A conceptual framework for software cost control and estimation

tu/e
Research Report

Eindhoven
University of Technology
The Netherlands

GRADUATE SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT SCIENCE

A Conceptual Framework for Software Cost Control and Estimation

by
F.J. Heemstra
R.J. Kusters

Report EUT/BDK/32
ISBN 90-6757-032-X
Eindhoven, 1988

INDUSTRIAL ENGINEERING AND
MANAGEMENT SCIENCE

A CONCEPTUAL FRAMEWORK FOR SOFTWARE COST CONTROL AND ESTIMATION

door

F.J. Heemstra
R.J. Kusters

Report EUT/BDK/32
ISBN 90-6757-032-X
Eindhoven, 1988

Eindhoven University of Technology
Department of Industrial Engineering and
Management Science
Eindhoven, Netherlands

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Heemstra, Fred J.

A conceptual framework for software cost control and estimation / Fred J. Heemstra, Rob J. Kusters. - Eindhoven : Eindhoven University of Technology, Department of Industrial Engineering and Management Science, Section of Management Information Systems and Automation, Research group Cost Control of Automation Projects. - (EUT report / Eindhoven University of Technology, Department of Industrial Engineering & Management Science ; BDK/32)
Met lit. opg.
ISBN 90-6757-032-X
SISO 520.9 UDC 657.31:681.3.06
Trefw.: software ; begroten.

1. Introduction

From experience we know that the estimation of development costs of software development is difficult. Projectmanagement has to predict the resources that have to be available to carry out the project. The most important resources for projectmanagement are time, money, manpower, hardware and software tools. In practice it is a rule rather than an exception that after completion of a project the real expenditure (in terms of effort, costs and development time) prove to be twice or even three times as much as was estimated in advance. In the last fifteen years many attempts have been made to change this. This started with a pioneering article of Wolverton (Wolverton 1974), in which he gave a first outline for a method to estimate the development costs of software. The important study of Walston and Felix (1977) followed. They tried to measure the productivity of software development. Furthermore a lot of software cost estimation models have been developed since then (Rubin 1985, Boehm 1981, Heemstra 1987, Noth and Kretzschmar 1984). Despite the abundance of models and articles on this subject, the state of the art of software cost estimation is still in its infancy. Numerous investigations (Kemerer 1987, Miyazaki and Mori 1985, Rubin 1985) confirm this proposition.
An important step in the solution of this problem is a theoretical foundation of the process of software development cost control, especially cost estimation. In this paper we present a conceptual framework for software development cost control and indicate the relation between estimating, project execution, project progress, measuring and correcting. A thread in this article is the strong argumentation in favour of project registration.
In section 2 some fundamentals of software cost control are explained. Special attention is paid to the role of software cost estimation and the importance of evaluating the differences between these predictions and the accompanying realizations in controlling software projects. In section 3 a conceptual framework for software project control is presented in which the fundamental issues, raised in the previous section are studied in greater detail. In section 4 we will give a short overview of the methods one might use in obtaining an estimation and section 5 will be devoted to project registration. In section 6 the advantages of using a database of

completed projects will be expounded and the paper ends with conclusions and some recommendations.

2. Software cost control

In figure 1 the cycle of software cost control of an automation project is presented. From this figure it is possible to see that controlling software projects means:
- describing the results to be achieved.
- agreeing on delivery time, budgets, quality, organization and information (estimate resources),
- measuring the progress of the project,
- comparing agreements and realizations,
- correcting differences between agreements and realizations.

Agreements are based on estimates of time and money, on specifications of quality requirements and on planning and arranging the necessary organization and information.
Starting point in this cycle of software cost control, especially in making a prediction of effort, costs and development time is the description of the system to be developed. At the beginning of the project this won't be more than a rough description. As project execution progresses, the understanding of the desired product increases; it now becomes possible to make a more detailed and more accurate description.

