On the use of software cost models

Applications

On the use of software cost models

Michiel van Genuchten

Management Information Systems and Automation, Department of Industrial Engineering, University of Technology Eindhoven, 5600 MB Eindhoven, Netherlands

Hans Koolen

Department OAP, Hollandse Signaal Apparaten, 7550 GD Hengelo, Netherlands

A number of methods and tools have been developed over the years to meet the increasing need to control software development. Among the tools are software cost estimation models. Well-known examples are COCOMO, PRICE S, Estimacs, and Function point analysis. The limited and often unsuccessful use of cost models is the motivation behind this paper. The authors oppose the idea that a model is the solution to the estimation problem, but are convinced that the use of a model can contribute to the control of software development, if it is used properly. The model should be used to generate a second opinion. It will have value as a means of communication, as a checklist, and as it forces the user to collect data on the development process. Certain organizational requirements should be fulfilled to be able to use the model properly. The proper use of the model and the organizational requirements are the main subjects of this paper.

Keywords: Software cost estimation, Software cost models, Use of software cost models, Software engineering control, Project management, Organizational requirements for use.

1. Introduction

A number of methods and tools have been developed over the years to meet the increasing need to control software development. The growing stream of publications on software cost estimation is dominated by those on one of the tools: the software cost estimation model. Well-known software cost models are COCOMO, PRICE S, Estimacs, and function point analysis. In practice, however, the models are used rarely. A recent survey showed that only 14 per cent of the respondents used a model to estimate software projects [6,11]. The same survey and our own observations indicated that the organizations that do use a model do not always use it successfully.

The limited and often unsuccessful use of cost models is the motivation behind this paper. Our aim is to describe how models can be applied successfully. The paper is based on experience with the successful introduction of a model in one large software development department and observations on the introduction of a model in a number of others. We oppose the idea that a model is the solution to the estimation problem, but we are convinced that it can contribute to the control of software development if it is used properly and if certain organizational requirements are fulfilled.

Michiel J.I.M. van Genuchten is employed as a consultant by Philips International (Lighthouse Consulting Group). He also works as a researcher at the department of Industrial Engineering at the University of Technology in Eindhoven in the Netherlands. He received a M.Sc. in Industrial engineering. His research interests include control and analysis of software engineering, the application of logistics concepts in software engineering, and reuse of software.

Hans J.A.H.M. Koolen received a M.Sc. in Applied Mathematics (Operations research) from the University of Technology in Eindhoven in The Netherlands. His practical experience includes seven years within Hollandse Signaal Apparaten, that has become part of Thomson CSF. He worked for several years as software engineer. He was responsible for the introduction of the estimation model PRICE S in a large software development department.

2. Software cost models

2.1. Software cost estimation methods

Over the years, a number of software cost estimation methods have been used. According to Boehm [3], three acceptable methods can be distinguished: the expert method, the analogy method, and the use of software cost estimation models. The expert and analogy method are discussed briefly; the use of models will be discussed extensively.

Expert judgement is widely applied as an estimation method. It involves consulting with one or more experts. The expert is usually an experienced project leader who uses experience on past projects and understanding of the proposed project to arrive at an estimate of its cost and development time. An advantage of the expert method is that an expert can consider specific project conditions, such as an extremely experienced staff or changing requirements.

Estimation by analogy involves reasoning by analogy, using experience with one or more completed projects to relate actual cost and development time to the cost and development time of the new project. Differences are determined and their impact on cost and development time is estimated.

There are obvious similarities between the expert and the analogy method. A difference however is that estimation by analogy is based on recorded facts: results from one or more specific, completed projects. In practice, a combination of the expert and analogy method will often be applied.

