# Software Cost Estimation - a Practical Approach

Violeta T. Bozhikova[1]

*Abstract* – **Software cost estimation is considered as one of the most challenging tasks in software project management. The process of software estimation includes estimating the size of the future software product, estimating the effort required, estimating the duration of the project and finally – the people required. This paper gives an overview of the most powerful cost estimation models, discusses their advantages and weakness and finally a hybrid cost estimation approach that combines their strengths is recommended**

*Keywords* – **Software Cost Estimation, Software Cost Estimation Methods, Software Cost Estimation Tools.**

## I. Introduction

Software cost estimation [1-5] is a continuing activity which starts at stage of the project proposal and continues through the overall life time of the software project. The goal of this continual cost estimation is to ensure that the expenses will not exceed the budget provided.

Considerable research has focused on development and evaluation of universal software cost estimation models and tools suitable for all software projects. After 20 years research, we could claim that there are many software cost estimation methods available, but no one method is suitable for all software projects. In fact, their strengths and weaknesses are often complimentary to each other. To understand their strengths and weaknesses is very important for the software estimators. The estimators are increasingly convinced that accurate software estimation is impossible using a single method and increasingly believe that a combination of methods will allow a more accurate and reliable software cost estimate.

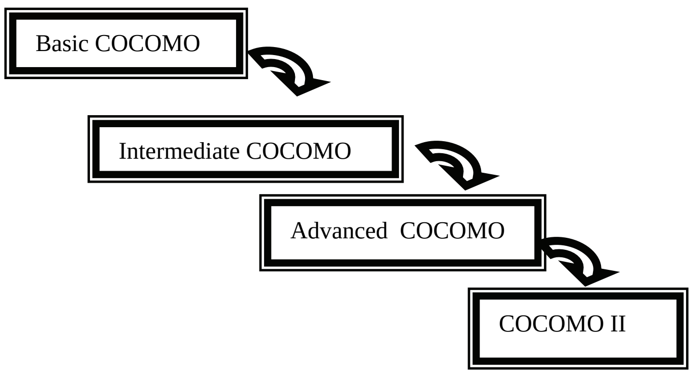
This paper gives an overview of COCOMO hierarchy and Function Points cost estimation models, discusses their advantages and disadvantages and finally a practical cost estimation approach that combines their strengths is recommended as a way for efficient cost estimation.

## II. COCOMO Models and Function Point Analysis

### A. COCOMO Family

One of the most commonly used software cost estimation methods are the **CO**nstructive **CO**st **MO**dels (COCOMO models). These methods for software estimation are considered as algorithmic because provide mathematical equations to perform software estimation. The COCOMO mathematical equations are based on extensive historical research and use inputs such as Source Lines of Code (SLOC), number of functions to perform, and other cost drivers such as language cost drivers, design methodology, skill-levels, risk assessments, etc. As algorithmic methods the COCOMO models have a lot of advantages. The most important are the objectivity, stability and the sensitivity of the results produced. Using such models the estimator gets repeatable results. In the same time, it is easy to modify input data, refine and customize formulas. The general disadvantage of these models is the strongly dependence of the estimations on the inputs. Some inputs can not be easily quantified. As a result, poor sizing inputs or|and inaccurate cost driver rating will result in inaccurate estimation

**Basic COCOMO** [1] is the first from the family COCOMO models. It is designed by Barry W. Boehm as a model for estimating effort, cost, and schedule for software projects in 1981. Now, a hierarchy of COCOMO models is available:



Basic COCOMO model computes software effort applied "PM" (development effort i.e. development cost) in "person-months" as a function of program size expressed in estimated thousands lines of code KLOC. Person month is the amount of time one person spends working on the software development project for one month. This number is exclusive of holidays and vacations but accounts the weekends. The basic Cocomo equations are:

$$PM = a_b\left(KLOC\right)^{b_b}\left[person - months\right]$$
$$TDEF = c_b\left(PM\right)SchedExp \quad \left[months\right]$$
$$Average \quad Staffing = PM / TDEF \quad \left[people\right]$$

The coefficients $a_b$, $b_b$, $c_b$ and SchedExp depend of the type of the project (organic, semi-detached or embedded) and are given in the next table:

| Software Project | $a_b$ | $b_b$ | $c_b$ | SchedExp |
|---|---|---|---|---|
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-Detached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

[1]Violeta T. Bozhikova is with the Faculty of Computing and Automation, Technical University of Varna, 9000 Varna, Bulgaria, E-mail: vbojikova2000@yahoo.com

TDEF is the Development Time in chronological months and Average Staffing is the People required for the whole project development This модел is good for quick, early and rough estimates of software costs, but its accuracy is limited because it doesn't account the influence of a number of well known factors such as hardware constraints, personnel quality and experience and so on that have a significant influence on software costs.

