TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Calibrating a software cost estimation model : why and how

Citation for published version (APA):
Ceulenaere, A. M. E., Genuchten, van, M. J. I. M., & Heemstra, F. J. (1987). Calibrating a software cost estimation model : why and how. *Information and Software Technology*, *29*(10), 558-568.

Document status and date:
Published: 01/01/1987

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 25. Aug. 2022

Calibrating a software cost estimation model: why and how

by A M E CUELENAERE, M J I M van GENUCHTEN and F J HEEMSTRA

Abstract: Calibration, has been found to be difficult in practice. Wide experience in using the estimation model is necessary; experience which the beginner naturally lacks. This paper indicates why it is important to calibrate a model and how the inexperienced user can be helped by an expert system. In addition, the development of, and experience with, the prototype of an expert system are described. The system dealt with here is intended for the calibration of the PRICE SP estimation.

Keywords: Software estimation, prototyping, software project planning, calibration, estimation model

Experience has shown that planning and estimating software projects is a difficult task. Budgets are constantly exceeded and agreements about delivery times have to be repeatedly updated. The two most important causes of this are as follows.

First, the large number of factors that influence the costs and duration of a software project. An investigation of these factors has shown[1] that there are no straightforward definitions of factors such as the volume, quality and complexity of the software used. In addition, it has proved difficult to quantify a number of factors and it is necessary to resort to criteria such as many, normal and few. Subjectivity plays a part in this; what one software designer classifies as 'a great many' may be regarded as belonging to the category 'many' by another. In addition, it has proved difficult to determine the effect of a particular factor on software costs. Studies on this occasionally contradict each other. A further difficulty is that the various factors influence each other mutually. Another problem is that when developing a new program, the software designer has to make an estimate of the costs and duration in advance. Among other things, this involves estimating the values of the cost drivers. For example, how many lines of code, or how many function points will the program comprise and what will the complexity be? In addition to the problems mentioned above, the uncertainty about the values of the factors also plays an important part now.

Second, the lack of data on completed projects. Knowledge of and experience with developing software, with specific product and project characteristics and with the influence of cost drivers only exists in the heads of a few people. For others who are confronted with such problems it is difficult, if not impossible, to locate this fragmentary and often unstructured knowledge and experience. In this way, mistakes are repeated. A databank with old project data, in which the knowledge and experience from the heads of the individuals are made explicit, can support project management in estimating the time, money and resources required by offering relevant information on old and comparable projects[2].

Under the increasing pressure to control the costs and lead time involved in software development there is a growing stream of publications on this subject and in the past ten years various models have been designed for estimating software costs. These are known as cost estimation models. Examples are COCOMO[3], SLIM[4] and Jensen's JS-2 and JS-3[5]. In these a project to be estimated is characterized in terms of the input variables of a model and, among other things, the model calculates the costs and the lead time of the project. Such models are based on a large number of historical projects and frequently projects from the USA. It is necessary to adapt such models to the environment in which they are to be used: in other words, the model must be calibrated. The environments in which software is developed differ so sharply that one environment cannot act as a model for the other. In calibration, values are assigned to one or more model variables. These values are derived from projects which have been carried out in the environment in which the model is to be used. An incorrect calibration has a negative effect on the quality of the subsequent estimates made with the model.

Nederlandse Philips Bedrijven, EDP-Industriële Toepassingen, Building HKB-4, PO Box 218, 5600 MD Eindhoven, The Netherlands

 0950-5849/87/100558-10$03.00 © 1987 Butterworth & Co (Publishers) Ltd.

Calibrating a model is a problem. To be able to carry out calibration, data on historical projects should be available. As already mentioned, it is precisely this information that is lacking. The above mentioned problems relating to the cost drivers are encountered when assigning values to model variables. In addition, when performing the calibration the user is often meeting the model for the first time, whereas experience is needed in using the model to be able to calibrate properly. A possible solution to this problem is to make the experience required for calibrating the model available to the inexperienced user. One way to do this is to use an expert system. This article describes the development of an expert system such as this for the PRICE SP cost estimation model used at Philips.

The next section deals in greater detail with the importance of calibration. Next, the PRICE SP model is described in broad outline and it is indicated why this model is difficult to calibrate. The section entitled 'An expert system as an aid' explains the part which an expert system can play in solving these difficulties. In addition, the development of the prototype of an expert system of this kind is described and the initial experiences with the prototype are discussed. The article ends with conclusions and recommendations.

This article is based on a study carried out by Eindhoven University of Technology (department of industrial engineering and management science, management information systems and automation group) and Philips (EDP – Industriële Toepassingen).

Need for calibration

The literature on cost estimation models is unanimous in its verdict that calibration is needed for every model, regardless of its type. Models such as COCOMO[3], SLIM[4], JENSEN[5] and ESTIMACS[6] are based on project data from a specific software development environment. For example, the comparisons in Boehm's COCOMO model are derived from a database of 63 projects carried out between 1964 and 1979 by the US company TRW Systems. It is doubtful whether such a collection of project data is representative enough for a development environment in Europe in 1987. Can the same cost drivers be distinguished in both situations and is their influence on costs and lead time the same in both situations?