Before we discuss the operation of a model, we describe the development of a model, because this gives insight into its operation and determines some of its limitations. The development of a model starts with data collection on a number of completed software projects. Based on the data and theoretical knowledge, the developers try to develop a descriptive model. The dependent variables of the model are those to be estimated, i.e., the effort and development time of the completed projects. The independent variables are a subset of the known cost drivers, such as complexity, kind of application, and the experience of the development staff. The model consists of a number of mathematical expressions that describe the relation between the cost drivers and the dependent variables. Most of the existing models use the size of the software product as an independent variable; this is usually expressed in the number of lines of source code, which is a usable variable when building a descriptive model. The relation between this variable and the cost and development time of a project is obvious.

Figure 1 gives a simplified model that estimates the cost and development time of a project based on five independent variables. Development time is sometimes referred to as schedule, implementation time or lead time.

The meaning of the first four independent variables is obvious. The productivity index defines the productivity of the environment in which the project is going to be developed. This index is determined while calibrating the model to the environment. Existing models use more than five input variables. For instance COCOMO, uses seventeen input variables, including the size of the product, which is to be expressed in thousands of lines of source code.

2.2 Limitations of software cost models

We consider a model as one of the possible tools that can be used to estimate software projects but think it is necessary to be aware of the limitations of the tool in order to apply it successfully. Therefore three limitations of software cost models are discussed.

The first limitation is the fact that the commercially available models do not originate from the environment in which they are to be used. Most of the models originate in the United States and are based on projects that were completed a number of years ago. It is questionable whether such a collection of projects can be representative for a development environment in, for example, present day Europe. Examples of differences between the European and American situation are the different personnel turnover rates and the number of work hours per week. The impact of the latter on the development time of a project is obvious. Due to the rapid developments in software engineering, the situation of, say, ten years ago and the current situation differ so much that one environment cannot act as a model for the other. An attempt to bridge the gap between the two environments is the calibration of the model.

A second limitation is the difficulty of evaluat-

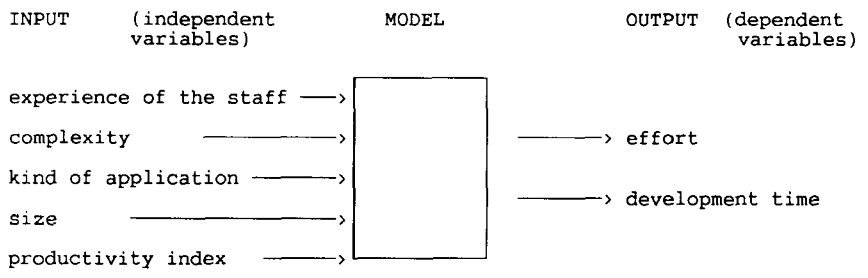

Fig. 1. A cost model.

ing some of the input variables. We have just described how cost models are developed as descriptive models. While an independent variable such as lines of source code may be very useful in describing completed projects, it may not be useful in predicting costs and development time of future projects. In the first case, the lines of code can be counted automatically, while in the latter case the lines of code are almost as hard to estimate as the effort and development time of the project. As Case [4] points out: such models "do a fairly good job of telling you how long the project will take – after you have written the code and then counted the lines". The same is true, to some extent, for variables such as the complexity and the personnel turnover. It may also be true for function points, a variable to describe the size of a software product as proposed by Albrecht [2], that is mainly used in information system development.

The third limitation of the existing models is the fact that no studies confirm the accuracy and usability of the models. The studies that have been made give very disappointing results. For instance Kemerer [7] estimated 15 completed projects using four uncalibrated models. He found an average overshoot of 772, 600, 100 and 85 per cent for the models SLIM, COCOMO, function point analysis, and Estimacs, respectively. Experiments by Mohanty [10] and Abdel Hamid [1] yielded similar results.

Recently, an experiment was conducted by the University of Technology Eindhoven for Philips on the early applicability of two software cost models [9]. During the experiment, experienced project leaders were asked to make a number of estimates for a project that had actually been carried out. The first estimate of the effort and development time was based on the project leaders' knowledge and experience. Next, two estimates were made using the packages selected, i.e. BYL and Estimacs. The projectleaders were asked to make a fourth, final estimate that was based on their knowledge and experience, combined with the insights gained from the use of the models. Some results of the experiment are given in *Table 1*. It should be noted that the models are not calibrated with respect to their environment. One should therefore be careful of direct comparison of the model estimates with reality. The standard deviation of the estimates made by the project leaders is, however, an indicator of the usability of the models.