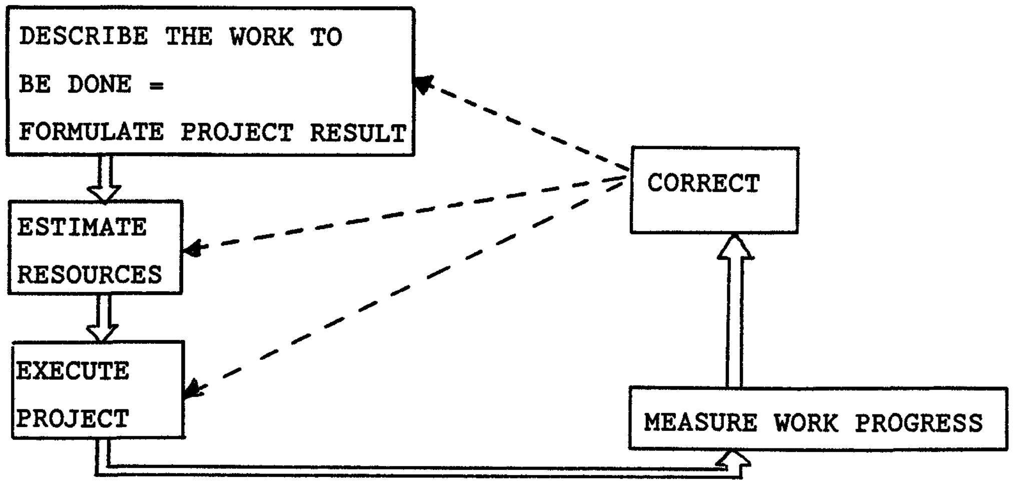

Figure 1: The cycle of software cost control.

In describing the work to be done a distinction has to be made between

- a description of the product (in terms of modules, documentation, designs etc.);
- a description of the project (in terms of phases, activities etc.).

The description of the product has to contain a description of the functionality of the desired system, in other words what kind of information the system has to produce. Besides this functional description a description of how well these functions have to be performed has to be made (Bemelmans 1987). For example the frequency, reliability, response time, amount of detail etc. have to be defined. More than once these conditions are not well, or sometimes not at all formulated. In the event of a threatening overshoot of the estimated effort, costs and development time projectmanagement often tries to save the project and complete it just in time by neglecting these important quality requirements.
Next to a description of the product, a description of the project has to be made as well. This description of the project indicates which phases are to be distinguished in the project and, for each phase, which activities have to be carried out in what sequence.
Next, when the description of the work to be done is completed and the desired project results have been formulated, it is possible to make an estimate of the required resources. Software cost estimation is not a once-only affaire to be carried out at the beginning of the project only. During project execution the insight in the project and product is growing. Project management is thus more and more able to indicate the necessary activities in the project and to decompose the product in components. This better information will make it possible to make more reliable estimates.
To achieve this, it is an absolute necessity that measurement of the progress of the project takes place. This allows a continuous evaluation of the progress of the project. At regular evaluation points/mile stones in the project, project management has to examine the difference between realized expenditure and the amount of expenditure that had been allowed according to the estimates till that moment. Deviations must be analyzed.
This analysis together with the growing insight and experience, obtained during project execution is the basis for correction. There

are three ways to correct differences between estimation/planning and reality.

1. To adapt the estimation, resulting in exceeding budgets, plans etc.;
2. To adapt the project execution, hoping to make up a great deal of lost time (for example by adding more people to a project);
3. To change the descriptions of the work to be done, because they prove to be unrealistic.

These corrections can only be made in an calculated and accountable way if both the plan/budget (in other words: the norms) and the project data are committed to paper. This means that besides a registration of the estimations, a registration of the project execution has to be made. This emphasizes again the importance of registration.

