Software cost estimation and control : lessons learned

SOFTWARE COST ESTIMATION AND CONTROL ; LESSONS LEARNED

Dr.ir. Fred J. Heemstra
Dr. Rob J. Kusters
University of Technology
P.O. Box 513
5600 MB Eindhoven
tel: 31 40 472290

1 Introduction

The research group "Control and Estimation of Automation Projects" of the Eindhoven University of Technology has been active in research on software cost estimation for several years now. This resulted in two Ph.D studies and several Master Studies. Research projects in industry are also carried out on a regular basis. After years of research the question that cropped up was: What have we learned after all ? Is it possible to formulate some general conclusions, Are we able to indicate the key problems or even the key solutions for software cost estimation (SCE). Our aim is to give some critical reflections in this paper.

We start with the formulation of five objectives of software cost estimation and control in section 2 . For each objective several critical remarks, that is to say shortcomings and advices for improvements, are given in section 3. By confronting objectives and evaluations our view on the current state of SCE is presented. One main conclusion is reserved for the last section, namely that SCE required active involvement by each organization.

2. Objectives of estimation

It is possible to sum up without much difficulty a list of objectives to achieve with an estimation. We will concentrate on what in our view are the five major goals.

1. **Insight in order to forecast.**
   From this perspective an objective of estimation is obtaining insight in the development process, in the product to be developed and in the means of development. This insight informs the developers about the rules in software development, the influence of cost drivers, the effects of productivity

improvement plans, etc. It is an important precondition for predicting the required development effort and time. Estimation is a simplification of reality. Because the process of software development is too complicated to decompose into detailed and predictable routine activities we have to make use of simplified descriptive models for estimation.

2. **A target in order to control.**
   From this perspective an estimation is regarded as a budget. An estimation has an aspect of setting a deadline. This has nothing to do with calculating or estimating start and end dates. E.g. a software developer will never be satisfied about his design. For him it is always possible to improve it and he is willing to do so as long as there is time. Setting deadlines will stop this activity.

3. **Commitment in order to accept.**
   An objective is to acquire shared understanding. Involving developers in the estimation process will contribute more towards motivating them to realizing their "own" estimate more than can be expected from estimates that are forced upon them.

4. **Communication in order to open up the discussion.**
   An estimation is a means of communicating one's ideas. It will as a result be easier to coordinate activities. Also, an open communication among the members of the development team and between client and development team leads to contentedness among client and users and a higher staff morale.

5. **Pre-requisite in order to decide go nogo.**
   An estimation of development effort and duration is one of the inputs for the go nogo decision of an automation project. Other inputs are estimation of maintenance and exploitation effort, cost benefit estimations etc. In this paper we will not concentrate on this objective. At the ISPA conference we presented a paper on this topic.

We want to emphasize that an estimation is much more than an accurate prediction. It has also to do with a good relation with satisfied clients, a satisfied project team, acquired insight.

3 Lessons learned

As we stated in the introduction, the lessons we learned from research in the field of SCE are arranged according the above mentioned objectives. The lessons are presented as statements. Before elaborating on the statements we want to start with presenting the complete overview.

a. Objective : Insight in order to forecast

Statements :
- SCE is an information problem;
- SCE is impossible without past project data;
- For SCE data collection of the ongoing project is required
- SCE requires what-if analyses;
- An estimation without a risk analysis has limited value;
- An estimation must be based on more than one estimation technique;

b. Objective : A target in order to control

Statements :
- SCE must be a part of the control cycle;
- SCE is an ongoing activity during the development process;
- For SCE holds : measurement enhances knowledge;
- An estimation must not only be focused on effort, time and resources but also on quality;
- SCE requires clear descriptions of the software requirements;
- Estimations without margins are not estimations;
- Estimating is a matter of clear definitions and agreements;

c. Objective : Commitment in order to accept

Statements :
- Involvement in estimating by the developers is a necessity;
- Data collection of past projects must be based on the closed loop control;

d. Objective : Communication in order to make open to discussion

Statements :
- A clear description of definitions, agreements, starting points and user requirements facilitates communication among participants;

e. Objective : Pre-requisite in order to decide go nogo

statements :
- Estimation of software development is just one piece of the go nogo puzzle.