For example, in the COCOMO database only seven of the 63 projects were developed in a semidetached environment. Of these seven, only one relates to the category of business applications. The program in question comprises 132000 lines of code, is programmed in PL/1 and the values for the cost drivers generally do not differ much from the nominal values. Obviously, an organization which mainly focuses on the development of administrative software, uses RPG as a programming language, and operates in a semidetached environment will find little or nothing to go by in the COCOMO database. The situation is even worse if the relevant organization uses methods and techniques which were not yet in existence at the time of the COCOMO projects. Examples which come to mind are fourth and fifth generation equipment, development environments, workbenches, prototyping and enduser computing. For these reasons, before using a model in a specific development environment for the first time it must be tailored for that environment. In other words: it is necessary to calibrate.

This need is underlined by a number of studies. For example, Kemerer[7] investigates whether cost estimation models can be generalized for environments other than those in which they have been developed. To answer this question he uses data from 15 completed projects. All these projects relate to comprehensive business applications. With the aid of four uncalibrated models an estimate is made of the number of man months required. Kemerer does this by using COCOMO, SLIM, ESTIMACS and Function Point Analysis (FPA). For each model and each project he investigates the difference between the estimated and the actual number of man months. For both COCOMO and SLIM it turns out that the estimate given is too wide for all the projects. When using SLIM the average overshoot is 772%, with COCOMO (regardless of whether basic, intermediate or detailed COCOMO is used) it is 600%. FPA and ESTIMACS give distinctly better results with overshoots of 100% and 85%, respectively. The results after calibration of the models proved to be significantly better. The figures show that cost estimation models cannot be transplanted to a different environment without paying the penalty. Accordingly, Kemerer advocates calibration.

A similar study was carried out by Rubin[6]. He made a comparison between the JENSEN, SLIM, GECOMO (a variant of COCOMO) and ESTIMACS models. Using these models, an estimate was made of the number of man months and the duration for the development of a specific (administrative) program. From Table 1 it can be seen that the estimates vary greatly. Rubin's explanation of this is that the models are based on various databases of historical projects and have not been calibrated for the specific development environment.

In the study done by Kitchenham and Taylor[8], too, the need for calibration is demonstrated by evaluating the COCOMO and SLIM models with reference to a large number of projects. Like Kemerer, they show that for both models the estimates of costs and duration work out much higher than reality in almost every case.

A number of studies have concentrated on the COCOMO model with regard to the aspect of calibration. The choice of this model is obvious, because in his

Table 1. Comparison of cost estimation models (MM = man months and m = months)

		Models			
		JENSEN	SLIM	GECOMO	ESTIMACS
Estimation	Cost	940 MM	20 MM	363 MM	17100 hours
	Schedule	31 m	17 m	23 m	16 m

book *Software Engineering Economics*[3] Boehm gives a very clear explanation of the model, the necessary input, the output and the database of old project data which is used.

Two studies must be mentioned in this context. Miyazaki and Mori[9] made an extensive evaluation of COCOMO, using the data from 33 old projects. These differ from the COCOMO projects in that they were generally developed in a semidetached environment, were frequently programmed in COBOL and, on average, were considerably wider in scope. In this study, too, it is shown that in the absence of calibration a marked overestimation of costs and duration takes place, the average deviation being 166% and less than 20% in only 6% of cases. On the basis of these research results, Miyazaki and Mori adapt the COCOMO model by taking account of the specific characteristics of the environment in which the projects had originated. They do this by eliminating a number of cost drivers which were not relevant (in their situation) from the COCOMO model. In addition, they adjust the model by changing the influence values of the various factors on the basis of their old project data. The effect of this calibration speaks for itself. The average deviation after calibration amounted to only about 17%. If one places the evaluation data of Kemerer and of Miyazaki and Mori side by side, it is seen that in the first case there is an average overestimate of 600% and in the second case of an average of 166%. These differences show that development environments can vary greatly and calibration is therefore essential.

A similar study was carried out by Saalfrank *et al*[10]. They describe a procedure called COKAL with which models of the COCOMO type can be calibrated. An evaluation of COCOMO employing COKAL produces significantly better estimating results than when it is not used.

The conclusion from the above is that calibration of a cost estimation model with reference to a specific development environment is essential. Finally, it must be pointed out that calibration is not a one-off activity but must be repeated periodically. The characteristics of a development environment can change in the course of time as a result of technological and methodological changes in software development and because of changes in personnel and organization. Recalibration then becomes necessary, in which respect weighting factors can be introduced to allow the influence of projects on the calibration to increase in line with the recentness of the project.