In this article the emphasis is put in the first place on the control of the aspects time and money. Because costs of personnel is 90% of the total costs in software development in a lot of cases, it is possible to derive an important part of the total costs in this way. However it is possible to distinguish more aspects that have to be controlled while developing software. Wijnen, Storm and Renes (1984) mention the following five aspects:

- Time,
- Money,
- Quality,
- Organization,
- Information.

It is nearly impossible to focus only on the aspects time and money while neglecting the other three. There exists a clear relation between these aspects; in figure 2 this cohesion is illustrated. More pressure on the development time (compression) has a direct effect on the aspects money (more expensive) and/or the aspect quality (lower). Similarly, higher quality means more time and/or more money. In software projects time and money requirements are mostly well defined. With regard to quality this isn't always the case. Often the functionality (<u>what</u> must the system do) is not complete and reliable. Performance (<u>how well</u> must the system do it) is most of the time poorly specified. The result is that when a

project threatens to exceed the arranged budget and delivery time, the quality level sinks deteriorates.
If the demands for the aspects time, money and quality increases simultaneously, the result will be a higher pressure on the organization and on the need for information. Better people, more coordination, stronger management etc. is needed in these situations (Galbraith 1973).

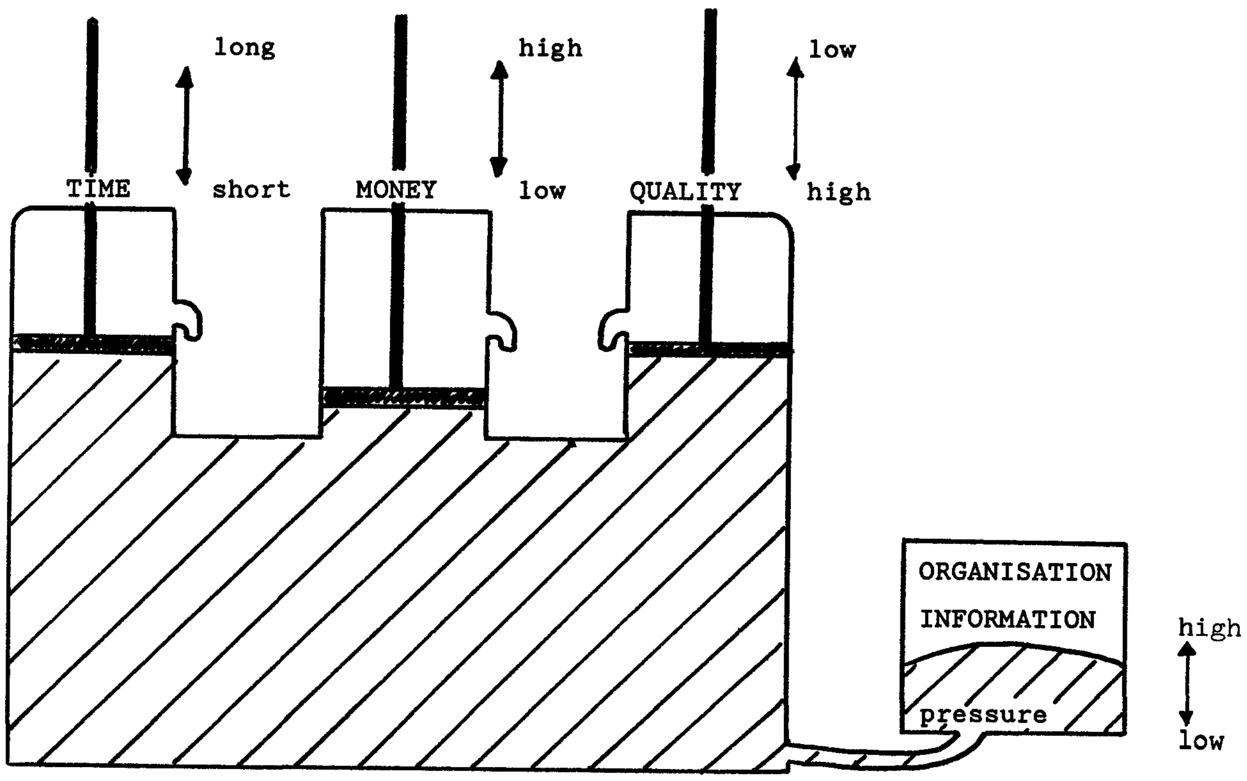

Figure 2: The relation between the aspects time, money, quality, organization and information.