As stated previously, the project had actually been completed. The real effort, and development time were:

Effort:	8 man-months
Development time:	6 months

The standard deviation of the estimates is huge. The difference between the model estimates and

Table 1
Some results of the experiment. The development time is given in months. The effort is given in man-months.

Variable	Mean	Standard deviation
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
Development time		
– manual estimate	11.2	3.7
– BYL estimate	8.5	2.4
– final estimate	12.1	3.4

Table 2
Strengths and weaknesses of software cost estimation methods [3].

Method	Strength	Weakness
Analogy	* Based on representative experience	* Representativeness of experience
Expert	* Assessment of representativeness, interactions, exceptional circumstances	* No better than participants * Biases, incomplete recall
Models	* Objective, repeatable, analyzable formula * Efficient, good for sensitivity analysis * Objectively calibrated to experience	* Subjective inputs * Assessment of exceptional circumstances * Calibrated to past, not to future

reality is also remarkable. Questions relating to the models were also answered by the people that developed the system. Feeding their answers into the models yielded the following results:

Effort with BYL:	18 man-months
Development time with BYL:	7.5 man-months
Effort with ESTIMACS:	54.4 man-months.

The conclusions of the experiment were based on quantitative results and the opinions of the project leaders concerned. An important conclusion was that, based on the difference found between the estimate and reality, it has not been possible to show that the selected models can be used for estimating projects at an early stage of their development. Another important conclusion was that, although both packages could not be evaluated as "good", the project leaders involved regarded the models as "useful".

Thus, we do not state that models are useless or inaccurate, we just assert that there are no studies that confirm that they are useful or accurate. One should not be discouraged by the limitations, but one should be aware of them in order to use the models appropriately.

3. The use of a software cost model

We now argue that one should use more than one estimation method and that a software cost model can act as one of the methods. We also describe the activities that will have to be performed if an estimation model is to be used.

3.1 The use of alternative estimation methods

Three methods to estimate software projects have been described. The expert and analogy method and the use of cost estimation models all have their strengths and weaknesses. Some of the limitations of software cost models have already been mentioned. Some of the strengths and weaknesses of the methods, as distinguished by Boehm are mentioned in *Table 2*.

With respect to the importance of accurate cost estimates and the strengths and weaknesses of the methods, we recommend the use of more than one method to arrive at an estimate. The decision to use alternative methods is, in our view, more important than the choice of the estimation method itself; the weak points of one method can be compensated by the strong points of another method. This is true especially for the expert or analogy method on the one hand and the use of software cost estimation models on the other, because their strengths and weaknesses are complementary. There is another way to cope with the weaknesses of the estimation methods: instead of using alternative methods, one could make several independent estimates using one method. Biases are considered to be a weakness in the expert method, so one could consult two experts independently and later have them discuss their estimates, thus decreasing the influence of possible biases. A disadvantage of this approach is that, despite their independence, the two estimators may look at the proposed project from the same angle and base their experience on the same projects.

Another alternative is to use an estimation model as a second method to arrive at an estimate. An advantage of this is that the use of a model forces one to look at the proposed project from another point of view, i.e. the point of view of the cost drivers that the model uses as independent variables. One is forced to evaluate the proposed project in a predefined way. This may point to omissions in the definition of the proposed project. As such the model systemizes the estimation process even before it has generated its first estimate. Looking at the project from different points