3.1 Lessons learned about "insight in order to forecast"

SCE is an information problem.
A prerequisite for estimating is insight in the things one wants to do, the way one wants to do it and the people, methods and tools one wants to use. Without such information estimation becomes "guesstimating". The accuracy of the estimation increases if the information is more complete and more reliable. The required information can be divided into:

- Product information;
  for example users requirements, global design;
- Process information;
  for example the way the software will be realized, the development work will be organized and responsibilities and authorities are divided.
- Development means information;
  for example people that are involved in the development process, e.g. experience and education level.
- Past project and product information;
  it concerns facts, knowledge and experience gained during the execution of projects in the past.
- Progress information of the ongoing project;
  this kind of information is important in case of adapting the estimations during project execution.

The relationship between the different kind of information is visualized in figure 1.

SCE is impossible without past project data.
As we said before, past project information is required to estimate. How can we predict our future if we don't even know our past. It doesn't matter which estimation technique is used, they all are based on data and experiences of completed projects.

The consequences for a software development organization/department is collecting and recording data. A remarkable conclusion of an extensive field study on software cost control was however that only few organizations collect data on past projects in a structural way (Siskens and Heemstra, 1989). For another study it took us a lot of time finding some organization with sufficient and high quality data to participate in a research on past project data. Even a lot of software houses with software engineering as their core business didn't meet our demands.

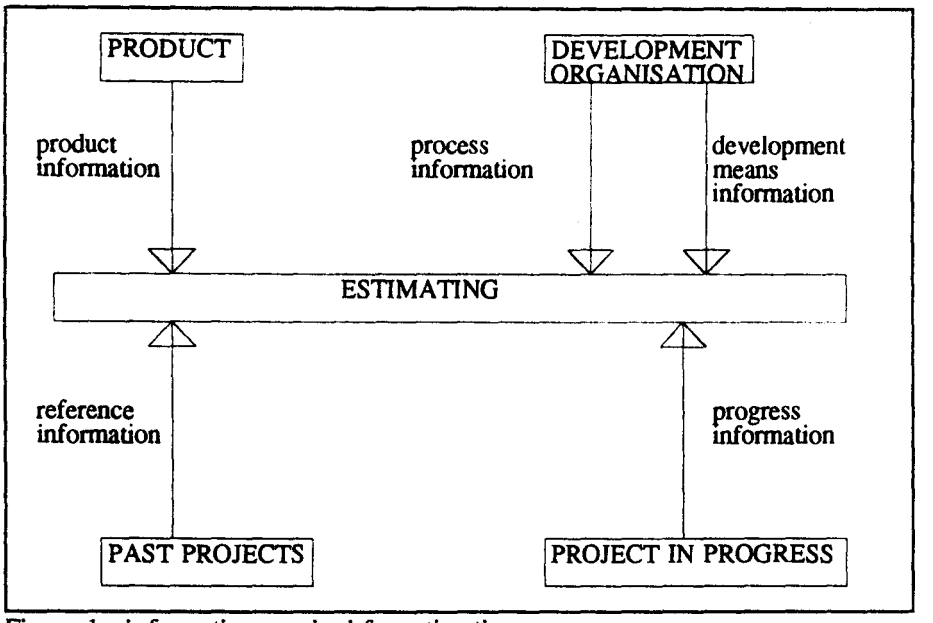

Figure 1 : information required for estimating

Table 1 : Objects and attributes of Noth's database

PRODUCTS		
description and classification size resource use cost lead time	risk estimation and evaluation change requests and failures problem reports reference to persons and functions involved reference to related products	
RESOURCES		
- PERSONNEL : name profile function in project department problem involvement productivity absence due to illness product involvement reference to job description	- TEAM : name profile members fluctuations absence due to illness productivity product involvement problem involvement	- HARDWARE/SOFTWARE : name classification function applicability supplier reference to buy decision problem involvement
ORGANIZATION		
place within organization function product involvement	problem involvement experience political interest preferences	

The main question is : what kind of past project information is relevant to collect. It is difficult to give a closely-reasoned answer and mostly the answer starts with "it depends". A cost driver like the use of 4GL tools can have much influence on development effort in organization A but is of no value for organization B developing software without 4GL tools. The adagium "local for local" is often and quite rightly used in this connection.

A great deal of the type of information is however organization independent, e.g. general project information (project name, name customer, lead time, effort). Table 1 shows an example of type of information to collect and record (Noth, 1987).

For SCE data collection of the ongoing project is required

Insight in the development process can be achieved in several ways. One way we just mentioned, namely collecting and analyzing past project data. Another way is measuring on ongoing projects. Genuchten et al. (1990) give an example how measurement at the activity level of software development increased the insight and improved the control and estimations of future activities. The measuring-instrument was extremely simple while the achieved advantages were extremely high. Figure 2 shows the measuring-instrument.

