

## A METHODOLOGY FOR SOFTWARE COST ESTIMATION

Ali ARIFOGLU

Norwegian Institute of Technology<sup>1</sup>  
Division of Computer Systems and Telematics  
Trondheim-NORWAY

### Abstract

Software Cost Estimation is an empirical process to be applied to find out basically the effort and development time requirements for the software product which is going to be developed. The process starts with the planning phase activities and refined throughout the development. It is very important for managing and scheduling the software project. Various cost estimation methods are available to be used for software development process. Depending on the size of the software, a macro ( for the information systems requiring more than 30 man years to develop) or a micro cost model can be used for estimation. The paper discusses available work on the cost estimation methods and proposes a methodological view in cost estimation. Basically, the methodology involves; how to combine available cost estimation techniques to have better estimation and, how to apply the methodology through software development. In the application of the cost estimation techniques, SD (Structured Development/Design) methodology is assumed as applied during the development. Some of the observations in the experiments are also given in the paper.

**Key Words :** Cost Estimation, Cost Refinement, COCOMO, Function Points, Esterling, Structured Development

### 1. Methods/Approaches for Cost Estimation : An Introduction

Cost estimation can be defined as ratherly an empirical process of estimating the effort and time costs for any software product before developing it. It is very important for project scheduling and management. Shooman [83] categorizes the approaches to cost estimation into four groups as :

- . **Unit Cost or Price :** Estimate the cost for each sub unit (e.g., a program, an instruction, a module etc.)
- . **Percentage of total cost :** Estimate the cost of the software cost of the whole software system of the organization.
- . **Specific Analogy :** Determining the similarities between the software system under development and previously developed ones, estimations can be performed.
- . **Parametric Equations :** Generally by applying statistical techniques to the historical data obtained from past developments parametric cost equations are obtained. They are empirical and the constants used in those equations should be revised after developments.

First three group involves ratherly subjective estimations compared to the last one. The experience of people doing the estimation gains more importance in the first three classes. However, the last one, parametric equations, includes a more structured or better formulated estimations since those equations have been constructed by application of various numerical analysis techniques on the data from present and past software projects. Therefore, such techniques are preferred in the paper.

Either macro or micro cost estimation models are available [Londeix,87]. The Micro model starts estimation by requiring detailed information about sub units of the software and continues with the higher levels ( i.e. they are bottom-up models ). Hence, micro models results

---

<sup>1</sup> Dr. Arifoglu has a research scholarship from Norwegian government for 9 months study in Norwegian Institute of Technology, started in October 1992. He is on his sabbatical in Norway for now and has his actual duty in the Department of Computer Engineering, Middle East Technical University, Ankara, Turkiye as an instructor.

with more accurate estimation. The macro model which is applicable to large-sized software products (e.g. more than 30 person-year) follows a top-down approach. It gives broader estimations for overall software development.

Available cost estimation techniques concentrates fundamentally on two aspects of software products; one is size and the others are effort and time. In the application, no relation among various techniques has been established and mostly techniques are applied individually. Main core of the paper is to explain how different methods can be involved in a methodological order for better cost estimation. For this reason, firstly the most important and popular cost estimation techniques which are proposed to be used in the application of the methodology are summarized and then the methodology is discussed. As an example, the application of the methodology during software development by Structured Design/Development by Yourdon[89] is explained. Finally an automated tool supporting the methodology is given.

## 2. Size Estimation Methods

These methods includes some techniques for estimating the number of source statements for a given software product. It is not easy to estimate the number of source statements directly. Therefore indirect methods are preferred in the early stages of the development for estimation. Two popular methods are Lines of Code (LOC) and Function Points Software Science methods. They are summarized below.

