Software cost estimation

F J Heemstra

The paper gives an overview of the state of the art of software cost estimation (SCE). The main questions to be answered in the paper are: (1) What are the reasons for overruns of budgets and planned durations? (2) What are the prerequisites for estimating? (3) How can software development effort be estimated? (4) What can software project management expect from SCE models, how accurate are estimations which are made using these kind of models, and what are the pros and cons of cost estimation models?

software, cost estimation, project control, software cost estimation model

SIMPLE QUESTIONS, DIFFICULT ANSWERS

Judging by reports from everyday practice and findings in the literature, software projects regularly get out of hand and invariably the effort expended on development exceeds the estimated effort, resulting in the software being delivered after the planned date. There is no doubt that SCE is a serious problem for software project management. At first glance the questions to be answered are simple: How much time and effort will it cost to develop the software? What are the dominating cost factors? What are the important risk factors? Unfortunately, however, the answers are neither simple nor easy.

The article gives an overview of the field of software cost estimation (SCE). Special attention is paid to the use of SCE models. These models are one of the techniques project management can use to estimate and control the effort and duration of software development. The paper starts with a description of the importance of accurate cost estimates. From this it will be clear that SCE is not easy, and management is confronted with many problems. In the following section some reasons for the problems will be highlighted, the paper going on to explain which prerequisites are necessary for an estimate to be possible. It is important to have knowledge about the product that must be developed, the development process, the development means, the development personnel, and the user organization. Also it is necessary to have available a set of estimation methods and techniques. An overview of the existing techniques for cost estimation is given in the fifth section, and the sixth section describes the principles of cost estimation models with an overview of models available nowadays. The rest of the paper deals with one of these techniques, that is to say parametric models. The penultimate section offers a comparison of SCE models, focusing mainly on the question 'How accurate are estimates made as a result of using models?' Despite the fact that software cost estimation is in its infancy plus the shortcomings of the current SCE models, the use of models has several advantages. The last section deals with the pros and cons and gives a critical evaluation of the state of the art of the use of these models.

Faculty of Public Administration and Public Policy, Twente University, POB 217, Enschede, The Netherlands

OVERSHOOTS OF SOFTWARE DEVELOPMENT COSTS

Estimation of effort and duration of software development has become a topic of growing importance. This is not surprising. It often happens that software is more expensive than estimated and completion is later than planned. Moreover it turns out that much software does not meet the demands of the customer. There are a number of examples of such automation projects. The development costs of the automation of the education funding in The Netherlands proved to be three times as much as expected. Delays and wrong payments are a daily occurrence (*Volkskrant*, 24 June 1987). The development of the software for the purpose of the house-rent subsidies, produced to government order, proved to be twice as much as planned (NRC *Handelsblad*, 28 February 1989). In September 1989 the Dutch media announced as front page news the results of a governmental audit concerning the automation for the police. It proved to be an expensive disaster. The development costs of a computerized identifying system were US$43 million instead of the estimated US$21 million. Furthermore the system did not answer the formulated goals. The findings of a well-known Dutch consultancy organization (Berenschot) were that the costs of the automation of the registration of the Dutch population at the municipal offices were more than twice as much as were estimated (*Volkskrant*, 5 January 1990). A few years ago the estimates of the costs were about US$25 million. New calculations show that there is a deficit of more than US$30 million.

A field study by the Eindhoven University of Technology[1] gives an overview of the present state of the art of the estimation and control of software development projects in 598 Dutch organizations. The most remarkable conclusions are:

- 35% of the participating organizations do not make an estimate
- 50% of the responding organizations record no data on an ongoing project
- 57% do not use cost-accounting
- 80% of the projects executed by the participating organizations have overruns of budgets and duration
- the mean overruns of budgets and duration are 50%

Van Lierop *et al.*[2] measured extensively whether development activities were executed according to plan. They investigated the reasons for the differences between plan and reality, and overall 80 development activities were measured. For all these activities 3203 hours were planned but 3838 hours were used, which means an overshoot of 20% on average of the planned number of hours. The duration of the activities (in days) proved to be 28% longer on average than planned. For all the activities 406 days of duration were planned, while the actual number of days proved to be 526.

In the literature the impression is given, mistakenly, that software development without overshoots of plans and budgets is not possible. This impression is inaccurate, and other measurements confirm this[3]. These show that 6% of all the activities had a shorter duration than planned and 58% were executed according to plan and were ready exactly on time. With regard to the development effort, it appeared that 25% of the activities needed less effort than estimated and 30% needed precisely the estimated effort. The reasons for the differences between plan and reality prove to be very specific for the development situation. In the organization where the measurements were taken the reasons were mainly related to things underestimation of the quantity of work, underestimation of the complexity of the application, and specifications which proved to be unrealistic from a technical point of view. In other organizations, where similar measurements were taken, other reasons were discovered. As a result, other control actions are, of course, necessary. This conclusion fits well with the results of research carried out by Beers[4]. Thirty experienced software developers, project managers, and others, were asked to give the reasons for unsuccessful software projects. The answers can be summarized briefly as 'many minds, many thoughts'. It was not possible to indicate just one reason. A long list of all kinds of reasons were given.

It is alarming that it is so difficult for organizations to control the development of software. This is sufficient reason to emphasize that software development cost estimation and control should take its place as a fully fledged branch within discipline of software development.

WHAT MAKES SOFTWARE COST ESTIMATION SO DIFFICULT?

The main question, when confronting the abovementioned problems, is what it is that makes software cost estimation so difficult. There are many reasons and, without going into detail, some can be listed as follows:

(1) There is a lack of data on completed software projects. This kind of data can support project management in making estimates.
(2) Estimates are often done hurriedly, without an appreciation for the effort required to do a credible job. In addition, too often it is the case that an estimate is needed before clear specifications of the system requirements have been produced. Therefore, a typical situation is that estimators are being pressured to write an estimate too quickly for a system that they do not fully understand.
(3) Clear, complete and reliable specifications are difficult to formulate, especially at the start of a project. Changes, adaptations and additions are more the rule than the exception: as a consequence plans and budgets must be adapted too.
(4) Characteristics of software and software development make estimating difficult. For example, the level of abstraction, complexity, measurability of product and process, innovative aspects, etc.
(5) A great number of factors have an influence on the effort and time to develop software. These factors are called 'cost drivers'. Examples are size and complexity of the software, commitment and participation of the user organization, experience of the development team. In general these cost drivers are difficult to determine in operation.
(6) Rapid changes in information technology (IT) and the methodology of software development are a problem for a stabilization of the estimation process. For example, it is difficult to predict the influence of new workbenches, fourth and fifth generation languages, prototyping strategies, and so on.
(7) An estimator (mostly the project manager) cannot have much experience in developing estimates, especially for large projects. How many 'large' projects can someone manage in, for example, 10 years?
(8) An apparent bias of software developers towards underestimation. An estimator is likely to consider how long a certain portion of the software would take and then to extrapolate this estimate to the rest of the system, ignoring the non-linear aspects of software development, for example co-ordination and management.
(9) The estimator estimates the time it would take to perform the task personally, ignoring the fact that a lot of work will be done by less experienced people, and junior staff with a lower productivity rate.

