

Providing quality in estimating software development costs

A. Arifoglu

*Department of Computer Engineering, Middle East
Technical University, 06531 Ankara, Turkey*

ABSTRACT

Software Cost Estimation is an empirical process to find out basically the effort and time requirements for software development process. The paper gives a quick overview on the available cost estimation methods with a discussion related to their anomalies. Then it focuses on the methodological application of those techniques throughout the development process with successive refinements. Two of the software risks which are known as unrealistic schedule and budget risks are aimed to be minimized by a new cost estimation model presented in the paper.

Keywords : Software Cost Estimation

INTRODUCTION

Cost estimation can be defined as an empirical process of estimating the effort and time costs for any software product before developing it. It is very important for scheduling and managing the software development process. Merging the classifications of Shooman [27] and Fenton[11], we can categorize the cost estimation approaches into four groups as :

- Estimate by Expert judgment
- Estimate by Decomposition:
- Estimate by Analogy
- Estimate by Parametric Equations

First three groups involve rather subjective estimations compared to the last one. In these classes, the experience of people doing the estimation gains more importance. However, the last one, parametric equations, involves a more structured or better

270 Software Quality Management

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 favored in the paper.

Either *macro* or *micro* cost estimation models are available [19]. 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 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.

The paper, firstly, summarizes the up-to-date cost estimation techniques in a quick way, emphasizing on the mostly used and the most general ones and then describes a model to estimate and monitor the costs of software development process in detail.

AVAILABLE COST ESTIMATION METHODS: A QUICK OVERVIEW

Available cost estimation techniques focus basically on two aspects of software products; one is size and the others are effort and time.

Size Estimation Methods

These methods include 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 should be used in the early stages of the development for estimation. Two well-known examples of such methods which are Lines of Code (LOC) and Function Points are summarized below.

Lines of Code LOC measure is the oldest estimation technique in cost estimation. It can be applied in following *Estimate by Expert Judgment* approach 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, the question of "Should a comment or a blank line be encountered as a LOC or not?" has different answers by different authors. Jones [16] 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

relatively 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.

Function Points Function points method is firstly proposed by Albrecht and Gaffney [1]. In this method, the size of a system can be computed by three components which are information processing size and Technical complexity adjustment factors and Function Points. Relevant remarks about those components are given 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 used were empirically determined values from past projects by Albrecht [1]. The sum of the weighted component values is defined as Unadjusted Function Points (UFP).

Technical complexity adjustment: This adjustment is performed by calculating the technical complexity factor (TCF). TCF is calculated by rating the questions about some of the characteristics of software product to be developed.

Consequently **Function Points (FP)** of the system can be calculated by the following empirical formula

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

Function points and LOC measures correlate very well. FP can be easily converted to LOC by using experimental results from past projects [21]. For example FP/LOC ratio is 150 for COBOL, 65 for PL/I etc.

Although, there are some critics on function points method [28], stating several deficiencies such that environmental factors are not considered in the method, still it is one of the mostly used methods,

272 Software Quality Management

specifically in the early stage of software development today. Symon's calibration on the function points method has not verified and well tested yet, but it sounds. In the application, FP method is being used as stand alone in estimations. There are several productivity measures based on the technique such that, function points per day, cost of one function point etc. In addition, function points method can be used for measuring the productivity of development activities [5]. A slight variant of the *FP* method, *Feature Points* is also available for estimating the cost of real time software systems [16].

Methods for Estimating Effort and Time

Constructive Cost Model (COCOMO) COCOMO is a set of hierarchical sub models which are basic, intermediate and detailed sub models [6]. 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. Furthermore, On the other hand, the software projects are classified into three modes as organic, embedded and semi-detached and different estimation formulas are used for each. 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. General form of COCOMO formula

$$\begin{aligned} \text{effort} &= a \times (\text{size})^b \times c \\ \text{time} &= d \times (\text{effort})^e \end{aligned}$$

where a, b, d, e are constants depending on the project mode and the COCOMO model used, c is the cost multiplier obtained by rating different attributes of the software to be developed.