The instrument was used for thousands of activities and gave the concerned development organization insight both in the differences between estimations and reality and in the reasons for these differences. It turned out to be that maintenance activities disturbed the development process continuously. The organization was not aware of the seriousness of this problem. The obtained insight lead to direct control measures and to estimation improvements.

Table 2 : A measurement-instrument to obtain insight into the development process

	Estimation	reality	differences
Duration			
Effort			
Starting date			
Ending date			

Collecting data in this way has several advantages. Without much effort - approximately 15 minutes for an activity of 40 hours - the organization gets insight into the quality of the estimations. The feedback on estimation is quick and can be used immediately for re-estimating.

The same as applied to past project data collection holds for measuring on ongoing projects. Despite the importance of getting insight in the development process by measuring, only few organizations spend effort on this activity in a structural way.

SCE requires what-if analyses

The estimation quality increases as the information on the product, development process and development means is more complete and reliable and if the development organization has sufficient past project and progress information at its disposal. It often happens however that no clear idea exists of the software to be realized, of the way it should be realized and of the means required. Sometimes the lack on information has to do with the specific nature of the project, for example developing innovative software with unexperienced users and developers unknown with the application domain. Lack of information usually exists in the early phases of development in which only the global outlines of the software are known. Important estimation information is missing which results in uncertainty.

In such circumstances an estimator is not interested in fixed estimations. It is much more important for estimator and project management to know how sensitive an estimation is to changing circumstances. For example: what is the effect of two more analysts on duration; what is the effect of time compression on development costs; what is the effect of under- or overestimation of software complexity on the estimation, etc. In dealing thus with the estimation problem, management gets a better grip on possible solutions and is able to decide well-balanced. Furthermore there now is a suitable basis for project progress control. If an estimation turns out to be extremely sensitive for value changes of one or more cost drivers, project management is warned to pay attention to these cost drivers during project execution.

In our research we discovered that what-if analyses are rarely used in software cost estimation in practice. For instance only few software cost models have sufficient possibilities for what-if analysis.

An estimation without a risk analysis has limited value

Estimating development effort and duration under uncertain circumstances has to include estimating risks. For a software development organization / department it is important to recognise possible risk factors, the effect of these factors, how to handle the risks etc. After answering such questions estimations of effort and duration can be interpreted better. An estimated effort of 100 man months has a completely different meaning for a high or low risk project.

SCE Risk estimating consists of:

- Risk-identification
  Which factors are have a possible impact on project success. Risk factors can be identified by using checklist and reviews of completed and comparable projects.
- Risk-valuation
  Define the effect of the risk factors. This effect can be found by combining the probability of the occurrence with the impact if it occurs.
- Risk-structuring
  Ranking the risk factors and defining the mutual dependencies. The existence of risk A can result in other risks.
- Risk-reduction
  Defining the possibilities of reducing the effects of risks.

Existing techniques for risk-analysis can be used especially for risk-identification and valuation. All such techniques have more or less the same approach. On the basis of a checklist questions must be answered regarding (see for instance Rijsenbrij et. al. 1990):

1. Project size.
   For example: is the development organization familiar with projects of the estimated size ?
2. Automation level.
   For example: What is the education and experience level of the development organization with regard to the software to develop ?
3. Technology.
   For example: how familiar is the development team, the user organization and the software supplier with the technology chosen for the project ?
4. Project organization.
   For example: how well is the project organized ?

5. Project environment.
   For example: under which circumstances is the project executed?

By answering the questions an indication / estimation is made of the project risks.

The combination of effort and duration estimating and risk estimating is not common property in practice.

An estimation must be based on more than one estimation technique
An estimator can choose from a great number of estimation models. Practically no model is able to estimate the development reasonably accurate (Abdel-Hamid en Madnick 1987). This claim is supported by unequivocal test results in literature (Mohanty, 1981) (Kemerer, 1987) (Rubin, 1985) (Kusters et al. 1991). All researches give an identical judgment: models are a useful support only if calibrated, that is to say adapted for the organization in question. A necessary condition is the availability of an locally collected extensive data set of past projects. As we mentioned earlier, a lot of organizations are lacking such data. From that perspective models have limited value in estimating accurate predictions, the more since a lot of models have no or limited possibilities for calibration. From that perspective it is sensible for an organization to base an estimation on more than one estimation technique. A combination of expert judgements, analogy-based and model estimations springs to mind. The reliability of the estimation is indicated by the differences between the obtained estimations.

