Practitioner's Approach to Software Cost Estimation

Baiba Apine

Riga Institute of Information Technology

Skanstes iela 13, LV-1013 Riga, Latvia

E-mail: baiba.apine@dati.lv

One of the most difficult and important software development activities is effective software estimation. Nevertheless it is one of the most important. Number of formal software project estimation methods have been developed. Several methods based on function points are discussed and problems dealing with practical usage of these methods are highlighted. Our experience in Basic COCOMO, COCOMO II and ObjectPoint methods is presented.

1. Introduction

Effective software estimation is one of the most difficult software development activities. Nevertheless it is one of the most important. Underestimating a project will lead to under staffing it, under scoping the quality assurance effort, and setting too short a schedule. That can lead to staff burnout, low quality, loss of credibility as deadlines are missed, and ultimately to an inefficient development effort that takes longer than nominal [1].

Overestimating a project can be almost bad, the project will take as long as estimated.

Usually estimates are made using past experience only. This solution is good in cases when new project is of the same size and requires the same amount of effort. Number of formal software projects estimation methods have been developed. All of

them have their own strength and weaknesses.

We are going to use these methods to estimate the workamount of the software projects we have to develop and do it as early as possible. Software total cost means the cost of the whole software development process: specification, coding, testing, documenting, configuration management etc. This process could be estimated from various aspects: productivity - output of the software engineering process, quality - how software developed satisfies the needs of customer and functionality - how the software is built [3]. Let us concentrate on functional aspect of the software development process: how to estimate the cost of the software functions. The first who proposed this method was 2cht [2]. Now there are various modifications of this method,

2. Method

From the user's point of view system functions would be grouped in five groups [3]: user inputs, user outputs, user inquiries (on-line input that results in the generation of some immediate software response), logical files (logical grouping of data, and there is no difference whether it is a database or temporary data for internal use only), external interfaces (machine readable interface used for data transmission to another application). Each item of the each group should be estimated as simple, average or complex. Multiple each counted item with appropriate weighting factor given in Table 1, and count total. The total you get is called unadjusted function points (FP). Determination of complexity is considered to be subjective, but there are resources giving more formal criteria for determination [4].

Measurement parameter	Weighting factor		
	Simple	Average	Complex
Number of user inputs	3	4	6
Number of user outputs	4	5	7
Number of user inquires	3	4	6
Number of files	7	10	15
Number of external interfaces	5	7	10

Table 1. Function oriented estimation weighting coefficients

Most of software cost estimation methods based on COCOMO (COCOMO II) use unadjusted function points as input data.

Function points are independent from the software development environment and this makes the method especially convenient in the large companies with a diversity of software platforms being used in projects. To estimate the number of lines of code, convertion coefficients are used. These coefficients differs for each software development environment and gives average number of lines of code per function point [3], [4]. Figures of estimated lines of code are very important. In companies having large software development experience these figures may give very precise estimation of the person months and calendar months necessary for software development.

A lot of different factors should be kept in mind, for instance, previous experience in solving similar problems, scheduling, communications, etc. A cost driver refers to a particular characteristic of the software development that has the effect of increasing or decreasing the amount of development effort, e.g. required product reliability, execution time constraints, project team experience.

Method could be adjusted for software maintenance, reengineering etc. In most cases we use one of the COCOMO estimation models - the Early Design model. This model is used in the very early stages of a software project when little may be known about the size of the product to be developed, the nature of the target platform, or detailed specifics of the process to be used [4].

Another useful model is the Application composition model (ObjectPoint method). This model addresses applications composed from interoperable components [4].

Our experience is that the most difficult and the most dangerous activity of the estimation process is calculation of function points.

1. Estimation Accuracy

In the earliest stages of software development life cycle, when the request for proposal is received from a customer, very little may be known about software development

environment, staff involved, etc. Some functionality may be not specified or imprecise. Chart given in Figure 1 [4] indicates the accuracy of software estimation. In cases when request for proposal is the only available document with functionality of the system described very approximately, project could be overestimated or underestimated up to four times, that could lead to significant project management problems. It is very important to involve experienced system analysts in software estimation process to achieve more precise results.