(10) There exists a serious mis-assumption of a linear relation between the required capacity per unit of time and the available time. This would mean that software developed by 25 people in two years could be accomplished by 50 people in one year. The assumption is seriously wrong. According to Brooks[5] the crucial corollary is: 'Adding people to a late project only makes it later'.

(11) The estimator tends to reduce the estimates to some degree, in order to make the bid more acceptable.

PREREQUISITES FOR SOFTWARE COST ESTIMATION

There are many ways to get to grips with the SCE problems. From an organizational perspective there are numerous ways to improve software project management: allocation of responsibilities; decision-making; organizing project work; monitoring and auditing of development tasks. Also software cost estimation can be looked at from a sociological and psychological point of view. This refers, for example, to commitment, organizing group cohesion, style of leadership, and so on. The technical side of the job is also an important issue to take into consideration. For example, the availability of good equipment such as design, programming, test and documentation tools, hardware facilities, etc.

There are many factors that have an influence on the effort and duration of software development. Several prerequisites must be fulfilled to address the problems listed above and to guarantee a sound basis for predicting effort, duration and the capacity to develop the software. These prerequisites are:

Insight in the characteristics of:

• the product (software) that has to be developed	WHAT
• the production means	WITH WHAT
• the production personnel	WHO
• the organization of the production	HOW
• the user/user organization	FOR WHOM

Availability of:

- Techniques and tools for software cost estimation.

In this section the attention will be focused on the WHAT, WITH WHAT, WHO, HOW and FOR WHOM factors, referred to as cost drivers in the literature. In the next section, SCE techniques and tools will be discussed.

There are many cost drivers. A study by Noth and Kretzschmar[6] found that more than 1200 different drivers were mentioned. Although there was considerable overlap in meaning, it is impossible to take them all into consideration during SCE. It is important for an organization to consider what are the most dominant cost factors. Within the context of this paper it is impossible to give an extended overview of the overwhelming number of drivers, so concentration will be on:

- a way of structuring the cost drivers
- listing the drivers which are commonly regarded as important
- some general considerations

Table 1 presents a structure of cost drivers in five categories. For each category the most important drivers are listed. From the literature and practice it is known that it is not easy to handle the cost drivers. When making an estimate one has to know which cost drivers are the most important in the specific situation, what the values are of the drivers, and what the influences are on effort and duration. In answering these questions it is important to pay attention to several issues:

Definition There is a lack of clear and accepted definitions for drivers, such as size, quality, complexity, experience, etc.

Quantification The majority of the cost drivers are hard to quantify. Often one has to use measures such as many, moderate, few, etc.

Table 1. A structure of important cost drivers[7]

WHAT (product)	WITH WHAT (means)	WHO (personnel)	HOW (project)	FOR WHOM (user)
Size of the software	Computer constraints —execution time —response time —memory capacity	Quality of personnel	Requirements project duration —stretch out —compression	Participation
Required quality	User of tools	Experience of personnel	Basis for project control —matrix org. —project org. —prototyping —incremental —linear devel. —software devel.	Number of users
Requirements volatility	Use of modern programming techniques —information hiding —chief prog. team —structured program —top-down design	Quality management		Stability of user organization, procedures, way of working
Software complexity		Availability for project		Experience of user with automation, level of education in automation
Level of reuse				
Amount of documentation				
Type of application				

Objectivity Subjectivity is a potential risk factor. What may be complex for developer A is not complex for developer B.

Correlation It is difficult to consider one driver by itself. A change in the value of driver A may have consequences in the values of several other cost drivers. This is a difficulty from the viewpoint of measurability.

Relation between driver and effort For estimation it is important to predict the relation between, for example, software size and the required effort, a specified quality level and required effort, etc. From the literature we know that there is little clarity about these relations.

Calibration It is impossible to talk about 'the most important' cost drivers in isolation. It differs from situation to situation.

Effectivity and efficiency There is conflict between effectivity and efficiency. From an effectivity perspective it is worthwhile to pay a lot of attention to, for example, user participation. For the efficiency of a project it is justifiable to avoid user involvement.

Human factors Almost all research agrees on the dominating influence of cost drivers, such as experience and quality of the personnel. This means that investment in 'good' developers is important.

Reuse In many studies reuse is regarded as (one of) the most important factors to increase productivity[8–10].

SOFTWARE COST ESTIMATION: TECHNIQUES AND TOOLS

In the literature you can find a great number of techniques for estimating software development costs. Most of them are a combination of the following primary techniques[11]:

(1) Estimates made by an expert.
(2) Estimates based on reasoning by analogy.
(3) Estimates based on Price-to-Win.
(4) Estimates based on available capacity.
(5) Estimates based on the use of parametric models.

Furthermore two main approaches can be distinguished:

(1) Top-down
In the top-down approach the estimation of the overall project is derived from the global characteristics of the product. The total estimated cost is then split up among the various components.
(2) Bottom-up
In the bottom-up approach the cost of each individual component is estimated by the person who will be responsible for developing the component. The individual estimated costs are summed to get the overall cost estimate of the project.

The reliability of estimates based on expert judgement (1) depends a great deal to the degree in which a new project conforms with the experience and the ability of the expert to remember facts of historical projects. Mostly the estimates are qualitative and not objective. An important problem in using this method is that it is difficult for someone else to reproduce and use the knowledge and experience of an expert. This can lead to misleading situations where the rules of thumb of an expert are becoming general rules and used in inapplicable situations. Despite the disadvantages, this technique is usually used in situations where a first indication of effort and time is needed, especially in the first phases of software development in which the specifications of the product are vague and continually adapted.

The foundation of a cost estimation technique based on reasoning by analogy (2) is an analysed database of similar historical projects or similar project parts or modules. To find a similarity between a new project and one or more completed projects it is necessary to collect and record data and characteristics of old projects.

The Price-to-Win (3) technique can hardly be called an SCE technique. Primarily commercial motives play an important part in using this approach. It is remarkable that the estimates of organizations which use Price-to-Win are no less accurate than organizations which use other methods[7].