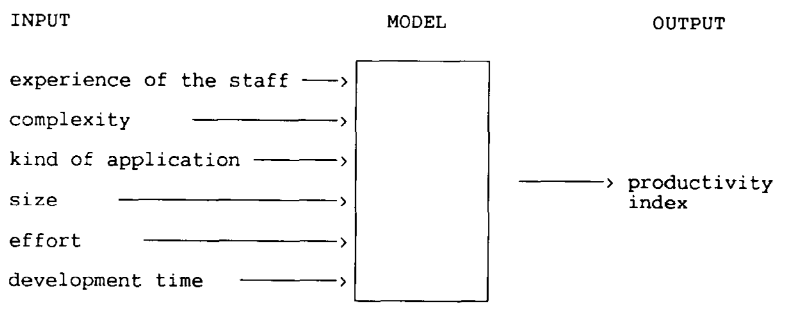

Fig. 2. Diagram of the calibration of a cost model.

of view decreases the risk of making an obvious mistake.

In our view, software cost estimation using alternative methods should be done according to the following pattern:

(i) using the analogy or expert method an estimate is made,

(ii) at the same time an estimate is made using a model,

(iii) if the two estimates agree, the estimate is accepted. If not, the reasons for the difference must be determined, preferably in a discussion with those responsible for the estimates. Experience in using this approach shows that differences can usually be explained from the use of different starting points by and presuppositions of the estimators. The process of (re)estimation has to converge in a small number of iterations. The differences between the estimates and the number of re-estimations required serve as an indication of the risk of the project. Consensus on the final estimation will increase thanks to discussion between estimators. Agreement must occur on the degree of difference in the estimations that will be tolerated, which will depend on the function of the estimation. For the first feasibility estimation larger differences will be tolerated but the difference should be small for a small project in a well-known field.

3.2. Calibration and estimation

The actual use of a software cost model maybe split into two phases: first calibration of a model; second actual use of the model.

Before a model can actually be used, it must be adapted to the environment of its use: it must be calibrated. This tests the fit of a model in an organization and enables the model to be adapted to the characteristics of that organization. The calibration involves describing a number of completed projects with the model. One way to do this is to evaluate the so-called productivity index. During the calibration, the dependent variable is a productivity index. A diagram of the calibration of a model is given in *Figure 2*.

The value of the productivity index is determined for, say, five to ten projects previously (recently) carried out by the department. Since these projects were carried out in the same environment, the values for the productivity index should not differ much from one to another. Once this condition is satisfied, the value of the productivity index is determined. If the condition is not satisfied, further calibration may be necessary [5].

Another way of calibrating a model is to adapt the weights ascribed to the variables. To calibrate a model in this way, two conditions must be fulfilled: first the content of the model must be known; this condition is not fulfilled for models such as PRICE, SLIM and Estimacs. The second condition is that a lot of data on completed projects must be available. For example, if we wish to calibrate the COCOMO model with approximately 75 weights attributed to 15 variables, then a lot of data are necessary to adapt the weights in an accountable manner. It is not likely that a single organization is capable of obtaining such volumes of data.

The time it takes to calibrate a model will vary according to the availability of data on completed projects. When calibrating a model, one should keep in mind that its use is just one of the ways to arrive at an estimate. The model and its calibration are not goals in themselves. This awareness will prevent endless calibration and will make it possible to use a model fairly quickly for its real purpose: to estimate proposed projects.

After the calibration, estimation starts. This is based on the information that is available on the proposed project. If the project leader has been selected, he is a possible candidate to evaluate the input variables in a discussion with an experienced model user. If the project leader is not yet known, somebody who can judge the proposed project must evaluate the input variables. Data on completed projects can be used to evaluate the input variables by comparing the values for the completed and the proposed projects. In this way, the input variables are used as a common language to compare the proposed to the completed projects.

A model should be used experimentally for some time. As soon as software developers and estimators have enough confidence in the model, more value can be attached to its estimates. Again, it is important to recognize the fact that a model is only one of the estimation techniques. This presupposition is not only realistic, it prevents endless experimental use that benefits nobody.

4. Organizational requirements for the use of a software cost model

The application of software cost estimation models is not always successful. Based on our experience and observations in several software departments, we believe that this is partly because some organizations are not aware of the requirements that have to be fulfilled for the successful use of a cost estimation model. Here we describe a number of important organizational requirements that are derived from the use of a model.