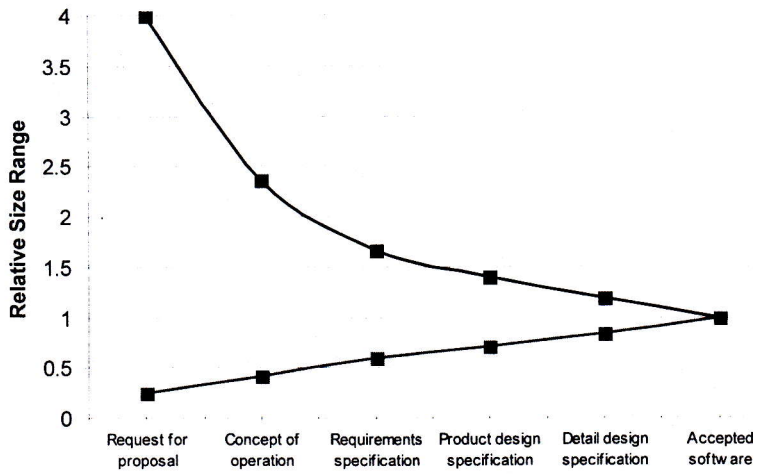

Figure 1. Estimation accuracy of software projects.

2. Common Problems and Solutions

We use estimation methods in the earliest stages of software development process to get preliminary results when request for proposal is received. Hence there are not enough useful information which could be used for counting of function points or estimating the software development platform and staff being involved in potential project. Some information may be omitted in these documents. Additional information could be gathered during interviews with potential customers, but in most cases request for proposal is the only document available. Results of estimation must be included in

proposal as accurate figures, but very often there is a lack of information to get them.

Not all customers provides clearly formulated requests. Some of them are brilliant experts in their own professional area, but have not enough experience in software systems. In this case interviews highlight basic software system demands. Results of interviews are the input material for both preliminary software estimation and the software system's proposal.

In cases when at least software requirements specification is available, estimation results are very close to real development effort, estimation error is approximately 15% - 25% (see Figure 1). From the developers viewpoint, it is better to overestimate project (50% or more) than underestimate it.

Situation, when customer is another software company that outsources a project, differs from described above. Basically they already have done some estimation with their own methods and provide some initial figures. In most cases these figures are believable, but tend to underestimate the software development effort.

It is hard in the large variety of methods to choose the best one. For several years we are using COCOMO and COCOMO II methods for software estimation. The main reason we choose COCOMO was that it supports different quality levels of input data different quality. More precise input gives more precise output and it would be kept in mind.

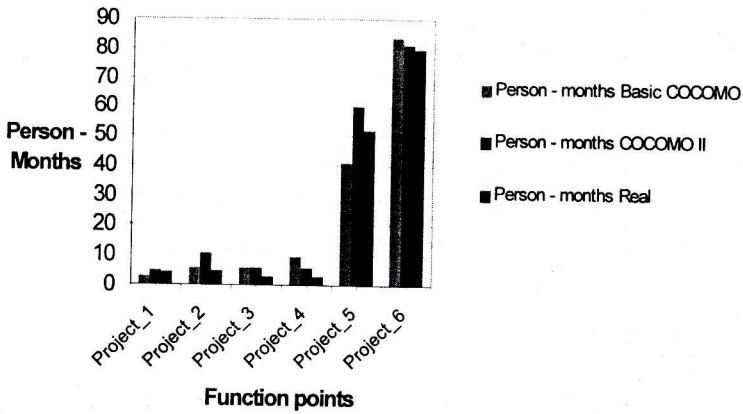

Figure 2. Comparison of estimation results

Basic COCOMO was the first method we started with. This method was quite good. Nevertheless some important aspects of software development process were ignored. Basic COCOMO method uses 14 very important cost drivers [3] besides the classification of the software project's type - organic, semidetached or embedded. They cover software system itself, but do not touch the software development process. Cost drivers covering software development process are added in COCOMO II. For instance, whether the software development environment supports software development life cycle, whether developers' teamwork is supported, staff capability and interactions, etc. While COCOMO II method's Early Design Model gives more precise estimation results, we still use both COCOMO and COCOMO II to achieve more believable results (see Figure 2).