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 [24]:

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

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.

The technology factor (E) may be chosen as 2 for a poor software development environment, as 8 for good software environment (e.g., model 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:

$$E = S / (K^{1/3} \times t^{4/3}).$$

Other Cost Estimation Techniques

Table - I : Some of the other Cost Estimation Models/Techniques

Name of the Model	Year	Reference
Delphi	1966	[14]
Nelson's SDC	1966	[20]
Wolverton	1974	[32]
RCA Price-S System	1976	[12]
Software Science	1977	[13]
Walston and Felix Model	1977	[30]
Esterling	1980	[10]
Parr Model	1980	[22]
SOFTCOST	1981	[29]
Bailey&Basili Model	1981	[4]
Bang Metrics	1984	[15]
Feature Points	1986	[16],[17]
Mark II Function Points	1988	[28]
Pfleeger Model	1989	[23]
TIC	1991	[31]

Some of the other cost estimation techniques/methods are listed in Table-I in chronological order with their original references. They are not as popular and widely used as the ones mentioned in

274 Software Quality Management

the previous sections. 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 model underneath. Conte et al [7] give a comparative study of some of the techniques above.

A QUALITY MODEL FOR COST ESTIMATION

Motivation

Studying available cost estimation techniques, one can easily notes the following observations:

Cost estimation techniques are in an emerging stage today: Still, the cost estimation techniques are very dependent on the environmental characteristics. That is, the cost equations used are mostly organization dependent and/or research oriented. They are too far from being general-purpose. Only very few of them have matured compared to others.

Usage of "One" Cost Estimation technique: CE techniques are generally being used as stand alone in today's applications. This observation is correlated to the first one too much. The Software practitioners are generally using only one cost estimation technique in their estimations.

Cost Estimation Techniques are Macroscopic: Most of the CE techniques are using "macroscopic" characteristics of project environment and personnel in their estimation. As an example most of the techniques are accepting one work-day as eight hours which means software developers have to work like robots without any interruption in a day.

Estimating the cost "Once" in the development process: Cost Estimation process is generally interpreted as a sub task of planning phase of software or information systems development process. Hence, the cost of the software project is estimated in the very early stage of the development and resulting with higher degree of errors. There is not much about how to control the costs.

The need for Cost Estimation Cycle: 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 model varies depending on various factors such as the model used in development and the size of the software product etc. Furthermore, the software development process have different but integrated tasks which are characterized as life cycle phases where different specification methods are being used in each. Then, different cost estimation techniques may be useful because of the difference of the tasks to be performed in phases. They should also be used in an integrated way. The model presented in the paper proposes a *Cost estimation Cycle* imbedded in the software life cycle and describes its stages.

The Model

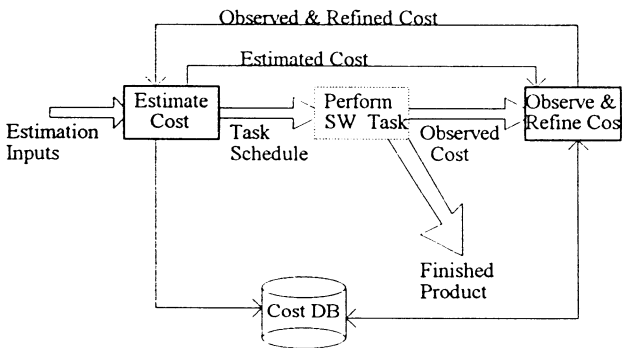


Figure 1. Software Cost Estimation Model

The software development process, is defined as a constitution of time-discrete and sequential stages where cost estimation and refinement processes are applied at the beginning and at the end of each. Each stage may include various software development tasks. While performing the tasks related to any of those stages, effort and time costs should be observed for better refinement.

As shown in Figure 1, the model simply consists of two steps as:

Step 1 : Estimate Cost,

Step 2 : Observe Cost Data and Refine Cost,

in one of its runs. Those steps should be applied as many times as

276 Software Quality Management

needed during the software development process. The number of runs of the model may be dependent upon several factors related to the software. Some of the most important factors are : the size of the project, the work breakdown structure of the project, the specification models used and the process models used. In figure 1, the software related task may be corresponded to any one of the development phases, say design phase as an example or to any sub phase determined by those factors above. The steps explained below in more detail.