3. A conceptual framework for software control and estimation

In section 2 a general overview of the cycle of software cost control was presented. In this section these ideas will be expounded into a general conceptual model which can be used as a referential framework for the control of software projects. In figure 3 a flow chart is presented which shows all relevant activities connected to

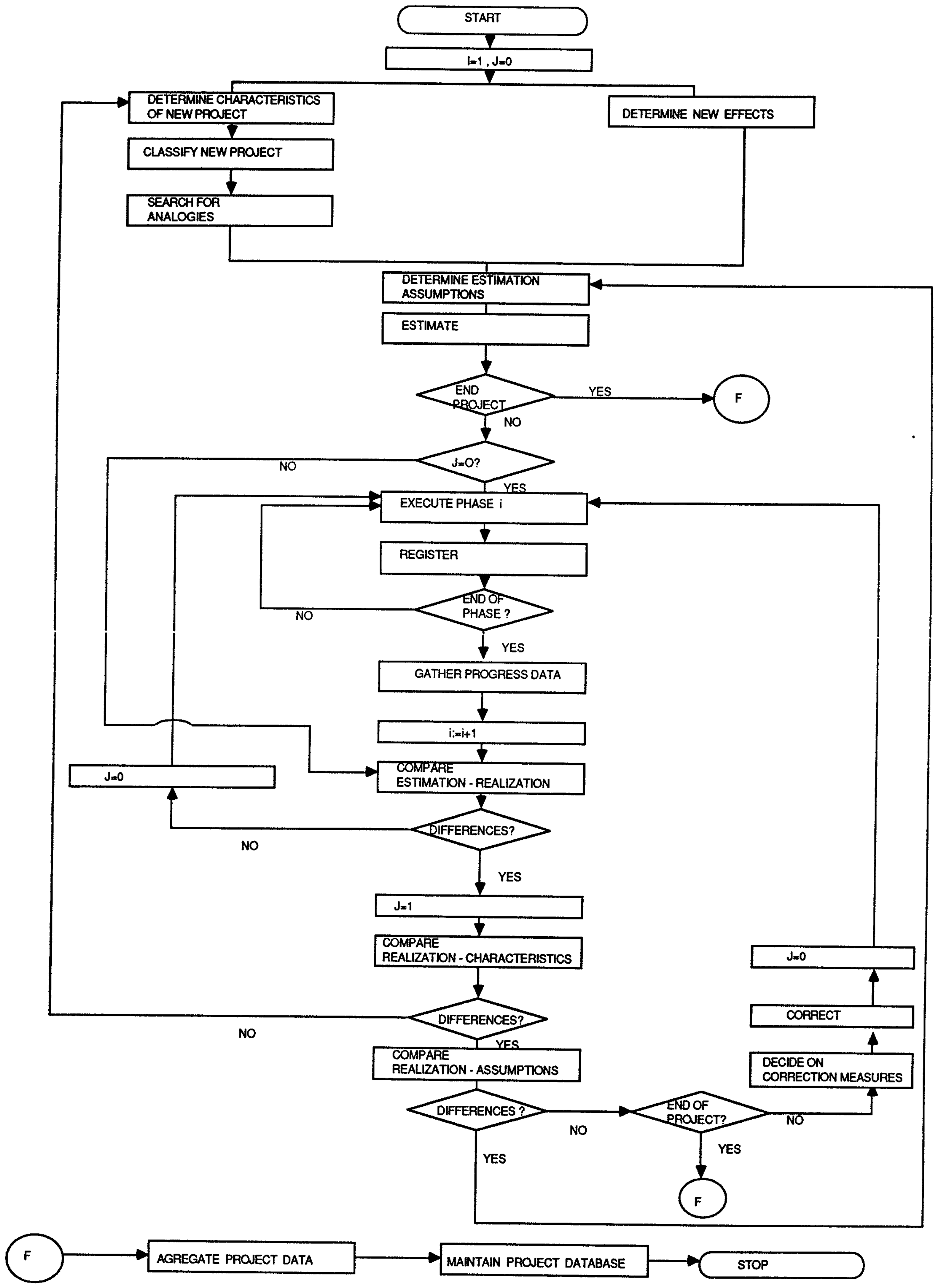

Figure 3: conceptual model

Legenda figure 3:

i indicates the project phase in execution

j = 0 indicates there is a difference between realizations and estimations

the control of software projects and the interdependencies that exist between these activities.
In this section for each activity a short description will be given. Furthermore it will be indicated what kind of input data are necessary and what kind of information is produced. In studying the framework the precedence-analysis technique is used (Lundeberg, Goldkuhl and Nilsson 1982).

Determining relevant characteristics

The characteristics of a new project (see figure 3) are needed for several purposes. Firstly they have to contain a complete list of all the product requirements, both functional and quality. This is necessary in order to get a complete and formal product description which can be used during the rest of the project as a reference.
A second purpose of the characteristics is to form a basis for software cost estimation. In case a parametric model is used, this means that the necessary characteristics will differ according to the model used. Research at the Eindhoven University of Technology has proved that the input for different models vary strongly). Several computerized versions of several models, like SPQR (Jones 1986), Wicomo (Demshki et al. 1982), Estimacs (Computer Associates 1986), Before You Leap (Before You Leap 1986), Bis estimator (BIS/Estimator 1987) and Function Point Analysis (Craenen 1987) were investigated. The variety of the inputs was one of the remarkable results (Genuchten, Heemstra and Kusters 1988).