The basis of the estimation method which regards SCE as a capacity (4) problem is the availability of means, especially of personnel. An example is: 'Regarding our capacity planning, three men are available for the new project over the next four months. So the planned effort will be 12 man months'. If the specifications of the software are not clear, this method can be successful. An unfavourable side-effect is that in situations of overestimation the planned effort will be used completely. This effect is based on Parkinson's law that 'Work expands to fill the available volume'.

In parametric models (5) the development time and effort is estimated as a function of a number of variables. These variables represent the most important cost drivers. The nucleus of an estimation model is a number of algorithms and parameters. The values of the parameters and the kind of algorithms are, to a significant extent, based on the contents of a database of completed projects. In the next section a more comprehensive explanation of estimation models is given.

As mentioned earlier only 65% of the organizations which participated on the field study estimate a software project. Table 2 shows the frequency of use of the different techniques. The figures show that most organizations make use of data from past projects in some way. Obviously this works on an informal basis, because only 50% of the participating organizations record data from completed projects. Estimates based on expert judgement and the capacity method prove to be quite popular despite the disadvantages of these methods.

Table 2. Use of cost estimation techniques (an organization can use more than one technique)

	Use (%)
Expert judgement	25.5
Analogy method	60.8
Price-to-Win	8.9
Capacity problem	20.8
Parametric models	13.7

The next sections of this paper focus on the use of SCE models. There was a rapid growth of models in the 1970s. In the 1980s and the 1990s, however, few new models have been developed despite the increasing importance of controlling and estimating software development. Most of the 1970 models are of no interest to present industrial practitioners. There is a tendency towards automated versions (tools) of (combinations or refinements) existing models. An important question is whether this kind of model can solve all of the problems discussed above.

SOFTWARE COST ESTIMATION MODELS

In this section, one estimation technique, namely SCE models, will be discussed and the principles of SCE models described, making a distinction between sizing and productivity models. The characteristics of some well-known models will also be given.

The principles of SCE models

Most models found nowadays are two-stage models[7]. The first stage is a sizer and the second stage provides a productivity adjustment factor.

In the first stage an estimate regarding the size of the product to be developed is obtained. In practice several sizing techniques are used. The most well-known sizers nowadays are function points[12] and lines of code[11]. But other sizing techniques like 'software science'[13] and DeMarco's Bang method[14,15], have been defined. The result of a sizing model is the size/volume of the software to be developed, expressed as the number of lines of source code, number of statements, or the number of functions points.

In the second stage it is estimated how much time and effort it will cost to develop the software of the estimated size. First, the estimate of the size is converted into an estimate in nominal man-months of effort. As this nominal effort takes no advantage of knowledge concerning the specific characteristics of the software-product, the way the software-product will be developed and the production means, a number of cost influencing factors (cost drivers) are added to the model. The effect of these cost drivers must be estimated. This effect is often called a productivity adjustment factor. Application of this correction factor to the nominal estimation of effort provides a more realistic estimate.

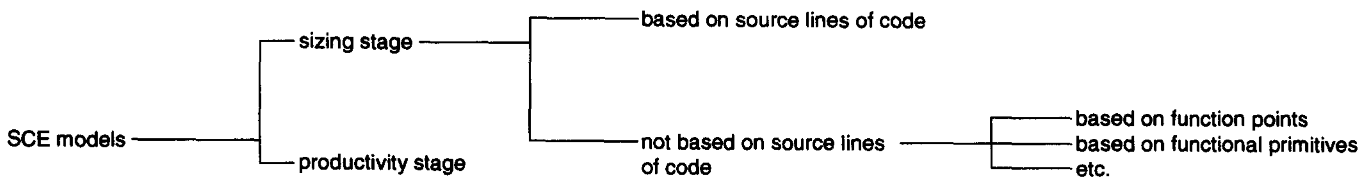
Some models, like FPA[16], are focused more on the sizing stage. Others, like the well-known COCOMO model[11] on the productivity stage and some tools, such as Before You Leap[17] combine two models to cover both stages. Figure 1 shows the two stages in SCE models.

Figure 2 shows the sizing and the productivity stages in the context of general cost estimation. In Figure 2 five components of the general cost estimation structure are shown. Besides the sizing and productivity components, a phase distribution and sensitivity/risk analysis component are distinguished. In the phase distribution component the total effort and duration is split up over the phases and activities of a project. This division has to be based on empirical data of past projects. The sensitivity and risk analysis phase supports project management — especially at the start of a project when the uncertainty is great — in determining the risk factors of a project and the sensitivity of the estimates to the cost drivers settings. Again data on past projects provide an important input for this component. Before using a model for the first time validation is necessary, and it may also be necessary to calibrate the model. Mostly the environment in which the SCE model has been developed and the database of completed projects on which the model is based will differ from the project characteristics of the environment(s) in which the model is to be used. To make validation and calibration possible, data on historical projects have to be available in an organization. As already mentioned, this information is often lacking.

Most of the tools implementing SCE models do not support project management in all of these steps. The seven steps are:

(1) Creation of database of completed projects.
(2) Size estimation.
(3) Productivity estimation.
(4) Phase distribution.
(5) Sensitivity and risk analysis.
(6) Validation.
(7) Calibration.

Calibration and risk and sensitivity analysis are especially lacking.



SCE models
sizing stage
productivity stage
based on source lines of code
not based on source lines of code
based on function points
based on functional primitives
etc.