Cost Estimation The approach explained here was firstly introduced in Arifoglu [2] for planning phase activities only. Later, it is extended to all of the life cycle phases [3]. Basically, the approach involves the use of a set of well known cost estimation methods to be applied step by step and as an integrated way to achieve better project planning and scheduling. The approach consists of the following four discrete and integrated steps as shown Figure 2:

- S1. *Estimate Size*
- S2. *Estimate effort and time costs*
- S3. *Normalize Costs to actual calendar time*
- S4. *Schedule the Project*

Various cost estimation techniques can be used in those steps, depending on the stage of the development process.

S1 - Size Estimation : In this step, basic activity is to make estimation about the size (i.e. The number of non-commentary source statements or LOC) of the software product under consideration. Basic output of this step is the NCSS estimate for the software product which is going to be developed.

S2 - Effort and Time Estimation : The size estimations obtained in S1 should be converted to *effort* and *time* units which are *work-months* (or work-days) and *months* (or days) respectively. Then, the number of people to be involved in development can easily be estimated as dividing the effort into time. The estimations performed in this steps are related to whole software development process. However, as stated earlier, the SD is regarded as a combination of the development stages. Therefore the estimated cost should be distributed to those stages. For this purpose various techniques may be used. As an example, COCOMO includes some guidelines for distribution.

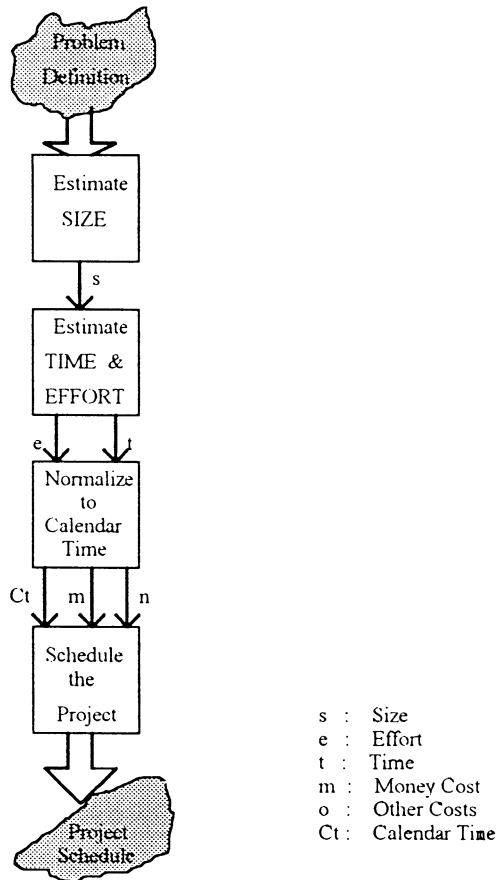


Figure 2. Cost Estimation

S3 - Normalization to Calendar Time : The results of S3 such as effort (in units of work-months or work-days) and time (in units of days or hours) need to be normalized to actual calendar time considering more microscopic characteristics of time for better accuracy in time, so does the money, estimations. As an example, a work-day is always interpreted as eight hours, but how much of it is being used efficiently in software production may depend on various environmental factors. Esterling Time Study Model [10] is a good candidate for this type of normalization. The outputs of the model are the estimations of calendar time and money costs.

S4 - Project Scheduling : 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

278 Software Quality Management

microcomputers, however, limited number of them are also available in mainframe computers [8,18,25]. Main differences among those packages comes from number of activities handled, flexibility, reporting facilities and variety of functions performed

Cost Observation and Refinement Cost observation and refinement process is required to control the costs and to achieve higher quality of cost management in software development. Otherwise cost refinement process becomes impossible. In this section, a formalization is given about what to observe during development, providing a systematic way.