### 2.1 Lines of Code

LOC measure is the oldest estimation technique in cost estimation. It can be applied in either following unit-cost-per price or specific analogy approaches above. There are many discussions and speculations on the technique. Firstly, the definition of a line of code is not clear. As a simple example, should a comment or a blank line be encountered as a LOC or not? Some authors are encountering the comments while some others not. Jones [86] gives detailed discussions about the anomalies of the technique and shows how it can be used effectively. In the case of recursive programs, the technique fails since a recursive program is generally short but it is ratherly complex to develop. LOC is a bottom up technique in general. That means, for each smaller unit (a module, a sub program or a program) of the software being developed, the LOC estimations are performed and then they are summed up. Some experimental measures such as cost per LOC in \$, # LOCs per day per programmer etc, to convert the LOC results into cost units which may be time, number of personnel and money. The word NCSS (Non-Commentary-Source-Statements) are also being used by some authors.

### 2.2 Function Points

Function points method is firstly proposed by Albrecht and Gaffney [83]. In this method , the size of a system can be computed by three components which are information processing size and Technical complexity adjustment factors. Those components are analyzed below :

**information processing size :** This size is determined by identifying system components as five types namely external or logical inputs, outputs, inquiries, external interfaces to other systems and the logical internal files. These components are further weighted as "simple", "average" or "complex" depending on their characteristics. Then the sum of all components is called as Unadjusted Function Points (UFP). The weights are given in Figure 1 below [ Symons,88]. Note that the total field in the figure is obtained by multiplication of count by chosen weight ( simple, average or complex).

**Table 1** Calculation of Unadjusted function points

Count	Weight			
	S	A	C	
Internal input	3	4	6	
External output	4	5	7	
Files	7	10	15	
ext. interfaces	5	7	10	S : Simple
inquiries	3	4	6	A : Average
				C : Complex

**Technical complexity adjustment :** This adjustment is performed by calculating the technical complexity factor (TCF). TCF is calculated by rating the 14 questions about the features given in Table 5.1 from 0 to 5 where 0 : not present or no influence, 1 : insignificant influence, 2 : moderate influence, 3 : average influence, 4 : significant influence, 5 : strong influence or essential, and summing them up.

**Table 2** Elements for technical Adjustment Factor

E1. Reliable Backup and recovery	E8. On-line Update
E2. Data Communications	E9. Interface Complexity
E3. Distributed Functions	E10. Reusability
E4. Performance	E11. Process complexity
E5. Operational environment	E12. Installation Ease
E6. On-line	E13. Multiple Sites
E7. Multiple Screens for Input	E14. Ease of Use

$$TCF = \sum_{i=1}^{14} E_i$$

Consequently function points of the system can be calculated by the following empirical formula

$$FP = UFP \times (0.65 + 0.01 \times TCF)$$

After function points have been calculated, it is very easy to convert them to NCSS by using the Table 3 which is obtained empirically by Albrecht and Gaffney [83] for COBOL, PL/I and 4GL's. The number given for Pascal is derived by Arifoglu[92] studying on the projects which are about 50,000 lines of Pascal code.

**Table 3** FP to NCSS conversion

Language	NCSS/FP
COBOL	110
PL/I	65
Pascal	55
4GL	25

Although, there are some critics on function points method [Symon ,88], stating several deficiencies such that environmental factors are not considered, it is difficult to estimate information processing size at early stage of development etc., hence it is the mostly used method today. Symon's calibration on the function points method has not verified and well tested yet, but it sounds.

In the application Function Points methods are being used as stand alone in estimations. There are several empirical measures such that, function points per day, cost of one function point. In addition, function points method can be used for measuring the productivity of development activities [Behrens,83]. The formula

$$\text{ProductivityIndex (PI)} = \frac{\text{ActualHoursPerFunctionPoint}}{\text{ExpectedHoursPerFunctionPoint}}$$

is used for calculating the productivity.

### 3. Methods for Estimation Effort and Time

Two well known cost estimation techniques, one micro model, COCOMO (Constructive Cost Model) and one macro model, PUTNAM are summarized in this section. Esterling Time study Model is also explained.

#### 3.1 Constructive Cost Model ( COCOMO )

COCOMO is a set of hierarchical submodels which are basic, intermediate and detailed submodels. Basic model aims a quick rough estimate from small to medium sized project. Intermediate model adds more accuracy to the basic model by involving some attributes related to product, computer to be used, project personnel and project itself ,in the calculations. Finally the detailed model adds two extra capabilities namely phase sensitive effort multipliers and three level product hierarchy.