**Intermediate COCOMO** is an extension of the Basic COCOMO. This model computes software developement effort PM as a finction of program size and set of "cost drivers" that include subjective assessements of 15 cost driver attributes that are grouped into 4 major categories "Product attributes", "Hardware attributes", "Personnel attributes", "Project attributes". Each of the 15 attributes is rated on a 6-point scale that ranges from "very low" to "extra high" (in importance or value). Effort adjustment factor (EAF) for a given project is calculated as the product of the fifteen effort ratings ($EM_i$, i=1…15). Typical values for EAF range from 0.9 to 1.4. The Intermediate Cocomo formula for PM now takes the form:

$$PM = EF\,(KLOC)^{ee}\,EAF\,[person - months]$$

Where:

$$EAF = \prod_{i=1}^{15} EM_i$$

Where the coefficient EF and the exponent ee are given in the following table:

| Software project | EF | ee |
|---|---|---|
| Organic | 3.2 | 1.05 |
| Semi-detached | 3.0 | 1.12 |
| Embedded | 2.8 | 1.20 |

The Development time (TDEF**)** and People required (Average Staffing) are calculated from PM in the same way as with Basic COCOMO.

**Advanced COCOMO** can be seen is an extension of the Intermediate COCOMO version. It calculates PM the same way as Intermediate COCOMO but with an assessment of the cost driver's impact on each stage (analysis, design, etc.) of the software engineering process.

The development of the new **COCOMO II** model by the Boehm's team is based on a study of about sixty projects at TRW (a Californian automotive and IT company) in 2002 and is the latest major extension to the original COCOMO. This model is turned to the newer software paradigms (for example OOP) and the modern software life cycles. For comparison, the previous COCOMO models have been very successful for projects up to 100000 lines of code, based mostly on the waterfall model of software development and for programming languages ranging from assembly to PL/I. In addition, the previous COCOMO versions were defined in terms of estimated lines of code LOC (and thousands of LOC, i.e. KLOC). The COCOMO II model bases the calculation of required effort PM on the software project's size measured in SLOC (and thousands of SLOC, i.e. KSLOC). The difference between LOC and SLOC (single Source Line of Code) is that a SLOC may include several physical lines. Each structured construction, for example the "if-then-else" statement would be counted as one SLOC. For comparison, in basic COCOMO model this statement might be counted as several LOC.

The first equation below ($PM_{nom}$) is the base model for the Early Design and Post-Architecture cost estimation of the software project. The inputs are the Size of software development in KSLOC, a constant A and a scale factor – B [3]. The size is in KLOCS is derived from estimating the size of software modules that will constitute the application program. It can also be estimated from unadjusted function points (UFP), converted to SLOC then divided by one thousand. The scale (or exponential) factor B derived from five scale drivers, such as Team Cohesiveness factor, Process maturity factor, Precedentness, Flexibility and Breakage factor and accounts for the relative economies or diseconomies of scale encountered for software projects of different sizes [3]. The constant A depends on the size of the project. The nominal effort $PM_{nom}$ and the adjusted effort $PM_{adjasted}$ calculations for a given size project and expressed as person months are presented by the next equations:

$$PM_{nom} = A(KLOC)^{B}\,[person - months]$$

$$PM_{adjasted} = PM_{nom}\,(EAF)\,[person - months]$$

Where:

$$EAF = \prod_{i=1}^{17} EM_i$$

COCOMO II has 17 cost drivers attributes (Analyst Capability, Applications Experience, Programmer Capability, Use of Software Tools, Multisite Development, Required Development Schedule, Required Software Reliability, Database size, Product complexity, Personnel Experience, Language and Tool Experience, Personnel Continuity, Execution Time Constraint, Main Storage Constraint, Platform Volatility, Required Reusability, Documentation match to life-cycle needs) which rating (expressed as a number $EM_i$, i=1…17) the estimator has to determine with the goal to calculate the value of effort required EAF .

## *B.Function Point Analysis*

Although counting lines of code is the first and most common software sizing methodology this sizing method is no longer practical due to the great advancements in software engineering and modern programming languages. Another commonly used sizing method is the IFPUG method [5] called Function Point Analysis (FPA). It is another method of quantifying the size in terms of the functions that the system delivers to the user. The function point measurement method was developed by A. Albrecht at IBM in 1979. The main advantages of function point analysis based model are:

- function points (FP) can be estimated from requirements specifications or design specifications, so using FPA it possible to estimate development cost in the early phases of development.