(a) The cooperation of software developers

Models estimate the cost and development time of projects that are to be carried out by software developers. Their cooperation in the use of the model is needed for two reasons. Firstly, they should feel committed to the final estimate: they will have to realize it. If this estimate is partly based on a model, they will not feel committed to the estimate if they had no influence on the outcome. Secondly their knowledge is necessary to evaluate the input variables for a proposed project.

Software developers are sceptical about the value of software cost models; obtaining their cooperation is therefore not easy. Scepticism is increased if the model is presented as the final solution to all estimating problems or as an instrument to review development teams. The only way to convince developers of the usability of a model is by achieving results that support the developers in the control of their projects.

Table 3
Distribution of the effort involved in the introduction of a model.

Percentage of effort	Year 1	Year 2	Year 3
introduction of the model	68	42	40
improvement of information supply	0	30	10
development of project database	22	25	10
other control activities	10	13	40
Total	100	100	100

(b) The availability of manpower

The introduction of a model requires an effort that should not be underestimated, especially if the introduction of a model requires an improvement of the information supply in the organization; this is usually the case. *Table 3* shows the distribution of effort for a major software development department during the introduction of a cost model [8]. During the first two years two people were involved its introduction, during the third year, only one was involved.

Table 3 shows clearly that the introduction of the model had important side effects and that the effort shifted from the actual introduction to other activities related to the control of software projects.

With respect to the work to be done, the employees that must introduce the model should have the following capacities: (1) knowledge of the software to be developed, (2) interest in the control of software development, and (3) social skills. Knowledge of the software is necessary to make it easy to interact with the software developers and to evaluate the model input variables. Social skills are necessary to interact with all the parties involved, such as software development, marketing, management and, administration. Taking the requirements and the manpower into consideration, it is obvious that the introduction of a model cannot be left to somebody who "just happens to be available anyway" or "can do it in his spare time".

We recommend that a model be introduced by two or three people. This is because there is much work to be done and individuals that have all the required capabilities will be a scarce resource. Continuity in the use of the model is another reason to spread the knowledge on the use of a model over a number of people.

(c) The commitment of management

The use of a model costs money, takes time, and may require adjustments in working practice. Money will have to be spent to acquire a model and to obtain the relevant training. The out-of-pocket costs are relatively small compared to the introduction costs, which comprise the effort that has to be made on the calibration of the model. The adjustments in working practice will concentrate on guidelines for estimation and obtaining information.

These requirements in their turn, demand the commitment of management to the introduction of a model. Commitment is preceded by the recognition of the problems involved in controlling software projects. In some places, this recognition must be preceded by a major overrun in an important software development project.

(d) Estimation guidelines

The use of a model requires some estimation guidelines, three of which will be discussed. First, in order to make an estimate using a model, one should give the estimators enough time. Second, the required information should be available to the estimators. If this information is not available, the estimate will be less accurate or even impossible to determine. The third guideline involves the distinction between a technical and a commercial estimation. Commercial arguments play their role in software cost estimation, e.g. "If our price exceeds 2 million, we will never get the order" or "If they allow us to develop the first prototype, they will depend on us for the remainder of the project".

We recommend that the commercial and technical estimation are separated. A technical estimation must be based on a specification of the product to be developed and therefore a model could be a useful tool to arrive at the technical estimate. If this estimate is commercially unacceptable, one should adapt the specifications or reconsider the presuppositions and re-estimate the project or take a loss for business reasons. However, the input of a model should not be manipulated in order to arrive at a predefined commercial estimate.

(e) Adequate information supply

It is recognized, both in theory and practice, that data on current and past projects are necessary to control future software development. Some people think that the introduction of a model relieves them of the obligation to collect and update data. The opposite is true; the use of a model only increases the need to collect data. The data are needed to calibrate and use the model. To state this clearly: the data that are needed to use a model are also needed to control projects. The question is not "What effort should be made to collect the extra data?", but "How can anyone attempt to control projects without these data?". How can you control a project if, for instance, you do not know how many hours have been spent on it? A recent survey shows however, that 50 per cent of the respondents did not record any project data on current or completed projects [6,11].