**Table 4** Cost formulas for basic and intermediate models

model	mode	manpower (K) (man-months)	development time (t) (months)
basic	organic	$K=2.4 \times S^{1.05}$	$t=2.5 \times K^{0.38}$
	embedded	$K=3.6 \times S^{1.20}$	$t=2.5 \times K^{0.32}$
	semi-det.	$K=3.0 \times S^{1.12}$	$t=2.5 \times K^{0.35}$
inter- mediate	organic	$K=3.2 \times S^{1.05}$	$t=2.5 \times K^{0.38}$
	embedded	$K=2.8 \times S^{1.20}$	$t=2.5 \times K^{0.32}$
	semi-det.	$K=3.0 \times S^{1.12}$	$t=2.5 \times K^{0.35}$

S : Number of estimated source statements in thousand

On the other hand, the projects under consideration are classified into three modes as organic, embedded and semi-detached modes. Familiar environment, small team of experienced programmers are main characteristics of organic mode projects. In the embedded mode, projects has tight constrains, more than one processor and various types of peripherals exist. Uncertainties in the problem definition is another characteristics of such projects. Semi-detached mode projects are intermediate projects between organic mode and embedded mode. Cost formulas for basic and intermediate models are summarized in Table 4.

#### 3.2 Putnam's Model

This is a macro model for estimation and gives good results in planning and estimating large projects ( e.g. they require more than 30 work-years effort) . Putnam uses Rayleigh-Norden curve for estimation [Putnam,78] :

$$S = E \times K^{\frac{1}{3}} \times t^{\frac{4}{3}}$$

where

S is the size of the software in NCSS,  
K is the manpower cost (in man-years),  
t is development time (in years ),

**E** is technology factor or environment factor. This technology factor (**E**) may be chosen as 2 for a poor software development environment, as 8 for good software environment ( e.g., methodology in place, adequate documentation etc.) and as 11 for an " excellent" environment (i.e., automated tools and techniques are in use for software development) or it can be determined by the organization using information from past projects and the following formula:

$$E = \frac{S}{K^{\frac{1}{3}} \times t^{\frac{4}{3}}}$$

### 3.3. Esterling Time Study Model

Esterling takes microscopic characteristics of the work environment [Esterling,80]. The parameters used in the model are :

- a : Average fraction of workday spent on administrative or other indirect work
- t : Average duration of working interrupts (minutes)
- r : Average recovery time after interruptions (minutes)
- k : Number of interruptions/workday from project personnel
- p : Number of interruptions/workday from non-project personnel
- i : indirect (overhead) cost/person expressed as fraction of base pay
- d : Differential pay for overtime expressed as fraction of base pay
- n : Number of people working in the project
- g : Average number of overtime work-hours per workday

Values of some of those parameters were determined empirically. However an organization may use its own parameters as well Using those parameters. Empirical values for those parameters to be used in implementation phase are given in Table 5.5.

**Table 5** Values for Esterling Parameters

parameter	range	factory workers	programmers		
			optimistic	typical	pessimistic
a	0 - 0.5	0.0	0.05	0.10	0.15
t	1 - 20	3	3	5	10
r	5 - 10	0.5	0.5	2.0	8.0
k	1 - 10	1	2	3	4
p	1 - 10	1	1	4	10
i	1 - 3	0.2	0.2	0.5	1.0
d	1 - 2	1.5	1.0	1.0	1.5

- A factor representing useful working time is calculated and it is used for calculating
  - . ratio of calendar time to person-days to complete project
  - . labor cost per workday for an average base salary s
  - . cost efficiency
  - . project cost per person day.