Some concluding remarks on software cost estimation models. Why should organizations invest in poorly performing estimation models, what benefits can they expect from these models? The value of a model can be summarized as:

- a model provides management with a set of standards, metrics and directives;
- a model draws one's attention to important cost drivers, sensitiveness of the software to variations in cost driver's values. In this the model functions as a kind of checklist;
- a model offers management a quick estimate that can be used as a second opinion;
- a model is a starting point of an awakening process and is a stimulus to think seriously on SCE;
- historical estimations from one model within the same organization offer a frame of reference.

3.2 Lessons learned about "Defining a target in order to control"

SCE must be a part of the control cycle

Estimating should not be an isolated activity but a part of a project control strategy. The role of estimating within project control will be illustrated by the control cycle (figure 2).

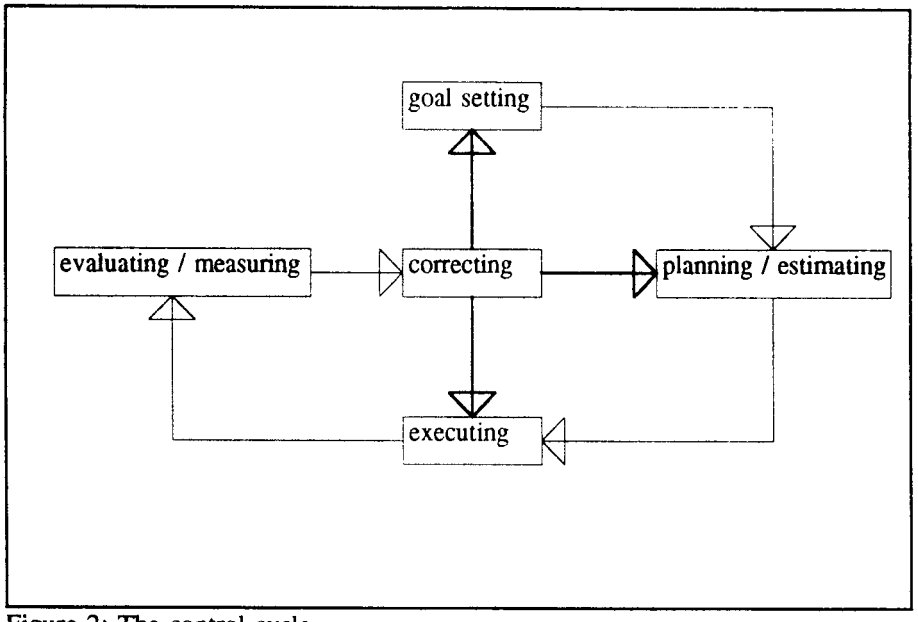

Figure 2: The control cycle

The goals of the project are the starting point of control. Goals or the project results are formulated in terms of quality and functionality. Sometimes it is possible to formulate the results concretely, often however it is not. Software quality in particular is hard to formulate and is often neglected. Project planning and estimating is based on the formulated results. The project is divided into phases, activities and tasks. Execution time, costs and capacities are estimated. It is clear: the better the project results can be formulated, the better planning and estimating can be done. Planning and estimation are the basis for execution. An important part of control is monitoring and evaluation the work. It is examined if the right software is made or if it has been made the right way. In the previous section a measurement instrument was presented to obtain insight in the progress of an ongoing project. Corrections are necessary in cases of differences between plan and reality. Corrections can lead to new and/or adapted targets which have to be replanned and re-estimated. Adapted plans and estimations in their turn result in rearrangements in executions that must be evaluated and monitored. And again corrections are possible. The subsequent activities are closely interrelated and are

performed more than once during project execution, starting from vague to exact and from global to more detailed at each iteration.

SCE is an ongoing activity during the development process
This statement is a logical result of the previous statement. Like we stated SCE is not an unique activity during project execution, but an evolving activity. The control cycle must be gone through more than once during project execution. How often depends on product characteristics such as size and complexity and on the degree of uncertainty of the project result. It is not useful to keeps one's finger on the pulse while constructing simple, well known software. In general however it applies that one cannot suffice with a unique, static estimation (van Vliet, 1988). It doesn't fit in with the dynamic characteristics of software development.