Now that the need for calibration has been demonstrated, the next section will discuss the calibration of a cost estimation model, namely PRICE SP.

PRICE SP

Background to the PRICE models

The PRICE models are used for estimating hardware and software costs. PRICE stands for Programmed Review of Information for Costing and Evaluation. The models have been developed and are supported by RCA PRICE Systems, part of General Electric. The number of man months and the lead time required for software projects are estimated by using the PRICE SP (Software Productivity) model.

The content of the PRICE models is secret. A model remains a black box, even for those who rent it. The PRICE user sends his or her input data via a modem to a time-sharing computer in the USA, UK or France and receives the estimate almost directly by return. In spite of this restriction and the high rental price the model is widely used in the United States. Some users are:

- Boeing
- General Dynamics
- US Ministry of Defense
- IBM
- General Electric
- Texas Instruments.

PRICE is less widely used in Europe.

PRICE SP model

A diagram showing the input and output of the PRICE SP model is presented in Figure 1.

The output of the model essentially consists of an estimate of the number of man months and the lead time required for the project. These are the dependent variables. As already mentioned, the model itself is and

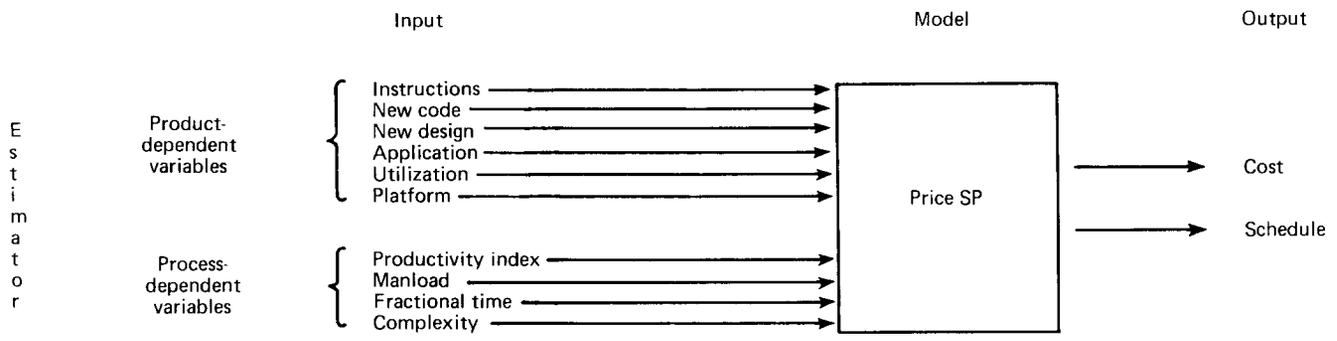

Figure 1. Diagram of PRICE SP

remains a black box. The input for the model is a characteristic of the project to be estimated. The characteristic consists of ten variables, called the independent variables. These variables and their definitions are shown in Table 2. The definitions are taken over literally from the PRICE manual[11]. Since these are sometimes not very enlightening, a brief explanation is given where necessary.

Input for PRICE SP

The ten input variables can be divided into two groups (see Figure 1). The first group of six variables describes the product to be developed. This group consists of the variables 'instructions', 'new code', 'new design', 'application', 'utilization' and 'platform'. The second group of four variables describes the development process that must result in the required product. This group consists of the variables 'productivity index', 'manload', 'fractional time' and 'complexity'.

The variable *instructions* is a measure for the size of the program. The user can state the number of source lines or the number of instructions.

The variables *new code* and *new design* indicate what part of the product must be entirely redeveloped. It may be possible for a part of the code and/or of the design to be taken from the literature or from a previous project. In such a case, the values of the variables new code and new design will be smaller than one.

As the term implies, the variable *application* describes

Table 2. Input variables for PRICE SP

Variable	Description PRICE manual[11]	Explanation
Instructions	is the total number of deliverable, executable, machine level instructions.	volume of the program, the manual talks about machine level instructions, however, some experienced users consistently use lines of source code as input
New code	is the amount of new code.	value between 0 and 1
New design	is the amount of new design.	value between 0 and 1
Application	summarizes the application mix of instructions.	value between 0.8 and 11 (see Table 2)
Utilization	is the fraction of available hardware cycle time or total memory capacity used.	value between 0 and 1
Platform	describes the planned operating environment for the software.	value between 0.6 and 2.5 (see Table 3)
Productivity index	is an empirically derived parameter that serves as a productivity, skill level, experience and efficiency index.	
Manload	is the average number of software personnel involved in the software project over the entire project.	
Fractional time	is the average fractional time dedicated to the software job.	
Complexity	describes the relative effect of complicating factors such as product familiarity, personnel software skills, hardware/software design interactions as they effect manpower costs.	(see Table 4)

Table 3. Table for determining the value of the variable 'application'