Then the formulas are

$$w = 0.125 \times [ 8 - 8a + g - (4r - p(t+r) - k(n-1)(t+r))/60 ]$$

$$T = 7 / ( 5nw ) \quad \text{ratio of calendar time to person-days to complete the project}$$

$$c = ns(gd + 8(1+i)) \quad \text{labor cost per work day for an average base salary s}$$

$$e = nw / c \quad \text{cost efficiency}$$

$C = c / (nw)$  Project cost per person-day

#### 4 Normalization of the Measures

For obtaining more accurate estimation, three values should be estimated for FP or for NCSS as optimistic, pessimistic and most likely values. If we represent those values by a, b and m respectively, then expected estimation value can be obtained by [Pressman,92]

$$E = \frac{a+4m+b}{6}$$

and it will have the standard deviation on module size as

$$s = \frac{b-a}{6}$$

which means ; " provided that no change occurred in the product requirement definition, the size of the module to be developed will have a probability of  
 99.8 % of being between  $E - 3s$  and  $E + 3s$ ,  
 95 % of being between  $E - 2s$  and  $E + 2s$ ,  
 68 % of being between  $E - s$  and  $E + s$  ".

This is also known as *beta normalization* in statistics and can be applied together with any of the cost estimation techniques.

#### 5. Other Cost Estimation Techniques

**Table 6** Some of the Other Cost Models/Techniques

Name of the Model	Organization	Year	Original References
Delphi	Rand Co.	1966	[Helmer, 66]
Nelson's SDC	SDC	1966	[Nelson, 66]
Wolverton	TRW	1974	[Wolverton, 74]
RCA Price-S System	RCA	1976	[Freiman&Park, 79]
Halstead's Software Science	-	1977	[Halstead, 77]
Walston and Felix Model	IBM	1977	[Walston&Felix, 77]
Parr Model	-	1980	[Parr, 80]
SOFTCOST	JPL	1981	[Tausworthe, 81]
Bailey and Basili Model	NASA	1981	[Bailey&Basili, 81]
Bang Metrics	-	1982	[DeMarco, 82]
Jensen Model	-	1984	[Jensen, 84]
Mark II Function Points	-	1988	[Symon, 88]
Pfleeger Model	-	1989	[Pfleeger, 91]

Some of the other cost estimation techniques/models are listed in Table 6 in chronological order with their original references. They are not as popular and widely uses as the models mentioned in the previous sections in the paper. They can be termed as "research" and "organization" models since they are mostly developed and tested in specific organizations or they are the results of the academic studies. They need to be more general and more application experiment needed for them. However, they can also be used in the application of the methodology underneath. Conte et al [86] give a comparative study of most of the techniques above.

#### 6 A Methodology for Cost Estimation

##### 6.1 The Methodology

The motivation for such a methodology can be explained as the lack of integrated cost estimation techniques to be used in software development process. Available cost estimation techniques are mostly being applied as stand alone in today's applications. Most of the techniques are organization specific and the parameters used in those techniques are environment specific also. A more general and integrated approach which uses the mostly

accepted and used cost estimation techniques are needed. The methodology explained here was firstly introduced in Aktas and Arifoglu [89] for planning phase activities only. Here it is more generalized and extended to all software life cycle. Basically, the methodology proposes the use of a set of cost estimation methods to be applied step by step and as an integrated way to achieve better project planning and scheduling. Basic steps of the methodology can be stated as follows:

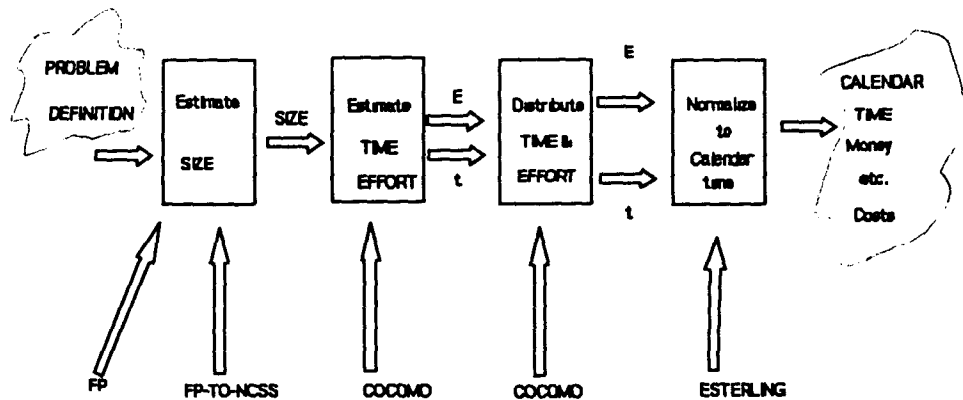


Figure 1. Cost Estimation Methodology

- Step 1. Estimate Size
- Step 2. Estimate effort and time costs
- Step 3. Distribute effort and time costs to the life cycle
- Step 4. Normalize Costs to actual calendar time