Lastly, for the purpose of classifying a project other data are needed. The need for data is determined here for an important part by the type classification that is used. Jones (1986) argues that there is a lack of good classification systems for software projects. Research in that direction is urgently needed.
Characteristics of a new projects are derived at the beginning of the project from the user requirements and from a list of selection criteria. This list is based on the method of classification used

and on the method of estimation which is used. However, during the execution of the project it may be necessary to change these characteristics. As the execution of a project progresses, project management obtains more insight in and knowledge of the project. More data becomes available and because of that project management is able to make a more precise characterization of the project. It could also turn out to be impossible to finish the product within the set of budgetary constraints. A possible solution then is to make a change in the functional or in the quality requirements in order to be able to finish some sort of product. And finally it is more the rule than the exception that user requirements change during project execution. Of course this has also an effect on the characteristics of the project. In all cases it has to be possible to adapt the product characteristics on the basis of progress data on the ongoing project. The resulting situation is shown in figure 4.

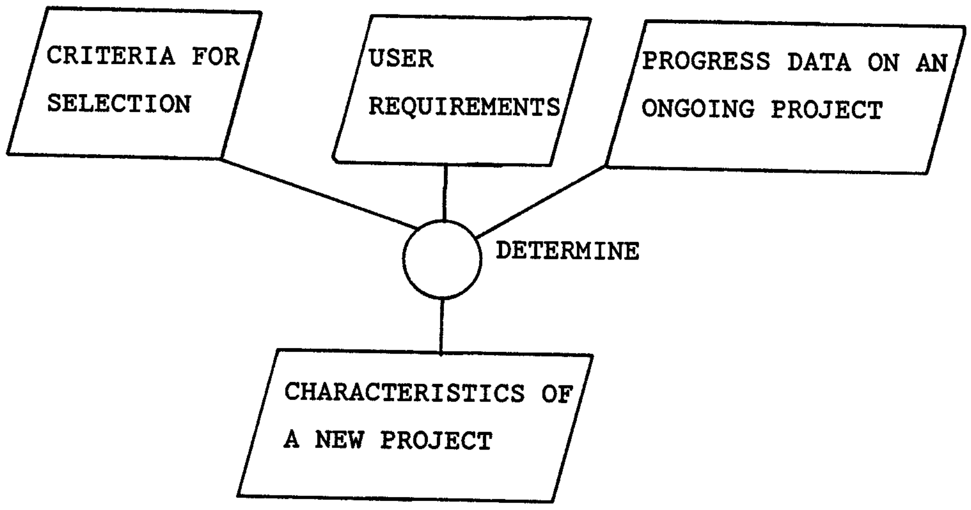
CRITERIA FOR SELECTION
USER REQUIREMENTS
PROGRESS DATA ON AN ONGOING PROJECT
DETERMINE
CHARACTERISTICS OF A NEW PROJECT