Estimation is like aiming for a moving target. A flexible use of a set of estimation techniques is required. More (reliable) data and details come available during project execution. As a result the estimation approach must be adapted to the changing circumstances and the new information. A combination of successively use of the Wide-Band-Delphi approach, an analogy approach, Function Point Analysis, Estimacs, and finally COCOMO during the project is an possible option. As time goes by the insight in the product to be developed increases and makes use of more formalized estimation techniques possible.

It often happens that software estimation is regarded as an isolated activity of project control. Estimating, measuring, re-estimating etc. is often done at an ad hoc basis and not an integrated part of software project management.

For SCE holds : measurement enhances knowledge
In the previous section we emphasised that measuring is a necessary condition for effective software control. Without data it is hard or even impossible to answer annoying questions like:

- how reliable are my estimations. What size are over- and underestimations;
- what is the reason for differences between plan and reality;
- what is the productivity of a development team or a specific developer;
- what is the effect of a case-tool on development effort and duration;
- what is the effect of reuse on productivity;
- which five cost drivers are the most dominant in my organisation;
- what is the relation between size and effort in my organization;
- etc.

Answers to such questions are strictly speaking indispensable for credible estimating. How can an organization estimate if it is unable to produce the elementary input. The only way an organization has at its disposal is measuring in order to produce eventually reference data. From our experience we know that the previous questions are indeed annoying for most organizations and project management becomes nervous confronted with such questions. From that perspective estimation and control are based strongly on intuition in stead of being based on ratio.

An estimate must be focused on effort, time and resources only, but also on quality
Effort, duration, capacity and quality are important aspects to control and estimate. This means continuously balancing between quality on the one hand and effort, duration and capacity on the other hand. Extra quality means more money and more time, shortening of development duration can not be done without adverse effects. The effects of time compression are mostly reduced quality and/or an extra increase in price. In figure 3 the relation is shown.

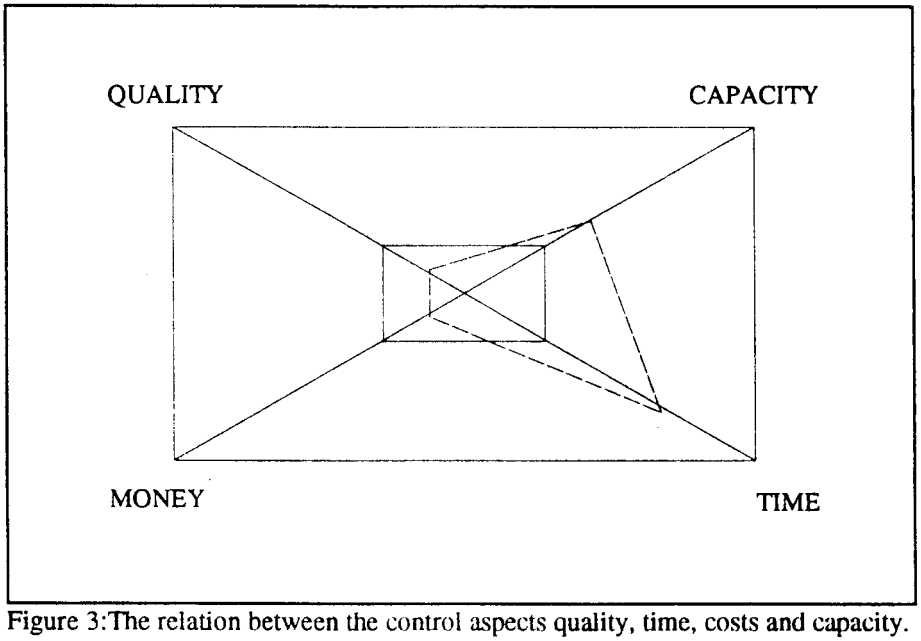

Figure 3:The relation between the control aspects quality, time, costs and capacity.

Clear agreements on time and money are made most of the time. The software must be delivered at the end of january 1993 and the price is 2 million guilders. It is equally important to agree on what has to be ready and who has to do what and when. Without any attention for the these relations, control and estimating has little value. The project result or the norm to compare the developed software with is missing or at least open to misinterpretation.