	Weight	Identifying characteristics
Operating systems	10.95	Task management. Memory management. Heavy hardware interface. Many interactions. High reliability and strict timing requirements.
Interactive operations	10.95	Real time man/machine interfaces. Human engineering considerations and error protection very important.
Real time command and control	8.46	Machine to machine communications under tight timing constraints. Queuing not practicable. Heavy hardware interface. Strict protocol requirements.
Online communications	6.16	Machine to machine communications with queuing allowed. Timing restrictions not as restrictive as with real time command and control.
Data storage and retrieval	4.10	Operation of data storage devices. Database management. Secondary storage handling. Data blocking and deblocking. Hashing techniques. Hardware oriented.
String manipulation	2.31	Routine applications with no overriding constraints. Not oriented toward mathematics. Typified by language compilers, sorting, formatting, buffer manipulation, etc.
Mathematical operations	0.86	Routine mathematical applications with no overriding constraints.

the kind of application. The user is expected to determine the value of this variable by selecting the class which best describes his project from Table 3. This is no easy task because of the sometimes vague description of the classes.

Possible hardware restrictions are described in the variable *utilization*. An example of this is the limited memory space in the computer on which the software product to be developed must operate.

The last variable which characterizes the project is *platform*. Platform describes the environment in which the software product to be developed will be used. The value of this variable for a department will be the same for the various projects. The user is expected to determine the value of the variable platform by means of Table 4. It is doubtful whether this table is suitable for general use as an aid. Intuitively, however, it is clear that the various platform make differing demands on the software to be developed.

Productivity index is a variable that is determined by calibration on the basis of a number of completed projects. Calibration of the model and the variable 'productivity index' are dealt with in the next section.

Table 4. Table for determining the value of the variable 'platform'

Operating environment	Platform
Production center internally developed S/W	0.6–0.8
Production center contracted S/W	1.0
MIL-spec ground	1.2
Military mobile (van or shipboard)	1.4
Commercial avionics	1.7
MIL-spec avionics	1.8
Unmanned space	2.0
Manned space	2.5

The variable *manload* is incorporated in the model because people and time are not mutually interchangeable. For example, in order to make the same product, five people (manload = 5) need more man months than two people (manload = 2). The difference is caused, among other things, by the time-consuming mutual communication. This fact is described in *The Mythical Man Month* by Brooks[12].

The variable *fractional time* describes the fact that fragmentation of attention leads to lower productivity[13]. In the software world, it frequently happens that people are partly engaged in developing a new product and partly in maintaining previous products. For example, if someone is engaged on a project for three of the five days each week, the value of the variable fractional time for this project is 0.6.

Finally, the variable *complexity* describes project characteristics which mainly influence lead time. The standard value of this variable is 1.0. Any deviations from this standard value must be determined by the user on the basis of Table 5.

Example: If a similar project has been carried out previously, but the language is new to the people on the project and the project organization is multinational, then the value of the variable complexity is 1.3 (see Table 5). For that matter, even in this simple example it is clear that several interpretations of the table are possible. For example, a 'multinational project' often means that the project is carried out at more than one location. So should the standard value of the variable complexity now be adjusted by 0.4 or 0.6?

Calibration of PRICE SP

The PRICE SP model is calibrated by describing a number of completed projects with the model. The dependent variables are now no longer the number of man months and lead time required, but the variable *productivity index*. Since the model must, as it were, be used in reverse, RCA has called the adapted calibration of the model ECIRP. A diagram of the adapted calibration of the model is shown in Figure 2 (for comparison, see Figure 1).

Table 5. Table for determining the value of the variable 'complexity'

	CPLX Adjustment	Example
Personnel		
Outstanding crew, among best in industry	−0.2	
Extensive experience, some top talent	−0.1	
Normal crew, experienced	0	
Mixed experience, some new hires	+0.1	
Relatively inexperienced many new hires	+0.2	
Product familiarity		
Old hat, redo of previous work	−0.2	−0.2
Familiar type of project	−0.1	
Normal new project, normal line of business	0	
New line of business	+0.2	
Complicating factors		
First time with language	+0.1	+0.1
First time with processor	+0.1	
New language	+0.2 to +0.3	
New hardware	+0.2 to +0.3	
More than one location/ organization	+0.2	
Multinational project	+0.4	+0.4
Hardware developed in parallel or many changing requirements	+0.2 to +0.3	
Assembly language	+0.2 to +0.3	
		+0.3

The value of the variable productivity index is determined for, say, between five and ten projects carried out by the department concerned. Since the projects are realized in the same environment, the values for the variable productivity index determined with the model must not differ too greatly from each other. Once this condition is satisfied, then the value to be used in making the estimate has been determined. If this condition is not satisfied, then it is not yet possible to start estimating new projects by means of the PRICE SP model. Obviously, if there are great differences between the departments in an organization the model must be calibrated separately for these departments.