Figure 4: Determine relevant characteristics

Classifying the new project

One of the inputs for the activity "search for analogies" (see figure 6) is a classification of the new project. To make such a classification, characteristics of a new project are needed (see also figure 5). Furthermore it is necessary to have a typology of the different classes (for example: typical aspects for a project to develop software for the government are: customer unknown with automation, one application for many users, changes in requirements more then once etc.). The degree of refinement of this classification depends on the number of descriptions of completed projects. It is therefore possible to refine the classification as the number of historical projects grows. In figure 5 it may be seen that one of the inputs of this activity are the project characteristics. It has been demonstrated that these characteristics do not necessarily have to remain fixed during the execution of the project. This also means that the classification activity may have to be carried out repeatedly. This observation is of course valid for the whole conceptual model. Each time an activity input changes this means that the activity will have to be carried out again as well.

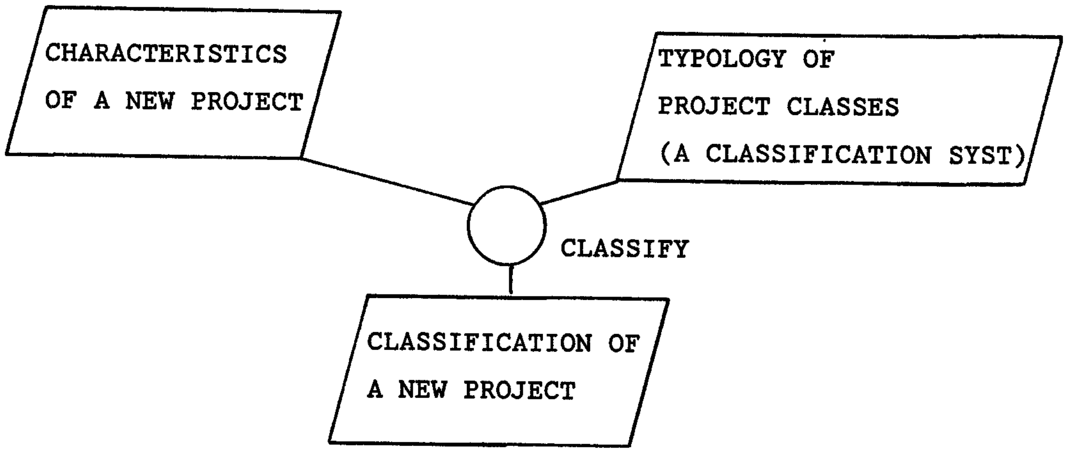
CHARACTERISTICS OF A NEW PROJECT
TYPOLOGY OF PROJECT CLASSES (A CLASSIFICATION SYST)
CLASSIFY
CLASSIFICATION OF A NEW PROJECT

Figure 5: Classify the new project

For an organization it will be wise to make its own classification system and not to rely on the existing systems. Especially in those situations where a specific development organization is specialized in a restricted area of software development this is advisable.
For creating its own classification system the software development organization needs its own database on completed projects and external information on all sorts of existing classification systems. It is necessary to check periodically the validity of the system in order to see if it needs to be corrected.