Successfully estimating means a well-balanced attention for the mentioned control aspects. A "sound" planning and estimation must be based on certain, unequivocal and well formulated project results. This however turns out to be an utopia in practice. Often the estimating emphasis is on time and money first while quality is neglected.

estimating requires clear descriptions of the software requirements
This statement has been mentioned at different places in the paper. Because it is one of the most important foundations of estimating, we want to formulate it as an independent statement. The reliability of an estimation is directly linked to the clarity of what has to be developed, what capacities are at one's disposal and when and for how long they are available: it is linked to the clearness of the Target. Without a clear target control becomes steering a car without a steering wheel. Clearness is not restricted to a precise description of the software requirements (that is to say the target) alone, it also means insight in the uncertainties of the requirements. With this insight a discussion on possible and impossible estimates can start. Despite the lack of clearness it often happens that (supposedly) clear estimations are made.

Estimations without margins are not estimations
An estimator has often to do with ill-structured problems, with different and conflicting goals (cost minimization, quality maximization, minimization of duration, optimal use of manpower), with many participants (principal, user, project manager, developers, etc. Exact estimations like "duration is 321 months, effort is 2031 man months, etc" are of little value. Such exact figures are not in accordance with the nature of the problem.

Software development and estimation is characterized by uncertainty due to unclear and ambiguous targets. That is the reason why each estimation must contain an indication of the level of uncertainty. In figure 4 it is illustrated that several levels of uncertainties exits. The way management has to deal with controlling and estimating software projects has to do with such levels. High uncertainty means low project controllability and high estimation accuracy. Exact estimations don't match to this situation. Estimations with margins are more appropriated. The higher the level of uncertainty the larger the margins.

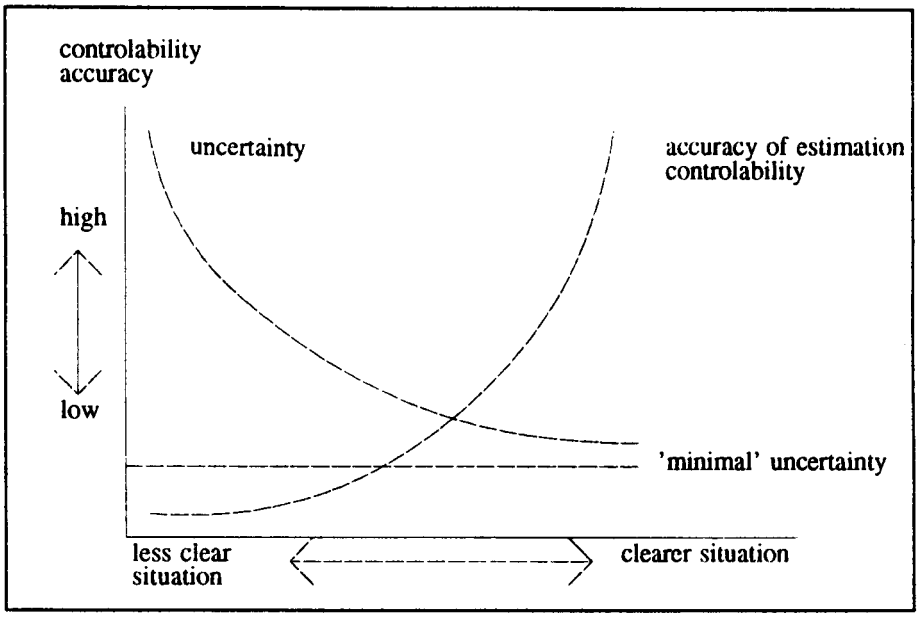

Figure 4: The relation between clearness and controlability of the project and accuracy of the estimation

Estimating is a matter of clear definitions and agreements

Condition that have to be fulfilled for successful software cost estimating are the existence of agreements, clear definitions, standards on the one hand and accepting and adherence them on the other. Such agreements and directives can refer to:

- How many times has an estimation to be made during project execution.
  For example: 5 times for each project with more than 12 man months required effort;
- In which stage of project execution is estimation required.
  For example: at the start of the feasibility study and the requirements phase, after completion of the global design etc;
- Who is involved in estimating.
  For example: the project manager, the principal, representatives of the development team;
- What has to be estimated.
  For example: all development activities that refer to the phases preliminary investigation, specification, design etc. or all activities inclusive training, documentation, conversion etc.
- What are the outcomes of an estimation.
  For example: costs in guilders, effort in man months and duration in months.

- Which factors can be regarded as the most important cost drivers and must be the basis for data collection of past projects.
  For example the factors size, required reliability, application type, personnel quality etc.
- Which metrics must be used.
  For example: size is expressed in function points which are converted to number of lines of code (exclusive comment- en blank lines).