- function points are independent of the programming language or the methodologies used for software implementation.

- since function points are based on the system user's external view of the system the non-qualified users have a better understanding of what function points are measuring

Different variations of Function Points have emerged over the years, such as Object Oriented Function Points, Use Case Function Points and so on. Function point estimation approach is widely used within COCOMO II because COCOMO II is oriented to the newer software paradigms and to the modern software life cycles.

## III. A Hybrid Software Cost Estimation Approach

Our approach is a combination between almost all COCOMO models: Basic COCOMO, Intermediate COCOMO, and COCOMO II with Function Point Estimation features. The raison to develop such hybrid approach collecting all these mentioned above methods is to give the estimators an opportunity for a suitable choice of cost estimation model, depending of the concrete project type and the specific and often incomplete initial knowledge about the software product in the early stages of its development.

The two basic steps, required to accomplish software estimation are:

- Estimate product size,
- Estimate effort applied, project’s duration and resources needed.

### *A. Estimate product size*

Our approach bases the calculation of required effort PM on the software project's size measured in COCOMO II SLOC (and thousands of SLOC, i.e. KSLOC). The calculation of SLOC (KSLOC) may be based on the expert’s estimation of the size of software project (if is possible to make such estimate) or on FP estimation. The usual Function point’s estimation procedure is based on information that is available early in the project life cycle. It begins with determining and classifying (by complexity level) the user functions as Inputs, Outputs, Files, Interfaces, and Queries (figure 1.). As a result, the Unadjusted Function Points (ФТ) quantity is calculated (figure 1). Next a Translation of Unadjusted Function Points (ФТ) into SLOC is realized. The unadjusted function points are converted into equivalent SLOC depending of a LangFactor of the language used. For example, the LangFactor [3] for Assembly language is 320SLOC/UFP, for C++ - 29SLOC/UFP, for Fortran 77 – 105SLOC/UFP, for Lisp – 64SLOC/UFP, for Pascal – 91 SLOC/UFP and so on.

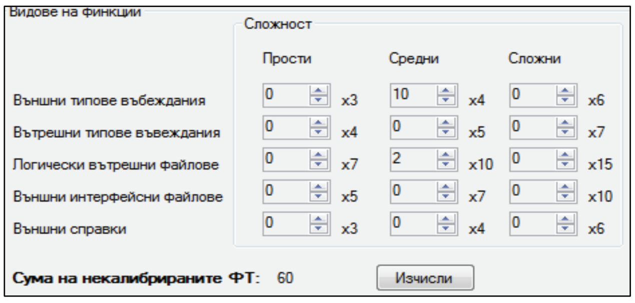
$$SLOC = \Phi T \times LangFactor \qquad (1)$$

The usual Function Point procedure accounts the degree of influence DI (2) of fourteen application characteristics (figure 2), such as distributed functions, performance, reusability, etc. The ratings of these 14 characteristics (rating scale of 0.0 to 0.05 for each characteristic) are added together, and added to a base level of 0.65 to produce a general characteristics adjustment factor that ranges from 0.65 to 1.35.

$$DI = \sum_{1}^{14} rating_i \qquad (2)$$

Our approach has respected this described above usual Function Point procedure to calculate the size of the project. The final equation that is used for cost estimates is shown below:

$$SLOC = \Phi T \times (0.65 + (0.01 \times DI)) \times LangFactor \qquad (3)$$



Видове на финкции

Сложност

| | Прости | | Средни | | Сложни | |
|---|---|---|---|---|---|---|
| Външни типове въбеждания | 0 | x3 | 10 | x4 | 0 | x6 |
| Вътрешни типове въвеждания | 0 | x4 | 0 | x5 | 0 | x7 |
| Логически вътрешни файлове | 0 | x7 | 2 | x10 | 0 | x15 |
| Външни интерфейсни файлове | 0 | x5 | 0 | x7 | 0 | x10 |
| Външни справки | 0 | x3 | 0 | x4 | 0 | x6 |

**Сума на некалибрираните ФТ:** 60

Fig. 1. FP calculation

### *B. Estimate effort applied, project’s duration and resources needed*

The general equation that we have used to calculate the effort needed (PM) for a given size project development, expressed as person months is given below:

Where:

$$PM = EF \times EAF \times KSLOC^{ee}[person - months] \qquad (4)$$

$$PM = \begin{cases} PM_{nom} & if \quad EAF = 1; \\ PM_{real} & if \quad Effort\ AdjasmentFactor\ is\ caclulated \end{cases}$$

If the effort adjustment factor EAF is 1 (it is its default value) PM is interpreted as the nominal effort $PM_{nom}$ needed for a given size project development, expressed as person months. The values of the coefficient EF and the exponent ee in this case are based on Intermediate COCOMO model.