Figure 1. Structuring of SCE models

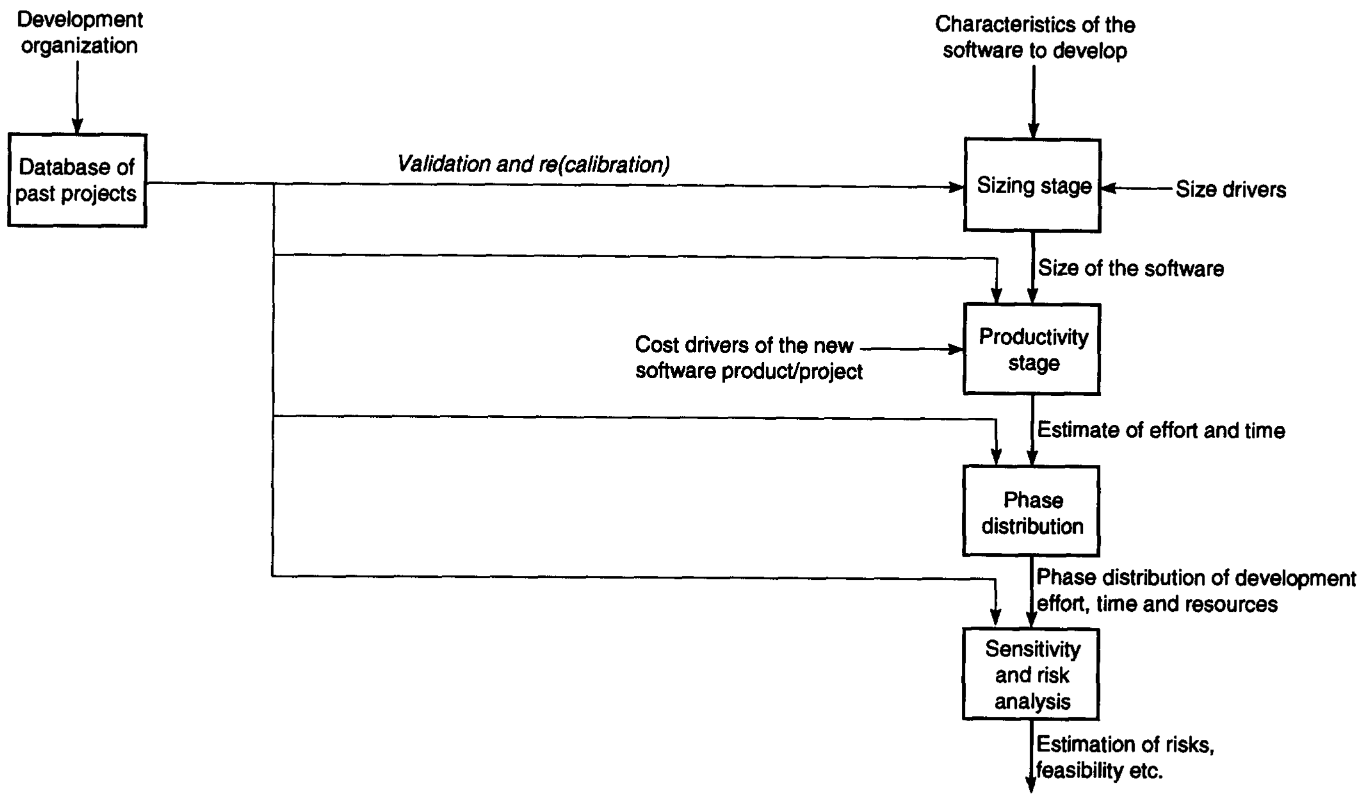

Development organization
Database of past projects
Validation and re(calibration)
Characteristics of the software to develop
Sizing stage
Size drivers
Size of the software
Cost drivers of the new software product/project
Productivity stage
Estimate of effort and time
Phase distribution
Phase distribution of development effort, time and resources
Sensitivity and risk analysis
Estimation of risks, feasibility etc.

Figure 2. General cost estimation structure

An overview of SCE models

In the past 10 years a number of SCE models have been developed. This section does not give an exhaustive treatment of all the models: the overview is limited to one example of a sizing model, one productivity model, some models which are relevant from an historical point of view, well documented and within the experience of the author, and some models which introduce new ideas.

The COnstructive COst MOdel (COCOMO)

COCOMO[11,18] is the best documented and most transparent model currently available. The main focus in COCOMO is upon estimating the influence of 15 cost drivers on the development effort. Before this can be done, an estimate of the software size must be available. COCOMO does not support the sizing estimation stage: it only gives several equations based on 63 completed projects at TRW. The equations represent the relations between size and effort and between effort and development time. The equations are shown in Table 3. A distinction is made between three development modes: the organic mode (stable development environment, less innovative, relatively small size); the embedded mode (developing within tight constraints, innovative, complex, high volatility of requirements); and the semi-detached mode (between organic and embedded mode).

The nominal effort is adjusted by the influence of 15 cost drivers. In Table 4 the 15 COCOMO cost drivers are listed with the adjustment for each driver value. For example: where the required reliability of the software is determined to be very high, the nominal effort has to be multiplied by 1.40. Furthermore COCOMO provides tables to apportion the adjusted estimated effort and development over the project phases and, in the detailed version of the model, to refine the adjustment for each phase. For example: the quality of the programmer has less influence in the feasibility phase than in the design phase. Thus phase dependent adjustment factors are used in the detailed model.

Function point analysis (FPA)

FPA has been developed by Albrecht[16] of IBM, and made widely available through the user groups *Guide and Share*. Albrecht was looking for a method to measure productivity in software development. For that purpose he developed FPA as an alternative measure to the number of lines of code. The method is programming language or fourth generation tool independent. The method has been refined several times by Rudolph[19,20], Albrecht and Gaffney[12], and Symons[21,22]. The principle of FPA is simple and is based on the number of 'functions' the software has to fulfil. These functions are

Table 3. The relation between the nominal effort and size and between development time and effort. KDSI = number of delivered source instructions/1000

Development mode	Man-month (nominal)	Development time (nominal)
Organic	$3.2*\text{KDSI}^{1.05}$	$2.5*\text{MM (nom)}^{0.38}$
Semi-detached	$3.0*\text{KDSI}^{1.12}$	$2.5*\text{MM (nom)}^{0.35}$
Embedded	$2.8*\text{KDSI}^{1.20}$	$2.5*\text{MM (nom)}^{0.32}$

Table 4. The COCOMO cost drivers and their influence on the nominal effort

	Value of the cost drivers					
Cost drivers	Very low	Low	Average	High	Very high	Extra high
Required reliability	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Complexity software	0.70	0.85	1.00	1.15	1.30	1.65
Constraints execution time			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Hardware volatility		0.87	1.00	1.15	1.30	
Response time constraints		0.87	1.00	1.07	1.15	
Quality analysts	1.46	1.19	1.00	0.86	0.71	
Experience with application	1.29	1.13	1.00	0.91	0.82	
Quality programmers	1.42	1.17	1.00	0.86	0.70	
Hardware experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Use modern programming techniques	1.24	1.10	1.00	0.91	0.82	
Use software tools	1.24	1.10	1.00	0.91	0.83	
Project duration constraints	1.23	1.08	1.00	1.04	1.10	

related to the types of data the software uses and generates. Within FPA the software is characterized by the five functions:

- the external input type
- the external output type
- the external inquiry type
- the logical internal file type
- the external interface file type

For each of these five types the number of simple, average and complex occurrences that are expected in the software is estimated. By weighting each number with an appropriate weight a number is obtained, the unadjusted number of function points. This indication for nominal size is then adjusted, using 14 technical characteristics. Figure 3 gives an overview of function point analysis.