Searching for analogies

The next step is to use the project classification in order to find analogue, comparable projects in a database of completed projects.

We will then try to find out which parts of a software project are completely new and which parts have similarities with projects or project parts, carried out earlier. In estimating effort, cost and development time for analogue parts it will be possible to use knowledge, facts and experience from completed projects. Estimating based on reasoning by analogy is now a preferable solution. "Real estimates" are only necessary for new parts (Genuchten and Gils 1988).

Suppose, for example, that the new project to develop is a business application, more precisely automation of a financial administration. A further specification is that the customer is an educational institute. With these specifications a classification is now possible and with an intelligent search mechanism it will be possible to find in the database similar projects. The effort spent on these comparable projects may then be used in the estimation activity.

It is remarkable that it is hard to find examples for databases on completed projects in practice, especially databases usable for software cost estimation in the way mentioned here. It is all the more remarkable because the advantages for data on historical projects are obvious and recognized everywhere.

A summary of this activity is presented in figure 6.

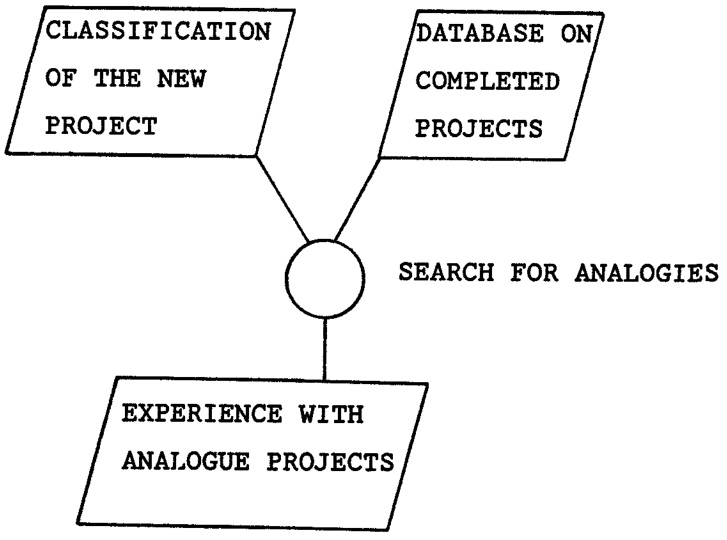
CLASSIFICATION OF THE NEW PROJECT
DATABASE ON COMPLETED PROJECTS
SEARCH FOR ANALOGIES
EXPERIENCE WITH ANALOGUE PROJECTS

Figure 6: Search for analogies

Determining new effects

Examples of new effects are fourth and fifth generation equipment, development environments, workbenches, prototyping and end user computing. If there is little or no experience with these kind of tools and methods, not much help can be expected from data on completed projects. In those cases experiments must be carried out and sufficient slack must be build in the estimation and planning for a pilot project.
The inputs for this activity are less formalized due to its nature. We are talking here about general knowledge on these new effects and the results of the tests that have been carried out. The output would be an estimation of the effect that these new methods are likely to have on the development productivity.

Determining estimation assumptions.

On the basis of the previous activities it is now possible to make a complete overview of all the assumption that have been made until now and which will serve as a basis for the estimation activity. All the results from the previous activities combine here to form an explicitly stated overview of the assumptions, agreements and understandings regarding (Heemstra 1987):

- What kind of project is to be executed. This regards factors like size, the required quality and functionality, the complexity, the amount of documentation needed, the size of the database and the rate in which previously developed code may be used.
- Which resources may be used. This concerns matters as restriction on the points execution time, response time and memory capacity, the way in which modern tools are used and the way in which modern programming methods are used.
- By whom is the project carried out. The quality and experience of the developing team, the turnover rate of the personnel and the quality of project management are factors that have to be considered here.
- How is the project carried out, meaning which method of project management is used and what are the demands that are put upon the duration of the project.