#### Step 1 : Size Estimation

In this step, basic activity is to make estimation about the size (i.e The number of non-commentary source statements) of the software product under consideration. The methodology proposes the usage of the before mentioned techniques of Function Points and Conversion of Function points into NCSS for size estimation. Depending on the methodology followed and the stage of the development, various other special size estimation techniques can be applied. Basic output of this step is the NCSS estimate for the software product which is going to be developed.

#### Step 2 : Effort and Time Estimation

COCOMO and its variations are proposed to be used to estimate effort and time in terms of work-months and months respectively.

#### Step 3 : Distribution to Life Cycle

Some techniques based on experience in the software production environment and some guidelines like given in COCOMO technique may be used for distribution.

#### Step 4 : Normalization to Calendar Time

Esterling is a very good candidate for this type of normalization. The method needs to be improved by some parameters for systems analysts and other development staff. The outputs are expected to be the calendar time and money costs.

After the application of Step 4, one of the automated project management packages can be used for managing and scheduling the project. They are mostly implemented on

microcomputers [ Levine,86]. Basic functions of those packages are

- . controlling time schedule of the project,
- . Resource planning,
- . Production of project budget,
- . Cost refinement.

Basic differences among those packages comes from number of activities handled, flexibility, reporting facilities and variety of functions performed [Csaba,87;Sevkal,88].

An estimation method is successful when it is easy to understand, refine able during software development process and the early estimation is within +/- 30% of the actual final cost. Therefore, cost estimation process should not be applied only once in entire development process phase. It should be repeated at various stages throughout the development process. After each repetition, the results of estimation should be refined by comparing with previous estimations. The stages of applying the cost estimation methodology may vary depending on various factors such as the model and or methodology used in development and the size of the software product etc. In the following section, an application of the methodology during software development by the approach of Structured Design/Development by Yourdon[89] is discussed generally.

## 6.2 Application in Structured Design/Development

Structured Design/Development methodology for software development was firstly proposed by Yourdon [89]. The methodology uses the following methods in software life cycle :

1. Planning Phase : Use very general Data flow Diagrams (DFD) and narrative texts for problem definition.
2. Analysis Phase : DFD's are main tools to represent the logical design of the software
3. Design Phase : Structure Charts and Pseudo Code for physical Design
4. Implementation : Relevant software tool, a programming language, a database management system or a 4GL for coding the software.

The cost estimation methodology is proposed to be applied once at least in each phase of SD. However depending on the size of the software and work breakdown structure, the cost estimation methodology can be applied many times in one or more phases of SD. Here, to be more general, four major phases of SD are considered for cost estimation.

*Planning Phase* : At the beginning of the planning phase, application of the CE methodology is shown in Figure 1. Firstly function points are calculated from the problem definition in general. Then Table 3 given in section 2.2 is used for FP-to-NCSS Conversion. COCOMO takes the NCSS as input and produces effort and time results which can be used as basic Inputs to Esterling. During the implementation of the activities of the planning phase, the observations about time and effort should be recorded for refining the costs at the end. This is important for the scheduling of the subsequent phases of the life cycle.

*Analysis Phase* : During the analysis phase the observations related to the cost should be recorded for comparison to the previous estimations. One suitable stage for application of the CE methodology may be the end of the phase. Such an application will be very helpful for planning and scheduling the further phases which are design and implementation. Major inputs of the cost estimation process are DFD's of the software system to be developed. At the end of the analysis phase, CE methodology runs on DFD's and other analysis data. The only difference in applying the CE methodology from the previous phase is that the function points of the system can be obtained directly from DFDs semi-automatically by the following algorithm:

1. Find the number of *external interfaces* by analyzing sources and sinks in Context diagram
2. Find the number of *inputs and outputs* by analyzing the flows in Overview diagrams,
3. Find the number of *files* by counting the data stores in Overview and Detail Diagrams
4. Find number of *inquiries* by counting the processes in Overview diagrams

Other information such as the weights of the items above and answers to the questions about technical adjustment factor should be revised manually.