The *observed effort*, e , for any stage, st_j , can be obtained by the following formula:

$$e(p_i) = \sum_{k=1}^{n_i} ot(p_k, ph_i)$$

where ot is the *observed time* function of which value will be obtained by measurements during the implementation of st_j for each person (p_k) separately.

The *time*, t_i , for each st_j can be observed as:

$$t_j = End(st_j) - Begin(st_j)$$

where $Begin(x)$ and $End(x)$ functions are simply giving the beginning and the end times of x respectively.

After those observations, the *time-effort efficiency*, te , of any development stage, st_j can be calculated as :

$$te(ph_j) = 1 - s(ph_j) / (n_j \times t_j)$$

where $s(st_j)$ is the slack time for the stage of st_j and calculated as:

$$s(st_j) = t_j \times n_j - e(st_j).$$

The time-effort efficiency values closer to 1 or greater than one are the indicators of good and efficient development work.

For each st_j , deviations (d_j 's) between observations (ϕ_j 's) and estimations (es_{ij} 's) should be calculated and new estimations for the

further stages of the development process should be performed when necessary. In Figure 3, this situation is shown. To avoid complexity, the notation of es_{ij} is used for both time and effort estimations and abs is for *absolute value* function.

	St_1	St_2	St_3		St_n	Deviations (d_i 's)
Estimation 1	es_{11}	es_{12}	es_{13}		es_{1n}	-
Estimation 2	o_1	es_{22}	es_{23}		es_{2n}	$d_1 = abs(o_1 - es_{11})$
Estimation 3	-	o_2	es_{33}		es_{3n}	$d_2 = abs(o_2 - es_{22})$
						
Estimation n			o_{n-1}	es_{nn}		$d_{n-1} = abs(o_{n-1} - es_{n-1})$
				o_{nn}		$d_n = abs(o_n - es_n)$

Figure 3. Estimations and Observations During software Development

For the overall evaluation, some of the statistical techniques such as ANOVA can be applied deviation values (d_i 's). The ideal and expected cases for those values are given below:

ideal case : $d_1 = d_2 = d_3 = = d_n = 0$

(only one successful estimation throughout the development process)

expected case :

$d_n < d_{n-1} < d_{n-2} < < d_1$
(good correlation between te and d values)

APPLICATION IN STRUCTURED DESIGN/DEVELOPMENT

Structured Design/Development methodology for software development was firstly proposed by Yourdon [33]. The model uses the following methods in software life cycle :

- Planning Phase : Use very general Data flow Diagrams (DFD) and narrative texts for problem definition.
- Analysis Phase : DFD's are main tools to represent the logical design of the software
- Design Phase : Structure Charts and Pseudo Code for physical Design

280 Software Quality Management

- Implementation : Relevant software tool, a programming language, a database management system or a 4GL for coding the software.

The cost estimation model 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 model 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 model is simply as is in Figure 2. Firstly function points are calculated from the problem definition in general. Then the guideline given by Jones [17] 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 method. 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 model 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 model runs on DFD's and other analysis data. The only difference in applying the CE model 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:

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

where

$NCSS_d$: NCSS Estimate of the design,

w_i : The Weight of D_i ,

D_i : i th Design Deliverable such that,

D_1 : Number of Pseudo-Code lines of Design,

D_2 : Number of Report Forms,

D_3 : Number of Screen,

D_4 : Number of DD/D Entries.

The weights may be dependent on the software development tools to be used in implementation. For example, w_1 value should be taken differently in implementation with Pascal than in a 4GL. Still, those weights needs more application experimented. The values of $w_1=25$, $w_2=w_3=20$, and $w_4=2$ sounded well in some of the student projects, and in an ongoing project SMIDO, which involves the automation of the information network between small and medium sized enterprises in Turkiye [26].

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.

282 Software Quality Management

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 to be used in further estimations.

An Automated Tool

An automated tool implementing the model discussed above was developed as a part of the Measurement and Evaluation Package, MEIS [2]. 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 a case tool, JSP-Tool, supporting Jackson's Structured Program Development approach. Generated code was in Turbo Pascal 5.0 in PC environment.