PRICE-S

The PRICE-S model (Programming Review of Information Costing and Evaluation — Software) is developed and supported by RCA PRICE Systems. An important disadvantage with regard to COCOMO and FPA is that the underlying concepts and ideas are not publicly defined and the users are presented with the model as a black box. The user of PRICE sends the input to a time-sharing computer in the USA, UK, or France and gets back his estimates immediately. Despite this disadvantage and the high rental price, there are many users, especially in America. There is, however, an important motivation for American companies to use the model. The US Department of Defense demands a PRICE estimate for all quotations for a software project. PRICE has separate sizer and productivity function.

The PUTNAM model

This SCE model was developed by Putnam in 1974[23]. He based his model on the work of Norden[34]. For many projects at IBM, Norden plotted frequency distributions, in which he showed how many people were allocated to the development and maintenance of a software product during the life-cycle. The curves he made fitted very well with the Rayleigh curves. His findings were merely empirical. He found no explanations for the shape of the effort curve. On the assumptions of Norden, Putnam formulated his model. There is not enough space in this paper to explain the principles of the model and the reader is referred to Putnam[23,24], Putnam and Fitzsimmons[25] and Londeix[26].

Before You Leap (BYL)

BYL is a commercial package based on a link-up between FPA and COCOMO[17]. BYL starts with a calculation of the amount of net function points. This amount is then translated into source lines of code, taking in account the language used. For Cobol, for instance, one function point is equal to 105 SLOC, for LISP 64, etc. This estimate of the size in SLOC is precisely the necessary input for COCOMO and the COCOMO part of BYL, taking into account the influence on effort of the 15 COCOMO cost drivers, calculates the estimates of costs and time-scale.

Estimacs

Estimacs has been developed by H. Rubin[27-29] and Computer Associates[30], and is available as a software package. The model consists of nine modules: a function point module; a risk module; an effort module (to estimate development and maintenance effort), etc. The most important and extensive module is Effort. The user has to answer 25 input questions. These questions are partly related to the complexity of the user-organization and partly to the complexity and size of the software to be developed. The way Estimacs translates the input to an estimation of effort is not clear. Like many other models, Estimacs is a 'closed model'.

Function count		←—Max range: Factor * 2—→				
		Level of information processing function				
Type ID	Description	Simple	Average	Complex	Total	
IT	External input	--*3 = --	--*4 = ---	--*6 = ---	-----	↑
OT	External output	--*4 = --	--*5 = ---	--*7 = ---	-----	Maximum
FT	Logical internal file	--*7 = --	--*10 = --	--*15 = --	-----	range
EI	External interface file	--*5 = --	--*7 = ---	--*10 = --	-----	factor 2.5
QT	External inquiry	--*3 = --	--*4 = ---	--*6 = ---	-----	↓
FC		Total unadjusted function points				

General information processing characteristics

	Characteristics	DI		Characteristics	DI
C1	Data communications	---	C8	On-line update	---
C2	Distributed functions	---	C9	Complex processing	---
C3	Performance	---	C10	Re-usability	---
C4	Heavily used configuration	---	C11	Installation ease	---
C5	Transaction rate	---	C12	Operational ease	---
C6	On-line data entry	---	C13	Multiple sites	---
C7	End-user efficiency	---	C14	Facilitate change	---
PC				Total degree of influence	---

DI Values

Not present or no influence	= 0	Average influence	= 3
Insignificant influence	= 1	Significant influence	= 4
Moderate influence	= 2	Strong influence, throughout	= 5

FC	(Function count)	≡	Total unadjusted function points
PC	(Process complexity)	≡	Total degree of influence
PCA	(Process complexity adjustment)	=	0.65 + 0.01 * PC
FP	(Function point measure)	=	FC * PCA

Figure 3. Overview of function point analysis

SPQR-20

SPQR stands for Software Productivity, Quality and Reliability. The model has been developed by C. Jones[31]. SPQR claims to be applicable for all kinds of software projects as well as an estimate of duration, costs and effort to develop software; the model also gives an estimate of maintenance costs. SPQR uses FPA to size the volume of a program. The model is based on an extensive database of past projects. There are four versions of model, SPQR 10, 20, 50 and 100 (the numbers stand for the number of questions the model user has to answer and gives an indication of the degree of refinement of the versions). SPQR-20 is the only commercially available version at the moment, not marketed by C. Jones any more but overtaken by his Checkmark product.

BIS-Estimator

BIS-Estimator is completely different from the previously described models. According to the documentation[32] the model claims to be a 'knowledge-based tool'. This cannot be fully confirmed, because the principles of the model are secret for the most part. The model starts with a 'soft' estimate. This is a rough estimate of duration and effort based on (far too few) input questions. Next a 'hard' estimate is made for each phase. Based on the estimates by phase, by means of extrapolation, an estimate of the complete project is made. The 'hard' estimate has to be made at the start of and/or during each phase. The model has facilities to base the estimate upon a comparison with a number of projects, selected by the model user. A positive feature of the model is the evolutionary approach. This means that the estimation process changes during software development. As a result of the kind of questions, data and considerations, an estimate is based on the model changes for each phase.

Several models and computerized versions (tools) are available, but just a few of these have been described briefly above. Without going into detail, Table 5 gives a more extensive list of models and tools. The reader is referred to publications in the literature for a more comprehensive description of each. The models in the list are in chronological order (year of publication). The first 11 are ancient models and of no current interest to practitioners.

COMPARISON OF SCE MODELS

During the past few years several empirical studies have been carried out to validate the various SCE models. Validation is important but difficult to do, because of the demand to capture large amounts of data about completed software projects. As mentioned before, data collection is not common in the software community. It is labour and time-intensive and requires an attitude not only focused on the constructive part but also on the analytical part of software engineering. Furthermore data collection, usable for validating SCE, is limited to a relative small number of software development organizations. Only a few organizations realize large software projects each year. Nevertheless, a number of validation research investigations have been carried out. In this section some of them will be discussed.