Chart given in Figure 2 shows software projects developed by using of Microsoft software development environments (MS Access, MS Visual Basic and MS Visual C++). All these projects were estimated using Basic COCOMO and COCOMO II Early design model with software requirements specification used as input data for the estimation. The real workamount for the project development is compared with the result of estimation. Basically both methods overestimate projects. There are two projects underestimated by Basic COCOMO method (see Figure 2). The reason is that the software design specification was not precise and approximately 60% of code had to be thrown away. Percentage of breakage must be counted to adjust the effective size of the product.

Chart given in Figure 2 shows differences between estimation results given by Basic COCOMO and COCOMO II. The reason is appliance of cost drivers not supported by Basic COCOMO, but affecting estimation results of COCOMO II significantly. These cost drivers deal with personnel capability and experience. Values of the cost drivers are adjusted experimentally, so in different companies they may differ. Even in one particular company it is necessary to adjust values for each software development group.

Both methods Basic COCOMO and COCOMO II provide development schedule estimates. Chart in Figure 3 shows development schedule estimation results for the software projects, which development effort in person-months is given in Figure 2. Adjusting calendar months for software development process is very important activity.

In most cases Basic COCOMO and COCOMO II overestimates calendar months necessary for software development. Nevertheless there are two projects underestimated in schedule by Basic COCOMO (see Figure 3 Project_5 and Project_6). Underestimation is relatively small: average 14% (see Figure 1).

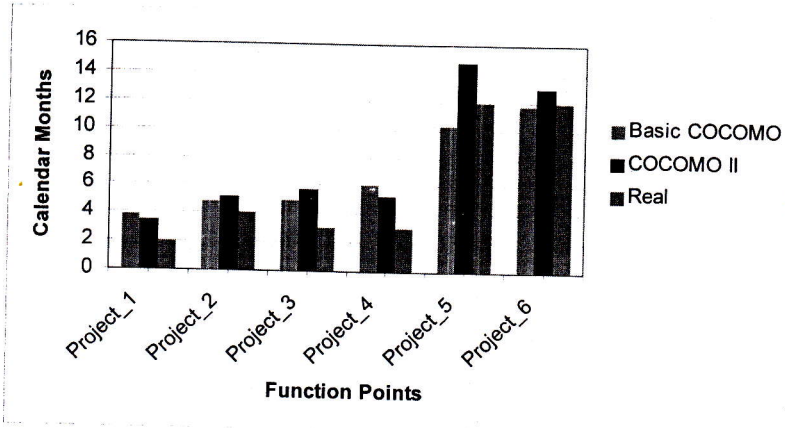

Figure 3. Time schedule estimation with Basic COCOMO and COCOMO II

Besides Basic COCOMO and COCOMO II we have tried to use ObjectPoint method for relatively small and simple applications, which could be built using standard data and user interface modules and without complex algorithms. Currently ObjectPoint method was rejected, because we have only few projects with software being developed by "ready to use" interoperable components only. This method tend to underestimate projects with additional programming activities (see Figure 4). Projects estimated with ObjectPoints method were developed using data access objects and standard visual objects connected to them (for instance, MS Access table and database grid control connected to this particular database table) and some additional programming was necessary for data selection.

Software development environments are changing and coefficients used for transformation of the function points to lines of code must be adjusted for new environments. After examining 10 different projects developed using Microsoft software development tools in our company, we found the following coefficients for converting function points to lines of code (Table 2).

Sometimes software system is initially divided into subsystems or could be divided during the development process. The software system could be estimated as the whole one or by its subsystems. Figure 5 shows the difference between estimation results for whole system and for decomposed one both estimated by using COCOMO II. There are three reasons for higher values for whole project than for decomposed one:

1) product complexity increases,
2) additional management is necessary for larger project,
3) staff communications are more time consuming.