PRICE is based on the assumption that the model can be calibrated by determining the value of the variable productivity index on the basis of a number of old projects. The differences between the organizations must be reflected in the value of that single variable productivity index. Information is lost as a result of modelling. It is, however, true to say that calibrating in only one variable is better than not calibrating at all.

Problems with calibration

The model can only be successfully calibrated if the model variables are interpreted and evaluated in a consistent manner. If this requirement is not met, there is a danger of calculating in the direction of an apparently accurate productivity index. In other words, one keeps working on the input variables until the model determines the same value of the variable productivity index for various completed projects. This index can, however, be completely wrong, with the result that there is a systematic error in the estimates.

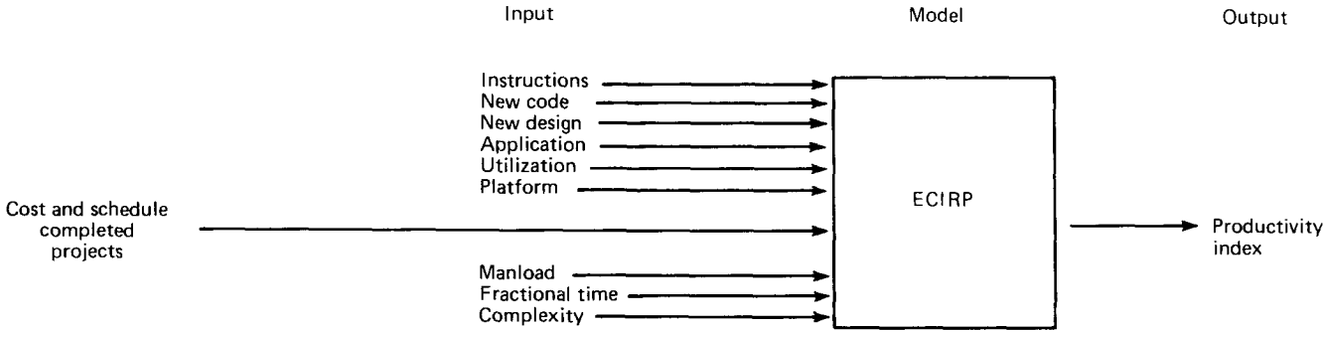

Figure 2. Diagram showing the calibration mode of PRICE SP

In practice it has proved that interpreting and evaluating the input variable consistently presents problems for the beginner. Three causes are discussed here. As already indicated, calibration often represents the first introduction to the model. The user is frequently not accustomed to thinking in terms of the model. It is not easy to interpret the various tables correctly (e.g. Tables 3, 4 and 5). The authors believe that the variables and the tables constitute a poor interface between the model and the beginner.

The second cause of the problem with calibration is the fact that the content of the models is secret. As a result, it is difficult to develop a feeling for the effect of the various values of the variables on the results obtained with the model. By way of example, a relationship between the variable *utilization* and the number of man months required is shown in Figure 3. The difference between the values 0.8 and 0.9 has a much greater effect on the number of man months than the difference between 0.6 and 0.7.

The third cause of the problems which occur during calibration is that the registration of completed projects sometimes contains insufficient information to enable the ECIRP input variables to be determined. The importance of having sufficient information about old projects has already been discussed.

A possible solution

The output of ECIRP is a value for the variable 'productivity index'. The input consists of the values of 11 variables: nine to characterize the project and two to express the number of man months and lead time realized.

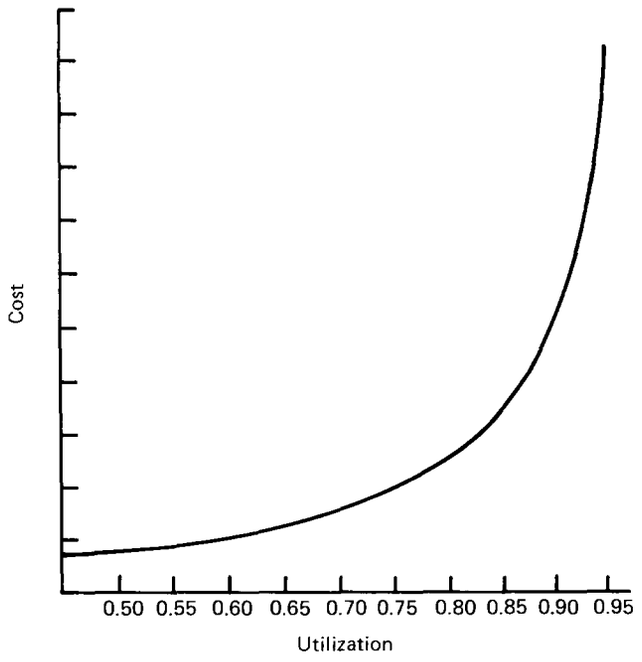

Figure 3. Relation between the variable 'utilization' and the number of man months required

In the previous section it was stated that calibration represents the first introduction of the user to the model. On the other hand, calibration requires the necessary experience in using the model. It is therefore necessary to support the model user during calibration. A possible way to do this is to support the user with the expertise of experienced users.