The models discussed earlier differ considerably. Experiments show that estimates made by the different models for the same project vary strongly. Furthermore the estimates differ very much from the real development cost and duration. To give an opinion upon the quality of SCE models, it must be known what kind of demands have to be made upon these models. In Table 6 an overview of these demands/requirements is presented. These requirements are a part of an evaluation method for SCE models. This method has been developed by Heemstra, Kusters and van Genuchten[1] and used to

Table 5. SCE models and tools with references

Model	Source
SDC	Nelson, E A *Management handbook for the estimation of computer programming costs*, AD-A648750, Systems Development Corporation (1966)
TRW Wolverton	Wolverton, R W 'The cost of development large-scale software' *IEEE Trans. on computers*, Vol c-23, No 6 (June 1974)
TELECOTE	Frederic, B C *A professional model for estimating computer program development costs*. Telecote Research Inc. (1974)
BOEING	Black, R K D, Curnow, R P, Katz, R and Gray, M D 'BCS software production data' *Final technical report, RADC-TR-77-116*, Boeing Computer Services Inc. (March 1977)
IBM/FSD	Walston, C E and Felix, C P 'A method of programming measurement and estimating' *IBM System J.* Vol 16 (1977)
DOTY	Herd, J R, Postak, J N, Russell, W E and Stewart, K R 'Software cost estimation — study results. *Final technical report, RA-DC-TR-77-220*, Vol 1, DOTY Associates, Inc., Rockville, MD (1977)
ESD1	Duquette, J A and Bourbon, G A 'ESD, A computerized model for estimating software life cycle costs' *FSD-TR-235* Vol 1 (April 1978)
SLIM	Putnam, L H 'A general empirical solution to the macro software sizing and estimating problem' *IEEE Trans. Soft. Eng.* SE-4, 4 (1978)
Surbock	Surbock, E K *Management software development* Projekten Berlin (1978) (In German)
GRC	Carriere, W M and Thibodeau, R 'Development of a logistic software cost estimating technique for foreign military sales' *GRC Report CR-3-839* (1979)
Grumman	Sandler, G and Bachowitz, B 'Software cost models — Grumman experience' *IEEE, quantitative software model conference* (1979)
PRICE-S	Freiman, F R and Park, R E 'The Price software cost model: RCA government systems division' *IEEE* (1979)
FPA	Albrecht, A J 'Measuring application development productivity' *Proc. of Joint SHARE/GUIDE/IBM application development symp.* (October 1979)
SLICE	Kustanowitz, A L 'System life cycle estimation (SLICE): a new approach to estimating resources for application program development' *IEEE first international computer software and application conference, Chicago* (1980)
FAST	Freiman, F R 'The FAST methodology' *J. of parametrics*, Vol 1 No 2 (1981)
Baily/Basili	Bailey, J W and Basili V R 'A meta-model for software development resource expenditures' *Proc. 5th Int. Conf. Soft. Engin., IEEE* (1981)
COCOMO	Boehm, B W *Software engineering economics* Prentice-Hall (1981)
SOFTCOST	Tausworthe, R C 'Deep space network software cost estimation model' Publication 81-7, *Jet Propulsion Laboratory*, Pasadena, CA (1981)
BANG	DeMarco, T *Controlling software projects: management, measurement and estimation* Yourdon Press, New York (1982)
JS 3/System-4/Seer	Jensen, R W 'An improved macrolevel software development resource estimation model' *Proc. 5th ISPA Conf.* St Louis MO (1983)
COPMO	Thebaut, S M and Shen, V Y 'An analytic resource model for large-scale software development' *Inf. Proc. Management*, Vol 20 No 1-2 (1984)
GECOMO	Gecomo 'Software tools for professionals' *GEC Software Documentation*, G & C Company, London (1985)
ESTIMACS	Computer Associates. CA-Estimacs *User Guide*, Release 5.0 (July 1986)
BYL	Before You Leap. *User's Guide*, Gordon Group (1986)
SPQR/Checkmark	Jones, C *Programming productivity* McGraw-Hill (1986)
Jeffery	Jeffery, D R 'A software development productivity model for MIS environments' *J. of Systems and Software* 7 (1987)
ESTIMATE/1	Estimate/1. Documentative Method/1: Automated Project Estimating Aid. Arthur Anderson (1987)
BIS	BIS/Estimator. User Manual, version 4.4, BIS Applied System Ltd (1987)
SECOMO	Goethert, W B 'SECOMO' in Boehm, B W *Documentation of the seminar: software cost estimation using COCOMO and ADA COCOMO*, SAL, London. 1988' ITT Research Institute, Data & Analysis Center for Software.

Table 6. Requirements for SCE models

Model requirements	Application requirements	Implementation requirements
Linked to software control method	Possibilities for calibration	User-friendliness of the tool
Applicability at the start of a project	Accuracy of the estimations	Possibilities for sensitivity analyses
Fit with the data that is available during development		Possibilities for risk analysis
Possible to adjust estimate due to changing objectives		Open model, is it possible to see how the results were obtained
Definition of domain model is suitable for		Clarity of input definition
		Completeness and detail of output

evaluate the eight models described above. The results of that evaluation are presented in Table 7 and described in more detail in Heemstra[7]. From the table it can be seen that there are only few plusses. The conclusion is that the quality of the models is poor and much improvement is necessary. The accuracy of the estimations were evaluated by several tests. The way the tests were executed and the results obtained will be described. The objectives of the tests were:

- to determine the accuracy of the estimate using SCE models in a semi-realistic situation
- to determine whether these models will be accepted by project management

After a severe selection procedure only two SCE models remained. These were the BYL and Estimacs models. During the tests 14 experienced project leaders were asked to make a number of estimates for a project that had actually been carried out. The project was described as if it was at the start of the project. The project leaders had to make three estimates. The first estimate of effort and duration (the 'manual' estimate) was made on the basis of the project leaders' knowledge and experience. Next, two estimates were made using the models selected. In conclusion, a final estimate was made on the basis of the project leaders' knowledge and experience together with the model estimates. Each estimate was evaluated directly using a questionnaire, and the tests ended with a discussion session. The results are presented in Table 8.

The real effort and duration were eight man-months and six months. The main conclusions of the experiment were that on the basis of the differences found between the estimates and reality, it has not been shown that the selected models can be used for a reliable estimation tool at an early stage of software development. All in all, the project leaders were not wildly enthusiastic about these tools, but they were, nevertheless, felt to be acceptable as a check-list and as a means of communication. It should be mentioned that the selected project is small. Most models are calibrated on data from medium/large projects.

Kemerer[33] shows that estimates of different models can differ considerably. For each model he investigated the difference between actual and estimated number of man-months. He used COCOMO, Estimacs, FPA and Putnam's model to estimate the required effort of 15 already realized projects. From Table 9 it can be seen that for both COCOMO and Putnam's model there were sharp overestimations. FPA and Estimacs gave distinctly better results with overshoots of 100% and 85%, respectively. A similar study was carried out by Rubin[29]. A project description was sent to Jensen (Jensen's model), Greene (Putnam's model SLIM) and Rook (GECOMO) and to himself (Rubin's model Estimacs).