*Design Phase* : A suitable application area for cost estimation may be the end of the design study for the software product. Here, at this stage there is no more need to start up with the function points technique to estimate the NCSS. Since, basic deliverables of the design phase are the structure charts, pseudo-codes, report/screen forms and data dictionary/directory (DD/D), an empirical formula can be obtained from them easily for estimating NCSS for implementation. The formula may have the following form:

$$NCSS_d = \sum_{i=1}^4 W_i \times D_i$$

where

- NCSS<sub>d</sub> : NCSS Estimate of the design,  
 W<sub>i</sub> : The Weight of E<sub>i</sub>,  
 E<sub>i</sub> : i th Design Deliverable such that,  
     E<sub>1</sub>: Number of Pseudo-Code lines of Design,  
     E<sub>2</sub>: Number of Report Forms,  
     E<sub>3</sub>: Number of Screen,  
     E<sub>4</sub>: Number of DD/D Entries.

The weights may be dependent on the software development tools to be used in implementation. For example, E<sub>1</sub> value should be taken differently in implementation with Pascal than in a 4GL. Still, those weights needs more application experimented. In some of the student projects, E<sub>1</sub>=25, E<sub>2</sub>=E<sub>3</sub>=20, and E<sub>4</sub>=2 sounded well for an ongoing project SMIDO, which involves the automation of the information network between small and medium sized enterprises in Turkiye [SMIDO,92].

The NCSS calculated as above is used in COCOMO estimations directly. The environmental factors of COCOMO should be revised manually or previously used values can be used automatically. Esterling model is also applied in a similar manner to COCOMO. The results are the calendar time, money and other estimates for the project and so does the implementation phase.

*Implementation Phase*: During the implementation phase work, observed cost parameters are recorded and at the end, the actual results are compared to the previous estimations to see why and where (if any) deviations happened. The new knowledge is stored kept to be used in further estimations.

### 6.3 An Automated Tool

An automated tool implementing the methodology discussed above was developed as a part of the Measurement and Evaluation Package, MEIS [Arifoglu,91]. The Cost estimation module of MEIS includes the automation of the methods of Function Points, FP-to-NCSS Conversion, COCOMO and Esterling as integrated. The tool is developed by Turbo Pascal 5.0 in microcomputer environment.

## 7. Summary and Further Work

Studies on the software cost estimation are briefly surveyed and a new methodology proposed in this paper. Basically, the methodology proposes an integrated usage of available cost estimation techniques in different phases of the development life cycle. Application of the methodology in developing software by SD approach is also discussed. As a next step, the methodology should be experimented in various projects and thus will be enhanced. Some applications are also needed for applying the methodology to other software development approaches such as object-oriented development, Jackson System Development & Programming (JSD & JSP) etc. Also, the tool needs to be ported to unix environment and should be enriched by the automation of cost estimation techniques other than mentioned in the paper.