It has been found that the experienced users of the model employ a great many heuristic procedures for determining the value of the input variables. This expertise was developed during the years when these users were working with the model. Transferring this knowledge and experience to new users has proved to be a time-consuming business. If it is possible to store these heuristic procedures in a system, then the beginner will always have a practical aid at hand. A possible form in which a system such as this can be achieved is an expert system.

An expert system as an aid

Role of the expert system in calibration

An expert system is a computing system capable of representing and reasoning about some knowledge-rich domain, with a view to solving problems and giving advice[14]. For extensive information on expert systems the reader is referred to the many books and articles which have appeared on this subject recently[15,16].

As stated earlier, the expert system must make expertise in using the model available to the beginner in using PRICE SP. That means expertise which the expert has built up over the years. The expert system may be regarded as an interface between the person who is calibrating and the input of the ECIRP model. The system will not generate a value for all the input variables of ECIRP, but only for those variables which require a great deal of expertise for determining the value. For example, there is no point in giving support for the determination of the value of the variable 'instructions' because in calibration this only involves counting. The input of the expert system consists of answers which the estimator gives to questions asked by the system. The role of the expert system is shown in Figure 4.

It will have to be easier to answer the questions of the expert system than to determine the input for ECIRP. In this way, the gap between the estimator and the model is reduced. In view of the need for calibration and the consequences of an inaccurate calibration, the expert system will constitute a useful aid in estimating software projects.

A prototype has been developed to investigate whether an expert system such as this is possible. This prototype

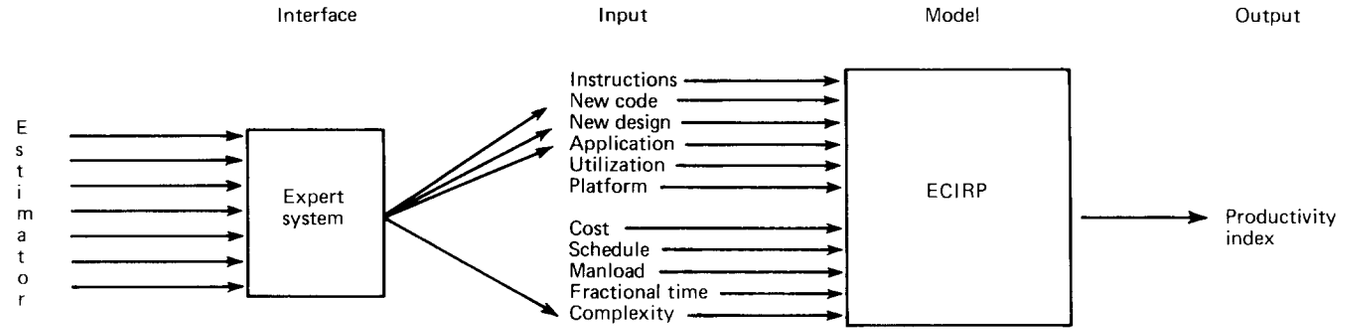

Figure 4. The expert system as an interface between the estimator and ECIRP

generates numerical values for the input variables 'complexity', 'application', 'new design' and 'new code' (see Figure 4).

To develop the expert system 17 discussions were held with the expert over a period of three months. In order to show the structure of the knowledge, a form of representation was selected which from now on will be indicated by the term 'decision trees'. On the basis of the decision tree, new discussions were held, resulting in the decision tree being adapted and extended. The developed system is a rule based expert system, this means that the knowledge is stored in the form of production rules ('if-then' rules).

Results

The prototype which was developed generates values for four input variables: complexity, application, new design and new code. The total system comprises about 200 rules. During the discussions with the expert it was found that there are factors which have an influence on several variables. For example, if it is clear when determining the value of the variable complexity that there are experienced developers who have already designed similar systems, then it is unlikely that the design will be totally new. In other words, it is unlikely that the variable new design will be given the value one. It therefore proves that the sub-areas complexity, application and new code/new design cannot be separated.

Some aspects of the development of the expert system are explained in greater detail. The underlying expertise for the variables application and complexity are discussed. The results of testing the accuracy of the expert system are then discussed with regard to these variables.

First, application. It will be explained how the expert, and hence the expert system, determines the value of the variable application. According to the PRICE manual the determination of this value involves making a choice from seven classes of projects (see Figure 5a). The fact that it involves more than only a choice can be seen from the way in which the expert determines the value.