The result is a comprehensive enumeration of metrics and standards used during software development within an organization. It is important that the chosen metrics are applied consistently. This will result in a consistent set of measured values in the long run.

Such a list of agreements etc. is seldom used in practice. From that perspective software engineering has a long way to go towards a full-grown engineering discipline in which standards, norms and definitions are used as a matter of course.

3.3 Lessons learned : commitment an indispensable prerequisite for successful cost estimation

Software cost estimation is no problem that can be solved with an algorithm. Like we mentioned before the goal of estimating is more than a just a prediction of effort, costs and time expressed in numbers. Estimating requires more than numerical algorithms. Human / psychological factors have an important impact on development effort and duration. Examples of these factors are:

. experience,
. capability / skill,
. training-level,
. turnover of labour staff,
. working environment,
. etc.

The importance of these factors is not always recognized in practice. Estimating is often looked at as a technical problem that demands a technical solution. One has no eye for a behavioral science approach. The neglect of this aspect can be shown by the fact that only few estimation models take into account the importance of human aspects. However, several well-known researches clearly show that the effect

of such factors, in particular on costs and duration, can not be denied (Boehm 1981, Walston en Felix 1977, Mizuno 1983, Jensen 1984 en Jones 1986).

The human aspects are related in a complex way. For example: the influence of a high education level and great experience on work performance is annulled by unpleasant work circumstances or by the indifference or even opposition of colleagues. Rivalry, a hostile sphere, internal political intriguing are fatal for the success of an software project and its estimation. Jones (Jones 1986) claims that 10 to 15% of the software projects fail because of these reasons. Matsumoto (1987) claims that an ideal working environment is the most important condition for a successful software development project. An ideal working environment is created by paying attention to working space, management style, organizational culture, rate of pay, career planning, training and education, availability of automated tools and the presence of a clear development strategy.

This short excursion into the human aspects of software engineering shows the complexity of these interacting effects on a software cost estimation and also show the limited view of estimation models. Estimating is a managerial problem of creating first of all ideal working conditions and secondly getting grip on the effects of "soft" cost drivers on work performance e.g. development time and effort.
The way the estimations were achieved are in the end maybe as important as the estimations themselves. The preliminary discussions, the collection of relevant information, attention for an ideal working environment etc. encourage a conscious dealing with the problems of software cost estimation and initiate awakening- and learning process in an organization.

Involvement in estimating by the developers is a necessity
Commitment of the development team members is an important aspect in estimating. Involvement in estimating becomes more important as uncertainty and fuzziness in the software development increases. An organization that develops software in uncertain circumstances will have to use professional, high qualified personal. In general professional developers have specific characteristics (Grinwis 1989). For example:

- individualism in the way of performing their professional activities is of great importance;
- a high qualified result is considered important;
- the professional wants involvement by the management in the way of working and in the required results;

- professionals don't like any interruption by management when doing their job. They prefer high independence.

From this perspective it is not sensible to confront developers with an estimation and plan imposed by the management. Plan and estimation must be realized in joint discussion. Added advantages are that the advice of experts/specialists is taken into account and the real executors are consulted about the estimation problematic. The main benefit however is that it results in higher involvement and in the end in a higher commitment, indispensable for a successful project.

Data collection of past projects must be based on the closed loop control
Another reason why commitment is important is based on the principle of closed loop control (Bemelmans, 1991). Starting-point of this principle is that the reliability of data depends in the end on the quality of the input data, that is to say of the collected and recorded data. To maximize the quality "suppliers" of data have to know why they have to supply the data, for what purpose the data are used, that is to say they have to be aware of the interest of their data supply. Awareness alone is not sufficient for motivation purposes. They also need quick feed-back of the supplied data as control information. Close loop control fits directly with software cost estimation. Collecting and recording of project data asks for extra effort of the developers. They are willing and motivated to spend extra effort, if they receive as compensation relevant reference information. The reference information must be directly applicable as input information for estimating and project progress control. Like we mentioned several times: creating the right conditions for successful estimating is a management task and is, in our opinion a very relevant part of software cost estimation.

3.4 Lessons learned : Estimations as a mean for Communication

The importance of measuring has been explained extensively. A final remark with regard to communication is required. Measuring is of course not a goal in itself. Our experiences with using the measurement instruments mentioned before were that the developers themselves started a discussion using the measurements as starting-point. Discussions were held on facts and not on myths. Insight in the development process was an important motivation for the developers to start the discussion.