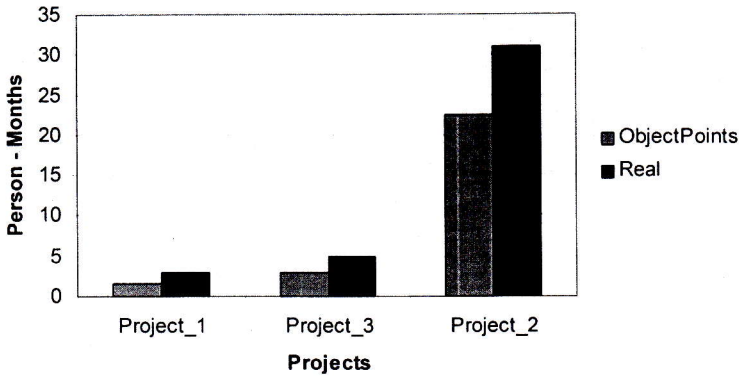

Figure 4. Software project estimation with ObjectPoint method

Software development environment	Source lines of code per unadjusted function point SLOC / UFP
Visual Basic 4.0, 5.0	25
Visual C++	27

Table 2. Coefficients used for converting function points to lines of source code

The conclusion from tendency given in Figure 5 should be: decompose complex system into subsystems and the price of the whole project will be lower. Besides it, chart given in Figure 6 shows difference in estimated calendar months between the whole

system and decomposed one. This difference grows faster than difference in cost (see Figure 5). Divided systems need more development time. If time limit allows, some decomposition could be made to lower total cost of the whole project.

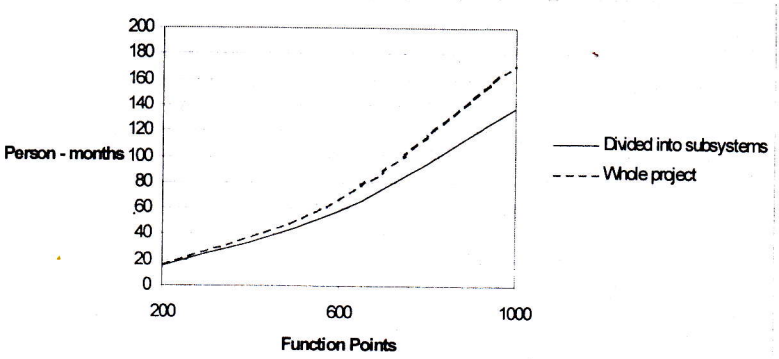

Figure 5. Estimation results for whole and decomposed system

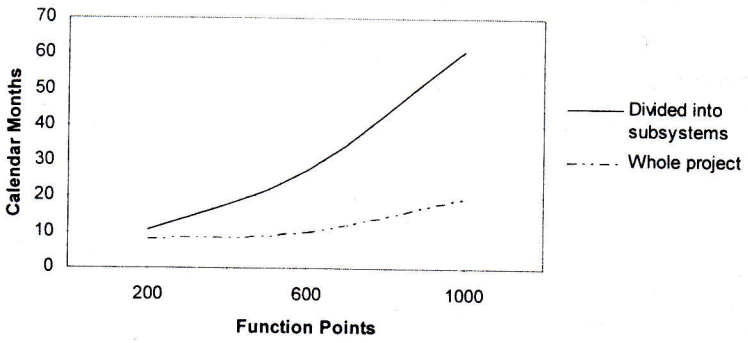

Figure 6. Estimation of calendar months for decomposed and whole systems

1. Conclusions

The main problem in the estimation process is lack of input information. Especially if formal estimation methods are used for software projects "could be". Results of

estimation must be included in proposal as accurate figures, but very often there is a lack of information for getting them.

The most complex and responsible part in the software estimation process is to get precise count of function points. The main reason of estimation faults is incorrect counting of function points. Therefore experienced system analysts could be involved in estimation process.

Although estimation process is based on formal models, it is the kind of art. It is very hard to make customer accept estimation results, which may be overestimated up to 4 times (see Figure 1). From the opposite side, upper level managers and administration tend to think that evaluation results are ideally precise.

2. References

[1] http//www.ifpug.org/ifpug

[2] A.J.2cht. Measuring Application Development Productivity._Proc. IBM Applic. Dev. Symposium, Monterey, California. 1979, pp. 83-92.

[3] R.Pressman. Software Engineering: A Practitioner's Approach._McGraw-Hill, 1992, pp 41-91.

[4] C.Abts, B.4, B.Clark, S.Devnani-Chulani. COCOMO II Model Definition Manual._University of Southern California, 68 p.