In establishing the value of the variable application, the following fundamental process can be distinguished: in a program, data is retrieved and formatted, if necessary, to carry out a calculation process with them. After this

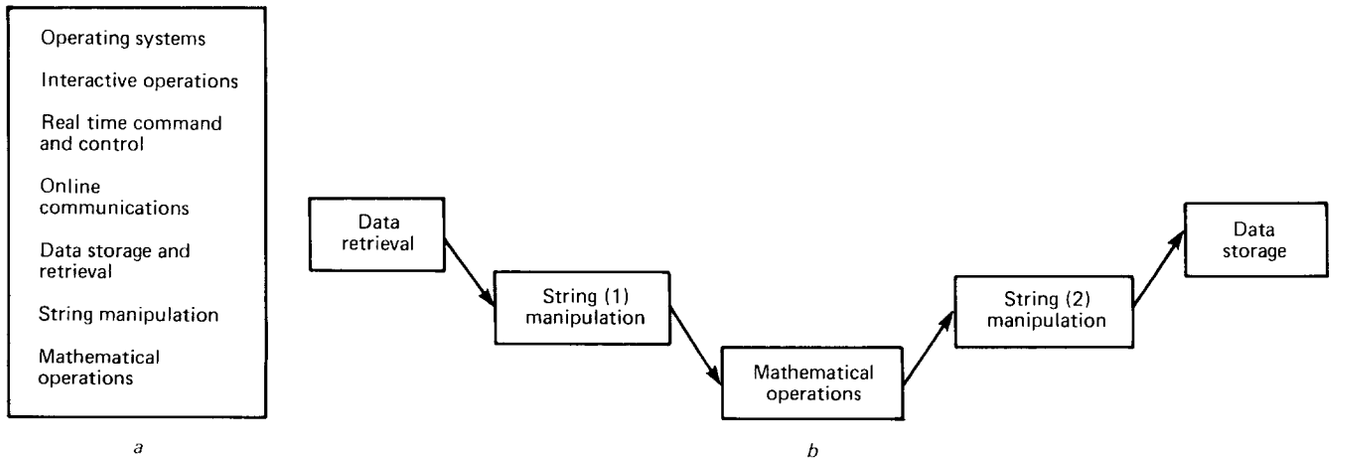

Figure 5. The classes according to RCA and the fundamental process

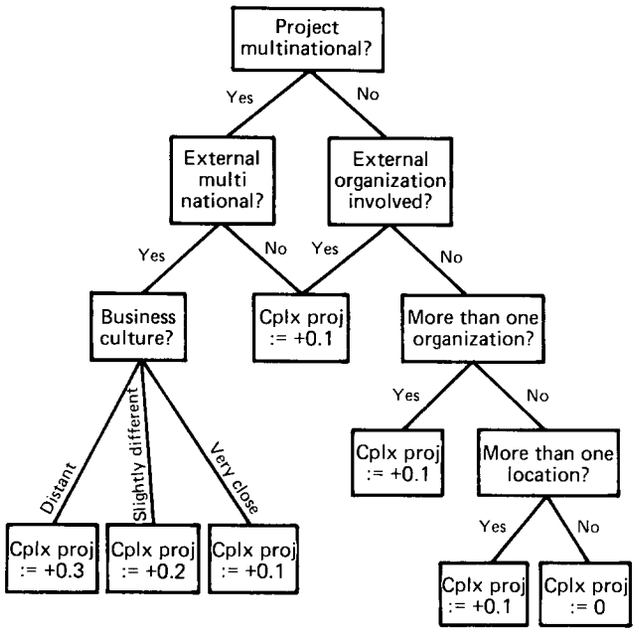

Figure 6. A decision tree

process, the data is formatted again and then stored somewhere. This process is shown in Figure 5b.

The user is asked to indicate what percentage of his or her program can be regarded as belonging to data retrieval, string manipulation (1), mathematical operations, string manipulation (2) and data storage. These five parts of the fundamental process correspond to the bottom three classes of the RCA application table (compare Figures 5a and 5b). Next, the user is asked whether any communication takes place with other systems. If so, it must then be stated what percentages of these five parts of the fundamental process have to do with communication. In this way, it is possible for some of the data to be retrieved through communication with another system. For example, if the user has classified 20%of the fundamental process as data retrieval, then half of this 20% can now be 'upgraded' to the class 'communication'. As a result, the value of the variable application becomes higher. In the same way, the program is next examined to see what part must be classified as real time command and control and as interactive operations (see Table 3). The foregoing ultimately results in a division of the program into the six classes from the application table (the class 'operating system' is considered separately). After this division, determining the value of the variable 'application' is only a matter of calculation.

As a second example, a part of the knowledge required for determining the value of the variable complexity is shown. The decision tree is shown in Figure 6. As stated, the decision trees acted as a guideline in the discussions with the expert. Two-thirds of the effort involved in the total development consisted of structuring and representing the knowledge in this form.

Third, testing complexity and application. The accuracy of the stored knowledge for determining the value of the variables 'complexity' and 'application' was determined on the basis of six projects. The test results are presented in Table 6. The columns marked A contain the values determined by the expert in the past. The columns marked B show the values generated by the expert system. The differences are indicated in the columns marked C. In the case of the variable 'complexity' only the deviations from the standard value are stated. A difference of less than 0.1 for the value of this variable may be described as good, because a difference such as this results in only a slight error in the output of the ECIRP model. A difference in value of 0.5 is acceptable for the variable 'application' (see also Table 3).