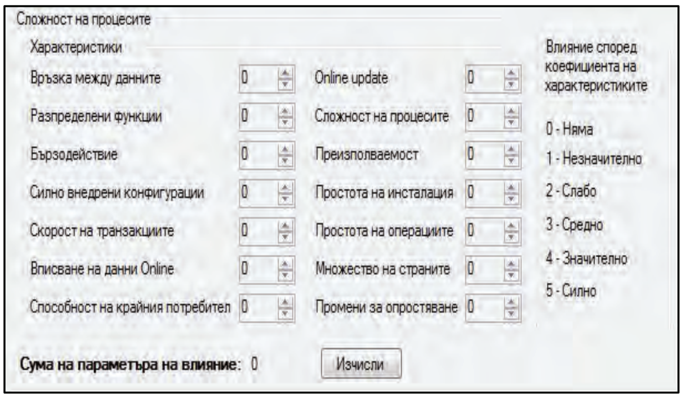
The calculation of the effort adjustment factor EAF (5) is related with the calculation of the adjusted effort $PM_{real}$. EAF estimation could be based on the fifteen COCOMO Intermediate cost drivers or on the seventeen COCOMO II Cost Drivers plus one. Total of eighteen Cost Drivers in the latter case are grouped into 3 major categories “Personnel attributes”, “Project attributes” and “Product attributes”. An additional user defined cost driver, named USER is added to the classic COCOMO II Cost Drivers. It gives estimators an opportunity to recognize the impact of a chosen project-specific factor, other than the provided in COCOMO II.

EAF for a given project is calculated as the product of the effort ratings of these attributes.

$$EAF = \prod_{1}^{CDN} EM_i \qquad (5)$$

Where,

$$CDN = \begin{cases} 18 & if \quad COCOMO\,II \text{ is used;} \\ 15 & if \quad COCOMO \text{ Intermediate } COCOMO \text{ is used.} \end{cases}$$



Сложност на процесите

| Характеристики | | | | Влияние според коефициента на характеристиките |
|---|---|---|---|---|
| Връзка между данните | 0 | Online update | 0 | |
| Разпределени функции | 0 | Сложност на процесите | 0 | 0 - Няма |
| Бързодействие | 0 | Преизползваемост | 0 | 1 - Незначително |
| Силно внедрени конфигурации | 0 | Простота на инсталация | 0 | 2 - Слабо |
| Скорост на транзакциите | 0 | Простота на операциите | 0 | 3 - Средно |
| Вписване на данни Online | 0 | Множество на страните | 0 | 4 - Значително |
| Способност на крайния потребител | 0 | Промени за опростяване | 0 | 5 - Силно |

**Сума на параметъра на влияние:** 0

Fig. 2. Application characteristics and DI calculation

The calculation of the duration TDEF of the project is based on the effort predicted by the effort equation:

$$TDEV = EF \times (PM)^{SchedExp}[months] \qquad (6)$$

Where:

PM is the effort (nominal or real) that is calculated, SchedExp is the schedule exponent derived from Basic COCOMO model and EF is a coefficient derived from Intermediate COCOMO model.

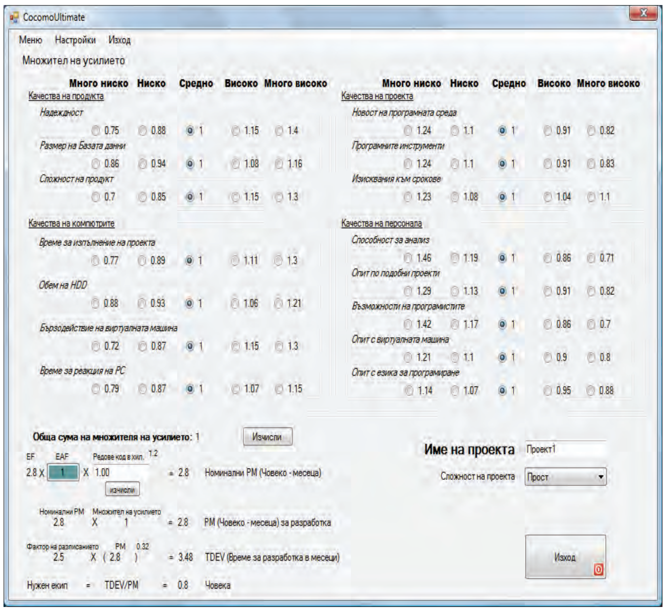
The average staffing is calculated as follows:

$$Average\ Staffing = {TDEV}\!\left/{PM}\right.[people] \qquad (7)$$

## IV. CONCLUSION

This paper gives a comparative overview of COCOMO and FPA models, discussing their advantages and disadvantages and proposes a hybrid cost estimation approach that combines their strengths. Our observation is that an approach that collects all these mentioned above methods gives the estimators an opportunity to choose the appropriate estimating method in a situation of often incomplete specifications and unclear requirements in the early stages of the project life cycle.

An interactive and flexible tool (figure 3) that implements the software estimation approach, discussed above, was developed. Depending on the specific characteristics of the project, the estimator can choose the appropriate sizing metric and method of cost estimation. The experiments prove that is not reasonable to use SLOC as sizing metric, but it is not also reasonable to use Function points as sizing metric for low level language projects estimation or for legacy system's estimation. Although the results are encouraging and match expectations for the tested projects, research must continue in the direction of evaluating large and complex projects.



Множител на усилието

| | Много ниско | Ниско | Средно | Високо | Много високо |
|---|---|---|---|---|---|
| Качества на продукта | | | | | |
| Надеждност | 0.75 | 0.88 | [✓ 1] | 1.15 | 1.4 |
| Размер на Базата данни | 0.86 | 0.94 | [✓ 1] | 1.08 | 1.16 |
| Сложност на продукт | 0.7 | 0.85 | [✓ 1] | 1.15 | 1.3 |
| Качества на компютрите | | | | | |
| Време за изпълнение на проекта | 0.77 | 0.89 | [✓ 1] | 1.11 | 1.3 |
| Обем на HDD | 0.88 | 0.93 | [✓ 1] | 1.06 | 1.21 |
| Бързодействие на виртуалната машина | 0.72 | 0.87 | [✓ 1] | 1.15 | 1.3 |
| Време за реакция на PC | 0.79 | 0.87 | [✓ 1] | 1.07 | 1.15 |

| | Много ниско | Ниско | Средно | Високо | Много високо |
|---|---|---|---|---|---|
| Качества на проекта | | | | | |
| Новост на програмната среда | 1.24 | 1.1 | [✓ 1] | 0.91 | 0.82 |
| Програмните инструменти | 1.24 | 1.1 | [✓ 1] | 0.91 | 0.83 |
| Изисквания към срокове | 1.23 | 1.08 | [✓ 1] | 1.04 | 1.1 |
| Качества на персонала | | | | | |
| Способност за анализ | 1.46 | 1.19 | [✓ 1] | 0.86 | 0.71 |
| Опит по подобни проекти | 1.29 | 1.13 | [✓ 1] | 0.91 | 0.82 |
| Възможности на програмистите | 1.42 | 1.17 | [✓ 1] | 0.86 | 0.7 |
| Опит с виртуалната машина | 1.21 | 1.1 | [✓ 1] | 0.9 | 0.8 |
| Опит с езика за програмиране | 1.14 | 1.07 | [✓ 1] | 0.95 | 0.88 |

**Обща сума на множителя на усилието:** 1

```
EF     EAF    Редове код в хил. 1.2
2.8 X   1   X  1.00             = 2.8    Номинални PM (Човеко - месеца)

Номинални PM  Множител на усилието
    2.8          X      1       = 2.8    PM (Човеко - месеца) за разработка

Фактор на разписанието   PM   0.32
    2.5            X  ( 2.8    )   = 3.48   TDEV (Време за разработка в месеци)

Нужен екип   =   TDEV/PM        = 0.8    Човека
```

**Име на проекта** Проект1

Сложност на проекта: Прост

Fig. 2. A tool for Software Cost Estimation (the main window)

## REFERENCES

[1] B.W. Boehm et al, "The COCOMO 2.0 Software Cost Estimation Model", American Programmer, 1996, pp.2-17.

[2] Boehm, B.W. "Software Engineering Economics", Prentice_Hall, 1981.

[3] COCOMO II Model Definition Manual, ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo97docs/modelman.pdf.

[4] Karen Lum et al. "Handbook for Software Cost Estimation", Jet Propulsion Laboratory, Pasadena, California, 2003.

[5] Liming Wu, "The Comparison of the Software Cost Estimating Methods", ttp://www.compapp.dcu.ie/~renaat/ca421/LWu1.html