Table 7. Evaluation of models

	Models							
Requirements	COCOMO	PRICE	PUTNAM	FPA	BYL	ESTIMACS	SPQR	BIS
Model requirements								
Linked to software control method	− −	− −	− −	− −	− −	+ +	− −	− −
Applicable at an early stage	− −	− −	− −	+	+	+ +	+	−
Using available data	+	− −	− −	− −	− −	− −	− −	+ +
Adjustment to objectives	+	+	+	− −	− −	+	+ +	− −
Definition of scope/domain	+	−	−	+ +	−	−	−	+ +
Application requirements								
Calibration	−	− −	− −	−	+	+	−	−
Accuracy	nt	nt	nt	nt	t	t	nt	nt
Implementation requirements								
User friendliness	+ +	−	+	+	+ +	+	+	+
Sensitivity analysis	− −	+	− −	− −	+ +	+ +	−	−
Risk analysis	− −	− −	− −	− −	− −	+ +	+	− −
Open model/traceability	+ +	− −	+ +	+ +	+ +	−	−	+
Definition input	+ +	−	+ +	−	+	+	+	+
Completeness and detail output	+	+ +	−	−	+ +	+ +	+ +	+ +

+ + = satisfies the requirement; + = sufficient; − = insufficient; − − = the model does not satisfy the requirement; nt = the model was not tested on accuracy; t = the models were tested

Table 8. Some results of the tests. Duration is given in months, effort in man-months

Variable	μ	σ
Effort		
Manual estimate	28.4	18.3
BYL estimate	27.7	14.0
Estimacs estimate	48.5	13.9
Final estimate	27.7	12.8
Duration		
Manual estimate	11.2	3.7
BYL estimate	8.5	2.4
Final estimate	12.1	3.4

Table 9. Estimates of the actual and estimated number of man-months using four different models

	Averages for all projects		
Models	Actual number of MM	Estimated number of MM	(Estimated divided by actual) * 100%
GECOMO	219.25	1291.75	607.85
Putnam	219.25	2060.17	771.87
FPA	260.30	533.23	167.29
Estimacs	287.97	354.77	85.48

The main purpose was to compare and contrast the different sort of information required by the four models. Also a comparison was made between the estimates obtained using the models, that is to say the number of man-months and the duration for the development of the selected project. From Table 10 it can be seen that the estimates vary significantly. Also Rubin's explanation is that the models are based on different databases of completed projects and have not been calibrated and the four participants made different assumptions in choosing the settings of the cost drivers.

THE IMPORTANCE OF SCE MODELS

The field study, mentioned earlier in the paper, shows that SCE models are currently not generally accepted in organizations surveyed. Only 51 of the 364 organizations that estimate software development use models. An analysis showed that these 51 model-users make no better estimates than the non-model-users. These results are disappointing at first glance. It does not mean, however, that it makes no sense to spend further research effort on models. All the investigations mentioned before agree that the poor quality is primarily due to using the models wrongly. For example: use of models requires organizational bounded data of past projects. Most of the time models are used without calibration. If models cannot be adapted the result will be less accurate estimates. The majority of the models do not support calibration.

It is worth while to promote the development of better estimation tools, despite the shortcomings of the existing models. In this section some arguments are put forward that underline the necessity to invest more effort and time in the development of SCE models.

In making an estimate, especially at an early stage of development, a lot of uncertainty and fuzziness exists. It is not known which cost drivers play a part in the estimation and what the influence of the cost drivers will be. There are many participants involved in the project (project manager, customer, developer, user, etc.). Often they all have their own hidden agendas and goals conflicting with each other (minimalization of the costs, maximalization of the quality, minimalization of the duration, optimal use of employees, etc.). For project management it is difficult to predict the progress of a project in such fuzzy situations. To make point estimations like 'duration will be 321 man-months of which 110 for analysis, 70 for design, etc.', will be of less importance. Such exact figures do not fit in with the nature of the problem. Project management will be more interested in a number of scenarios from which alternatives can be chosen and in the sensitiveness of an estimation to specific cost drivers. For example: what will be the result on the duration of the addition of two more analysts to the project: what will be the influence on effort if the available development time will be decreased sharply; what will be the result on effort and duration if the complexity of the software to be developed has been estimated too high or too low, etc. An approach of the estimation problem like this gives project management more insight and feeling for alternative solutions. Furthermore this approach offers a proper basis for project control. If an estimate proves to be sensible for changes of a specific cost driver, this provides a warning for project management to pay full attention to this cost driver during development.

Often project management will be confronted with little tolerance in defined duration, price and quality. In such cases project management wants support in choosing the values of the decision variables. What are the available possible choices to meet the given objectives. Which personnel in combination with which tools and by means of which kind of project organization are suitable as possible solutions. The conclusion is that there is no need for a rigid 'calculation tool'. This does not fit with the characteristics of the estimation problem, namely uncertainty, fuzziness, little structuring, and unclear and incomplete specifications.

An important prerequisite for successful estimation is the development, acceptance and use of a uniform set of

Table 10. Comparison of SCE models by Rubin[29]

		Effort	Duration
Mode	Jensen	940 MM	31 m
	Putnam	200 MM	17 m
	GECOMO	363 MM	23 m
	Estimacs	17 100 hrs	16 m

MM = man-months; m = months

definitions and standards. This results in agreements such as:

- How many times an estimate is made for a project. For example: five times for each project that costs more than 12 man-months.
- In what phases during execution an estimate is made. For example: during the feasibility study, during the specification phase and after finishing the design.
- Which employees are involved in the estimation process. For example: project management, customers, developers.
- What will be estimated. For example: all development activities with regard to the phases feasibility, specification, design, etc. or all activities including training, documentation, etc.
- The output of an estimate. For example: costs in dollars, effort in man-months, duration in months.
- The factors which can be regarded as the most important cost drivers and have to be recorded. For example: size, reliability, type of application, quality of personnel, etc.
- A set of definitions. For example: volume will be expressed in function points, documentation contains of . . . , high complexity means . . . , etc.

The result will be a comprehensive list of standardized agreements. It is important that these are really applied in the subsequent project. An SCE model that meets requirements such as a set of clear definitions, measurable and relevant cost drivers, flexibility with regards to other control methods, etc. will result in a more structural approach to software cost estimation and control.

CONCLUSIONS AND RECOMMENDATIONS

In this final section some concrete guidelines for controlling and estimating software development will be offered. Most of these guidelines have been discussed at different levels of detail in the previous sections.