### References

- [Aktas&Arifoglu,89] Aktas, A. Z., Arifoglu A., "Information Systems Planning : Tools, Methods and a Methodology", Proceedings Intern. AMSE Conference Signals&Systems", Brighton, July 12-14, 1989, AMSE Press., Vol.3, pp.51-63
- [Albrecht,83] Albrecht, A. J., Gaffney, J. E., "Software Function, Source Lines of Code and Development Effort Prediction : A Software Science Validation," IEEE Trans. Software Eng., Nov 1983, pp. 639-648
- [Arifoglu,91] Arifoglu A., "Measurement and Evaluation of Information Systems and an Application: MEIS Package", Ph.D Thesis, Dept. of Computer Eng., METU, Ankara, August 1991
- [Arifoglu,92] Arifoglu, A., "A Function Point Study for Pascal", Technical Paper, Dept. of Comp.Eng. METU, Ankara, July 1992
- [Bailey&Basili,81] Bailey, J. W., Basili, V.,R., "A Metamodel for Software Development Resource Expenditures" Proceedings of the Fifth International Conference on Software Engineering, 1981, pp. 107-116
- [Behrens,83] Behrens C.A., "Measuring the Productivity of Computer Systems Development Activities with Function Points", IEEE Trans. on Soft. Eng., Vol-Se 9, No.6, Nov 1983, pp. 648 -652
- [Conte et al, 86] Conte, S. D., Dunsmore, H. E., Shan, V., Y., Software Engineering, Metrics and Models, Benjamin/Cummings, 1988
- [Csaba,87] Csaba E., J., "Project Management Tools for Software Development", Auerbach Systems Development Management (ASDM), 34-01-60, 1987
- [DeMarco,82] DeMarco, T., Controlling Software Projects, Yourdon Press, New York, 1982
- [Esterling,80] Esterling, R., "Software Manpower Costs : A Model," Datamation, March 1980
- [Freidman&Park,79] Freidman, F., R., Park, R., E., "Price-S Software Model Overview", Internal Paper, RCA, Cherry Hill, NJ, 1979
- [Halstead,77] Halstead, M., Elements of Software Science, Elsevier North Holland, 1977
- [Helmer,66] Helmer-Heidelberg, O., Social Technology, Basic Books, New York, 1966
- [Jensen,84] Jensen, R.,W., "A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models", Proceedings of Internal Society of Parametric Analysis", 1984, pp.96-106
- [Jones,86] Jones, C., Programming Productivity, McGraw Hill, New York, 1986
- [Jones, 88] Jones, Meilir Page, The Practical Guide to Structured Systems Design, Yourdon Press, USA, 1988
- [Levine,86] Levine, Harvey A., Project Management Using Microcomputers, McGraw Hill, Osborne, 1986
- [Londeix,87] Londeix, B., Cost Estimation for Software Development, Addison-Wesley, Great Britain, 1987
- [Nelson,66] Nelson, E., A., "Management Handbook for the Estimation of Computer Programming Costs", AD-A648750, SDC, Oct, 1966
- [Pressman,87] Pressman,R.S., Software Engineering, A Practitioner's Approach, McGraw Hill, Singapore, 1987
- [Parr,80] Parr, F.N., "An Alternative to Rayleigh Norden Curve Model for Software Development Effort", IEEE Trans. on Software Eng., SE-6(3), May 1980
- [Pfleeger,91] Software Engineering : The Production of Quality Software, MacMillan, USA 1991
- [Putnam,78] Putnam,L., "A General Empirical Solution to Macro Software Sizing and Estimating Problem", IEEE Trans. on Software Eng., SE-4(4), 1978, pp. 345-361
- [Sevkal,88] Sevkal S., " Yoneticilere Yonelik Bilgisayar Kullanimi", Bakis Bilgisayar ve Teknoloji Dergisi, Yil:3, Sayi:9, Nisan-Haziran 1988, Sa:12-14
- [SMIDO,92] Arifoglu et al., Systems Analysis and Design Reports for Information Centers of SMIDO, (2 Reports), Dept. of Computer Eng., METU, 1992
- [Shooman,83] Shooman L., M., Software Engineering, McGraw Hill, Japan, 1983
- [Symons,86] Symons R., C., "Function Points Analysis, Difficulties and Improvements", IEEE Trans. on Soft. Eng., Vol-Se 14, No.16, Jan 1988, pp. 2-11
- [Tausworthe, 81] Tausworthe, R., C., "Deep Space Network Software Cost Estimation Model", Publication 81-7, Jet Propulsion Library, Pasadena, CA, 1981
- [Walston&Felix, 77] Walston, C.E., Felix, C.P., "A Method for Programming Measurement and Estimation", IBM Systems Journal, No.16, 1, 1977, pp. 54-73
- [Wolverton,74] Wolverton, R.,W., "The Cost of Developing Large Scale Software", IEEE Transactions on Computers, C-23, 6, 1974, pp.615-636
- [Yourdon, 89] Yourdon E., Modern Structured Analysis, Prentice Hall, USA, 1989