of statistical data analysis 63 projects (mostly US Ministry of Defense CTBA) of various types. Isa are three sets of parameters {a, to} in Dependencies from difficulties to develop relevant software:

a. for simple, easily understood about the projects and = 2,4, b = 1.05;

b. for complex systems and = 3,0, b = 1.15;

c. for embedded systems, and = 3,6, b = 1, 20.

The model was easy to use, but did not provide the required accuracy.

2. Detailed model COCOMO. Refined set of parameters {a, to}, except of total formula Accepted the form: E = M • a • S b, where M - Specifying the

coefficient is calculated as the product of 15 correction factors of 4 the categories of (final product factors, the computational environment, personnel, project), in the range of 0.7 to 1.66, which can be found in the special table . These changes are a basic fashion if allowed to significantly improve the point of evaluation, especially in the case applies the method of the individual components, and not to the system as a whole.

Results

The ideas described allowed the development of a prototype of a system for the refinement of labor estimates based on the COCOMO II model. According to the databases of the bugtrackers of three open source projects (Firefox, Fedora, KDE), estimates were made that averaged 4% more accurately than the estimates obtained with the help of COCOMO II, without clarification through the method considered (Table 1). In the future it is planned to supplement the developed system with simplified versions of existing methods, which will make it possible to use it as a full-fledged independent software package. Increased accuracy assessment for some opensource-projects.

Project	The simulated date of the prediction	Actual release date	COCOMO II	COCOMO II + Genetic	Increase in accuracy,%
Firefox 49.0.2	15.08.2016	20.10.2016	24.10.2016	26.10.2016	2,7
Fedora 25	10.08.2016	11.10.2016	03.10.2016	14.10.2016	8,1
KDE 5.7.2	04.06.2016	19.07.2016	28.07.2016	29.07.2016	1,3

Prospects for the development of proposed ideas

To date, there is a sufficient number of advanced tools for analyzing project repositories (for example, SolidSTA), however, they do not make any predictions about the actual deadlines, usually only adapted for use by managers or analysts and, in fact, provide only the collection of statistics and visualization, Not doing a detailed analysis, since it also makes it difficult to take into account a huge number of factors that fluctuate from project to project.

The goal of further research and development is to create an extension for the IDE, which must meet the following requirements:

- Provide the ability to control the performance of each individual developer, and the entire team as a whole;
- based on the analysis of design re poser signal the deviation s from the planned schedule and Regis IU one - click produce Recalculation Terms Putting project (and / or A separate module / assembly) in operation at the moment;
- possess a sufficient degree of universality (should not rely on any particular model of the gap processing, programming language, etc...);
- It shall be calculated not only on the project management, but also on the ordinary developers, as well as all those who The interested in getting the idea of those pace of development for the planned schedule (for self-control, various forms of analytics and others.);
- Provide an opportunity to visualize the results of the analysis.

REFERENCIES

1. *Albrecht J., Gaffney JE*. Software function, source lines of codes, and development effort prediction: a software science validation, IEEE Trans Software Eng. SE-9, 1983. P. 639-648.
2. *Aron J.* Estimating Resource for Large Programming Systems Rome, NATO Science Committee, 1969.
3. *Boehm B.* Software engineering economics. New Jersey, Prentice-Hall, 1981.
4. *Fenton N.* Software Metrics: A Rigorous and Practical Approach, London, Chapman and Hall, 1991.
5. *Halstead M.* Elements of software science, NY, Elsevier, 1977.
6. *Jones C.* Applied Software Measurement, Assuring Productivity and Quality, NY, McGraw-Hill, 1977.
7. *Parkinson G.* Parkinson 's Law and Other Studies in Administration NY, Houghton-Miffin, 1962.
8. *Royce W.* Software project management: a unified framework. Reading, Addison-Wesley Professional, 1998.
9. *The Boehm.* Software Cost Estimation with Cocomo II. NewJersey, Prentice-Hall, 1981.