The expert system was subjected to a user test on a modest scale. For some questions it was found that the number of possible answers was too limited. Additions were made in consultation with the expert. The user test also showed that some heuristics are tied to place or time. Clearly, in developing a definitive system an important place will have to be given to testing the accuracy of the stored knowledge and to the user test.

Table 6. Test results for the variables 'complexity' and 'application'

Application				Complexity			
Project	A Expert	B Expert system	C Difference	Project	A Expert	B Expert system	C Difference
1	8.04	7.50	0.54	1	0.09	0.10	0.01
2	7.28	7.24	0.04	2	0.23	0.21	0.02
3	5.41	4.93	0.48	3	0.25	0.30	0.05
4	5.00	4.68	0.32	4	−0.08	−0.10	0.02
5	7.09	6.89	0.20	5	0.58	0.50	0.08
6	4.73	4.33	0.40	6	0.90	0.70	0.20

Conclusions and recommendations

The estimation of software projects is important and has proved to be a difficult task in practice. Models are an aid to estimating. Estimation models should be calibrated. Calibration must be carried out accurately because this is the basis for every subsequent estimate made with the model. Various studies have underlined the need for calibration, but this is difficult to perform. An important cause of this is that when performing the calibration the user is often meeting the model for the first time, whereas calibration requires experience in using the model. This experience is necessary to be able to interpret and evaluate the input variables of the model in a consistent manner. The authors believe that in making the calibration the user should be supported by expertise from an experienced model user. One means of distributing expertise is an expert system. The authors have developed a prototype of an expert system to support the characterization of projects for the purpose of calibration in the PRICE SP input variables. The results to date and the positive reactions of both the users and the expert have shown that an expert system is a suitable aid for this application.

The authors have tried to improve the determination of the input for the calibration of PRICE SP by means of an expert system. Another possible way of improving model estimations is to adapt the model itself. Here it is necessary, to use input variables with values that can be easily estimated. No matter how well a model may describe the development of software, if the values of the input variables are difficult to define then the estimates made by the model, and hence the model itself, will be of limited value.

References

1 **Heemstra, F J** 'Wat bepaalt de kosten van software' *Informatie* Vol 29 No 7 special (1987)

2 **Noth, T** 'Unterstützung des Softwareprojektmanagements durch eine Erfahrungsdatenbank' *Proc. Compas '87* Erfolgsfaktoren der integrierten Informationsverarbeitung, AMK Berlin, FRG (May 1987)

3 **Boehm, B W** *Software Engineering Economics* Prentice Hall, Englewood Cliffs NJ, USA (1981)

4 **Putnam, L H and Fitzsimmons A** 'Estimating software costs' *Datamation* September, October and November (1979)

5 **Jensen, R W** 'A comparison of the Jensen and COCOMO schedule and cost estimation models' *Proc. ISPA Sixth Annual Conf.* (May 1984) pp 96–106

6 **Rubin, H A** 'A comparison of cost estimation tools' *Proc. 8th Int. Conf. on Software Engineering* IEEE (1985)

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

8 **Kitchenham, B A and Taylor, N R** 'Software project development cost estimation' *J. Syst. Software* Vol 5 (1985)

9 **Miyazaki, Y and Mori, K** 'COCOMO evaluation and tailoring' *Proc. 8th Int. Conf. on Software Engineering* IEEE (1985)

10 **Saalfrank, R F, Schelle, H and Schnopp, R** 'Produktivitätseffekte von aufwandeinflubgrobe bei der softwareeentwicklung' *Angewandte Informatik* No 3 (1987)

11 *RCA PRICE Systems* PRICE S/SP manual (1985)

12 **Brooks, F P** *The Mythical Man-Month, Essays On Software Engineering* Addison-Wesley, Reading MA, USA (1975)

13 **Sierevelt, H** 'Observations on software models' *J. Parametrics* Vol 6, No 4 (December 1986)

14 **Jackson, P** *Introduction to Expert System*, Addison-Wesley, Reading MA, USA (1986)

15 **Harmon, P and King, D** *Expert Systems, Artificial Intelligence in Business*, Wiley Press NY, USA (1985)

16 **Waterman, D A** *A Guide to Expert Systems* Addison-Wesley (1986)

Further reading

Cuelenaere, A M E, Genuchten, M J I M van and Heemstra, F J 'Een expert systeem voor het gebruik van een software begrotingsmodel' *Informatie* Vol 29 No 7 special (1987)

Hayes-Roth, F, Waterman, D A and Lenat, D B Building Expert Systems, Addison-Wesley, Reading MA, USA (1983)

Liebowitz, J 'Useful approach for evaluating expert systems' *Expert Syst.* Vol 3 No 2 (April 1986) □