SUMMARY AND FURTHER WORK

Studies on the software cost estimation are briefly surveyed and a new model proposed in this paper. Basically, the model proposes an integrated usage of available cost estimation techniques in different phases of the development life cycle. Application of the model in developing software by SD approach is also discussed. As a next step, the model should be experimented in various projects and thus will be enhanced. Some applications are also needed for applying the model 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

1. 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
2. Arifoglu A., 'Measurement and Evaluation of Information Systems and an Application: MEIS Package', Ph.D Thesis, Dept. of Computer Eng., METU, Ankara, August 1991
3. Arifoglu, A., 'A Methodology for Software Cost Estimation', ACM

Software Engineering Notes, April, 1993

4. 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
5. 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
6. Boehm, B., W., Software Engineering Economics, Prentice Hall, 1981
7. Conte, S. D., Dunsmore, H. E., Shan, V., Y., Software Engineering, Metrics and Models, Benjamin/Cummings, 1988
8. Csaba E., J., 'Project Management Tools for Software Development', Auerbach Systems Development Management (ASDM), 34-01-60, 1987
9. DeMarco, T., Controlling Software Projects, Yourdon Press, New York, 1982
10. Esterling, R., 'Software Manpower Costs : A Model,' Datamation, March 1980
11. Fenton, N., E., Software Metrics, A Rigorous Approach, Chapman Hall, 1991
12. Freidman, F., R., Park, R., E., 'Price-S Software Model Overview', Internal Paper, RCA, Cherry Hill, NJ, 1979
13. Halstead, M., Elements of Software Science, Elsevier North Holland, 1977
14. Helmer-Heidelberg, O., Social Technology, Basic Books, New York, 1966
15. 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
16. Jones, C., Programming Productivity, McGraw Hill, New York, 1986
17. Jones, Meilir Page, The Practical Guide to Structured Systems Design, Yourdon Press, USA, 1988
18. Levine, Harvey A., Project Management Using Microcomputers, McGraw Hill, Osborne, 1986
19. Londeix, B., Cost Estimation for Software Development, Addison-Wesley, Great Britain, 1987
20. Nelson, E., A., 'Management Handbook for the Estimation of Computer Programming Costs', AD-A648750, SDC, Oct, 1966
21. Pressman, R. S., Software Engineering, A Practitioner's Approach, McGraw Hill, Singapore, 1992
22. Parr, F. N., 'An Alternative to Rayleigh Norden Curve Model for Software Development Effort', IEEE Trans. on Software Eng., SE-6(3), May 1980

284 Software Quality Management

23. Pfleeger, S., W., Software Engineering : The Production of Quality Software, MacMillan, USA 1991
24. 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
25. Sevkali S., 'Yöneticilere Yönelik Bilgisayar Kullanımı', Bakis Bilgisayar ve Teknoloji Dergisi, Yil:3, Sayi:9, Nisan-Haziran 1988, Sa:12-14
26. Arifoglu et al., Systems Analysis and Design Reports for Information Centers of SMIDO, (2 Reports), Dept. of Computer Eng., METU, 1992
27. Shooman L., M., Software Engineering, McGraw Hill, Japan, 1983
28. Symons R., C., 'Function Points Analysis, Difficulties and Improvements', IEEE Trans. on Soft. Eng., Vol-Se 14, No.16, Jan 1988, pp. 2-11
29. Tausworthe, R., C., 'Deep Space Network Software Cost Estimation Model', Publication 81-7, Jet Propulsion Library, Pasadena, CA, 1981
30. Walston, C.E., Felix, C.P., 'A Method for Programming Measurement and Estimation', IBM Systems Journal, No.16, 1, 1977, pp. 54-73
31. Wellman, F., Software Costing, Prentice Hall, USA, 1992
32. Wolverton, R., W., 'The Cost of Developing Large Scale Software', IEEE Transactions on Computers, C-23, 6, 1974, pp.615-636
33. Yourdon E., Modern Structured Analysis, Prentice Hall, USA, 1989