Some other data have to be recorded besides the effort and development time expended on the project. One can think of the size (for instance in lines of code or function points), the kind of application and the complexity. The data can be recorded in terms of the input variables and can also be used to support an estimation based on the analogy or expert method. As such, the data may be gathered as a result of the use of a model, it is however more widely usable.

The importance of an adequate information supply to control software projects exceeds the importance of the model use. If the information system is improved as a consequence of the use of a model, it has made an important contribution.

4. Conclusions and recommendations

This paper describes how a software cost estimation model can contribute to the improvement of the software cost estimation process. Our first recommendation is: estimate a proposed project by alternative methods. The methods described in this paper are the expert method, the analogy method, and the use of models. If alternative methods are used, the weak points of one method can be compensated by the strong points of another. If the estimates generated by alternative methods agree, the estimate is accepted. If not, the reasons for the difference must be determined, followed by another series of estimates. Experience in using this approach shows that differences can usually be explained by the effect of different starting points and different presuppositions of the estimators. Another advantage of this approach is that consensus on the final estimate will increase if there are discussions between the estimators. The use of a model as one of the alternatives has several benefits. Firstly, the model forces one to look at the proposed project from another point of view, i.e. that of the cost drivers. Secondly, the model forces one to systemize the control of software development.

The application of software cost models is not always successful in practice. We believe that one of the reasons for this is that organizations are not aware of the requirements that must be fulfilled in order to use a model properly. The following are important requirements:
- cooperation of software developers
- availability of manpower to introduce and use the model
- commitment of management
- estimation guidelines to allow proper use of the chosen estimation methods
- adequate information supply.

We are aware that it is not easy to fulfil these requirements. However, if they are fulfilled, there are benefits in the control of software projects in general. It is even beneficial if one is not using a software cost estimation model. If the use of a model draws attention of an organization to the requirements, the model has already made an important contribution even before it is actually used.

References

[1] T.K. Abdel-Hamid, S.E. Madnick, "On the portability of quantitative software estimation models." *Information and Management*, 13, 1-10, 1987.

[2] A.J. Albrecht, J.E. Gaffney, "Software Function, source lines of code, and development effort prediction: a software science validation." *IEEE Transactions on Software Engineering*, volume SE-9, no. 6, 1983.

[3] B.W. Boehm, *Software engineering economics*, Prentice Hall, Englewood Cliffs, 1981.

[4] A.F. Case, *Information system development*, Prentice Hall, Englewood Cliffs, 1986.

[5] A.M.E. Cuelenaere, M.J.I..M van Genuchten, and F.J. Heemstra, "Calibrating a software cost estimation model: why and how". *Information and Software Technology*, volume 29, no. 10, December 1987.

[6] F.J. Heemstra, R. Kusters, "Controlling software development costs", Proceedings of the conference on Organization and Information Systems, pp 652-664, Bled, Yugoslavia, September 13-15, 1989.

[7] C.F. Kemerer, "An empirical validation of software cost estimation models." *Communications of the ACM*, volume 30, no. 5, May 1987.

[8] H. Koolen, Report of the introduction of PRICE S", report Hollandse Signaal Apparaten, January 1988 (in Dutch).

[9] R. Kusters, M.J.I.M. van Genuchten, F.J. Heemstra, "Are software cost estimation models accurate?", *Information and Software Technology*, Volume 32, no. 3, April 1990.

[10] S.N. Mohanty, "Software cost estimation: present and future.", *Software – practice and experience*, pp. 103-121, 1981.

[11] W.J.A.M. Siskens, F.J. Heemstra, H. van der Stelt, "Cost control in automation projects; a survey", *Informatie*, Volume 31, January 1989 (in Dutch).