Determine the level of uncertainty

High uncertainty needs another approach of cost estimation and control than does low uncertainty. High uncertainty corresponds with risk analysis, estimating and margins, exploration oriented problem-solving, expert-oriented estimating techniques, etc. Low uncertainty corresponds with cost estimation models (calculation tools), experiences from past projects, realization oriented problem-solving, the estimate is regarded as a norm, etc.

Cost estimation and data collection

Collection of data of completed projects is necessary for successful cost estimation. Cost models, estimation by analogy and experts require such data. It is no solution to use data collected from other organizations. The relevant data are different for each organization.

Use more than one estimation technique

A lot of research shows that the quality of the current estimation techniques is poor. The lack of accurate and reliable estimation techniques combined with the financial, technical, organizational and social risks of software projects, require a frequent estimation during the development of an application and the use of more than one estimation technique. More and different techniques are required, especially at the milestones of the development phases. The level of knowledge of the software whose cost we are trying to estimate is growing during a project. A possibility is to use another model during a project, because more information and more *accurate* information is available; a cascade of techniques — for example Wide Band Delphi, Estimacs, DeMarco, COCOMO — is a possible solution.

Cost estimation needs commitment

Software development has to be done by highly qualified professionals. For such people some characteristics are relevant, such as:

- individuality in work performance is important
- a good professional result of their work is important
- professionals want to be consulted in decisions, work planning, the desired result, etc.
- professionals do not want to be disturbed by management during the execution of their work

It is not wise to confront professional developers with a plan and estimate without any consultation. A hierarchical leadership is not suitable. In consulting the developers not only their expertise is used but also their involvement in the estimation process is increased. This results in a higher commitment than is necessary for the success of a project.

Cost estimation: a management problem

Software cost estimation is often wrongly regarded as a technical problem that can be solved with calculation models, a set of metrics and procedures. However, the opposite is true. The 'human aspects' are much more important. The quality, experience and composition of the project team, the degree in which the project leader can motivate, kindle enthusiasm and commit his developers, has more influence on delivering the software in time and within budget than the use of rigid calculations.

REFERENCES

1 **Heemstra, F J, Kusters, R and van Genuchten, M** 'Selections of software cost estimation models' *Report TUE/BDK* University of Technology Eindhoven (1989)

2 **Lierop van, F L G, Volkers, R S A, Genuchten, M van and Heemstra, F J** 'Has someone seen the software?' *Informatie* Vol 33 No 3 (1991) (In Dutch)

3 **Genuchten, van M I J M** 'Towards a software factory' *PhD Thesis*, University of Technology Eindhoven (1991)

4 **Beers** 'Problems, planning and knowledge, a study of the processes behind success and failure of an automation project' *PhD Series in general management, No 1* Faculty Industrial Engineering/Rotterdam School of Management, Erasmus University Rotterdam (1991) (In Dutch)

5 **Brooks, F B** *The mythical manmonth. Essays on software engineering* Addison-Wesley (1975)

6 **Noth, T and Kretzschmar, M** *Estimation of software development projects* Springer-Verlag (1984) (In German)

7 **Heemstra, F J** *How expensive is software? Estimation and control of software-development* Kluwer (1989) (In Dutch)

8 **Druffel, L E** 'Strategies for a DoD Software initiative' *CSS DUSD(RAT)* Washington, DC (1982)

9 **Conte, S D, Dunsmore, H F and Shen, V Y** *Software engineering metrics and models* Benjamin Cummins (1986)

10 **Reifer, D J** 'The economics of software reuse' *Proc. 14th Annual ISPA Conf.*, New Orleans (May 1991)

11 **Boehm, B W** *Software engineering economics* Prentice-Hall (1981)

12 **Albrecht, A J and Gaffney, J E** 'Software function, source lines of code, and development effort prediction: a software science validation' *IEEE Trans. Soft. Eng.* Vol SE-9 No 6 (1983)

13 **Halstead, M H** *Elements of software science* North-Holland (1977)

14 **DeMarco, T** *Controlling software projects: management, measurement and estimation* Yourdon Press, New York (1982)

15 **DeMarco, T** 'An algorithm for sizing software products' *Performance Evaluation Review* 12 pp 13–22 (1984)

16 **Albrecht, A J** 'Measuring application development productivity' *Proc. Joint SHARE/GUIDE/IBM application development symp.* (October 1979)

17 **Gordon** 'Before You Leap' *User's Guide* Gordon Group (1986)

18 **Boehm, B W** 'Software engineering economics' *IEEE Trans. Soft. Eng.* Vol 10 No 1 (January 1984)

19 **Rudolph, E E** 'Productivity in computer application development, Department of Management Studies' *Working paper No 9* University of Auckland (March 1983)

20 **Rudolph, E E** 'Function point analyses, cookbook' own edition from Rudolph (March 1983)

21 **Symons, C R** 'Function point analysis: difficulties and improvements' *IEEE Trans. Soft. Eng.* Vol 14 No 1 (January 1988)

22 **Symons, C R** *Software sizing and estimating—MARK II FPA* Wiley (1991)

23 **Putnam, L H** 'A general empirical solution to the macro software sizing and estimating problem' *IEEE Trans. Soft. Eng.* SE-4, 4 (1978)

24 **Putnam, L** 'Software costing estimating and life cycle control' *IEEE Computer Society Press* (1980)

25 **Putnam, L H and Fitzsimmons, A** 'Estimating software costs' *Datamation* (Sept. Oct. Nov. 1979)

26 **Londeix, B** *Cost estimation for software development* Addison-Wesley (1987)

27 **Rubin, H A** 'Interactive macro-estimation of software life cycle parameters via personal computer: a technique for improving customer/developer communication' *Proc. Symp. on application & assessment of automated tools for software development, IEEE*, San Francisco (1983)

28 **Rubin, H A** 'Macro and micro-estimation of maintenance effort: the estimacs maintenance models' *IEEE* (1984)

29 **Rubin, H A** 'A comparison of cost estimation tools' *Proc. 8th Int. Conf. Soft. Eng. IEEE* (1985)

30 **Computer Associates** CA-Estimacs *User Guide* Release 5.0 (July 1986)

31 **Jones, C** *Programming productivity* McGraw-Hill (1986)

32 **BIS/Estimator** *User manual version 4.4.* BIS Applied System Ltd. (1987)

33 **Kemerer, C F** 'An empirical validation of software cost estimation models' *Communications of the ACM* Vol 30 No 5 (May 1987)

34 **Norden, P V** *Useful tools for project management* (Operations research in research and development) Wiley (1963)