A clear descriptions of definitions, agreements, starting point and user requirements facilitate the communication among the participants
Discussions are a much easier and have more meaning is they are based on facts and are understandable because the same language has been used. An organization that has such a set of definitions, metrics, directives at it's disposal has a excellent point of departure to make estimating successfully. Owning definitions, etc. is not sufficient. The organization must be willing to use these consistently. Some pressure by management is required. Also training, information and the availability of consultation whenever problems arise are necessary. Just like any other engineering discipline software engineering must learn to work according to standards and norms.

4. Estimating: you have to do it yourself in the end !

Despite the lessons learned and all the advices mentioned before, despite the availability of an extensive set of estimation techniques and models, software cost estimation is for 90% an activity that must be initiated, carried out and done by the organization itself. It is for example insufficient to buy an estimation model. The organization itself has to formulate what kind of information it needs, what relevant data have to be collected and recorded as reference information, which procedures and definitions it wants to use, how estimating must be imbedded in the current project control approach, how software developers can be motivated and committed to estimate according to a prescribed approach and supply project data etc. Estimation models can only be a additional support in this process.

From our critical remarks at the end of the statements mentioned above the conclusion is justified that the control of software engineering activities has a long way to go to become a full-fledged engineering discipline. A lack of history, reference material and experiences, ill-formulated definitions, agreements and specifications, working under high pressure without much attention for the engineering aspect and no or at least little attention for control aspects, an underestimation of human c.q. managerial aspects, etc - that's all together the state of the art at this moment

The development of an estimation strategy requires a substantial sacrifice of the organization. Collecting reference information for example will cost several years. The benefits will surely exceed the costs in the long run.

References

- Abdel-Hamid, T.K., en Madnick, S.E.
  "On the portability of quantitative software estimation models." **Information and Management**, 13, 1-10, 1987.
- Basili, V.R., Rombach, H.D.
  "The Tame Project: towards improvement oriented software environments", **IEEE Trans. Software Eng.**, Vol. SE-14, no. 6, pp 758-773, 1988.
- Bemelmans, T.
  Management Information Systems and Automation, Stenfert Kroese, 1991 (in dutch).
- Boehm, B.W.
  Software Engineering Economics. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.
- Genuchten, M., Heemstra, F., van Lierop, F., Volkers, R.
  "Has someone seen the software already ? (in dutch), In: Informatie (1990)
- Grinwis, P.
  "Controling education departments" **Lecture Education Centre for Industry and Government (in dutch)**, 1989.
- Jensen, R.W.
  "A comparison of the Jensen and COCOMO Schedule and Cost Estimation Models." **Proceedings of the sixth ISPA Conference**, San Fransisco, 1984.
- Jones, C.
  Programming Productivity, McGraw-Hill, 1986.
- Kemerer, C.F.
  "An empirical validation of software cost estimation models." **Communications of the ACM**, volume 30, nr. 5, mei 1987.
- Kusters, R.J., Van Genuchten, M., Heemstra, F.J.
  "Are software cost estimation models accurate ?" **Information and Software technology**, Vol. 32, pp. 187-190, April 1990.
- Matsumoto, E.Y.
  "Approaching productivity and quality in software production - How to manage a software factory." **Proceedings of the information technology payoff, managing productivity and risks**, Diebold research program-europe, Parijs, mei 1987.
- Mizuno, Y.
  "Software quality improvement." **IEEE Computer**, 1983.
- Mohanty, S.N.
  "Software cost estimation: present and future." **Software - practice and**

experience, 1981.

- Noth, T.
  "Unterstutzung des softwareprojectmanagements durch eine Erfahrungs datenbank." **Proceedings Compas '87, Erfolgs faktoren der integrierten Informationsverarbeitung**, AMK Berlijn, Mei 1987.
- Rijsenbrij, D., Bauer, A., Kouwenhoven, H.
  Project diagnosis, Cap gemini Publishing, 1990 (in dutch).
- Rubin, H.A.
  "A comparison of cost estimation tools." **Proceedings of the 8th international conference on software engineering, IEEE**, 1985.
- Siskens, W.J.A.M., Heemstra, F.J., van der Stelt, H.
  "Cost control of automation projects, a field study". **Information**, Volume 31, no. 1, (in dutch) 1989.
- Van Vliet, J.C.
  Software Engineering, Stenfert Kroese, 1988.
- Walston, C.E., en Felix, C.P.
  "A method of programming measurement and estimating." **IBM system journal**, 16, 1977.