- Finally one also has to look at <u>for whom</u> the project is carried out. The degree of user participation, the number of users, the stability of the user requirements during the project and the experience of the staff in the user organization are factors that can have a significant effect on the project execution.

By formulating all these assumptions in an explicit way one obtains a consistent basis for the estimation activity. Furthermore it is easier to notice deviations from these assumptions during project execution, thus making it possible to take action either by adapting the estimate to the new situation, by taking action regarding the project execution or by changing the quality or functional requirements. The inputs and outputs for this activity are summarized in figure 7.

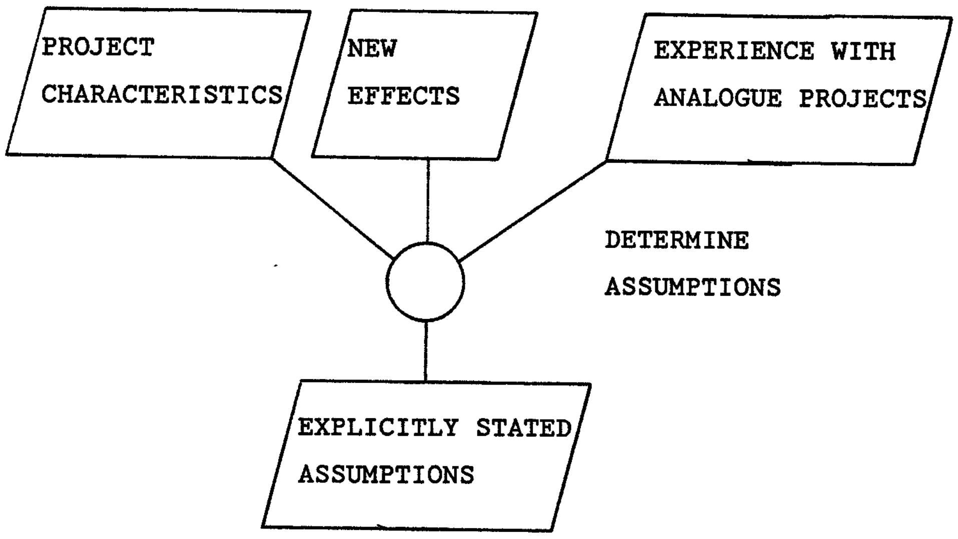
PROJECT CHARACTERISTICS
NEW EFFECTS
EXPERIENCE WITH ANALOGUE PROJECTS
DETERMINE ASSUMPTIONS
EXPLICITLY STATED ASSUMPTIONS

Figure 7: determining assumptions.

Estimating

As was shown in the previous section one of the main aspects of project control is a proper estimate of the needed resources. It is not the purpose of this paper to enter deeper into this matter. In section 4 a short overview of some of the major estimation methods is given. For a more thorough treatment of the subject we would like to refer to the quite extensive literature on the subject (e.g. Boehm 1981, Heemstra 1987b, Dekker and Bosch 1983)).

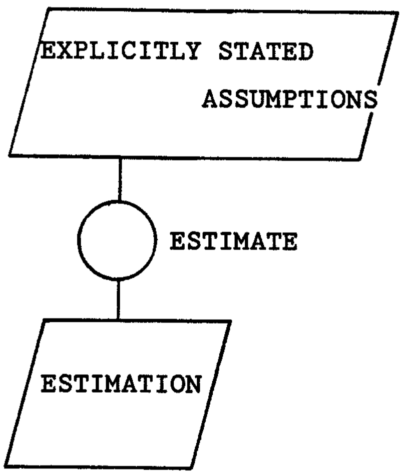

Figure 8: Estimate

The main input for any estimation activity, however it is carried out, are the explicitly stated assumptions which were derived in the previous activity. The output, of course, will be an estimation of the resources necessary to carry out the project. (See also figure 8).

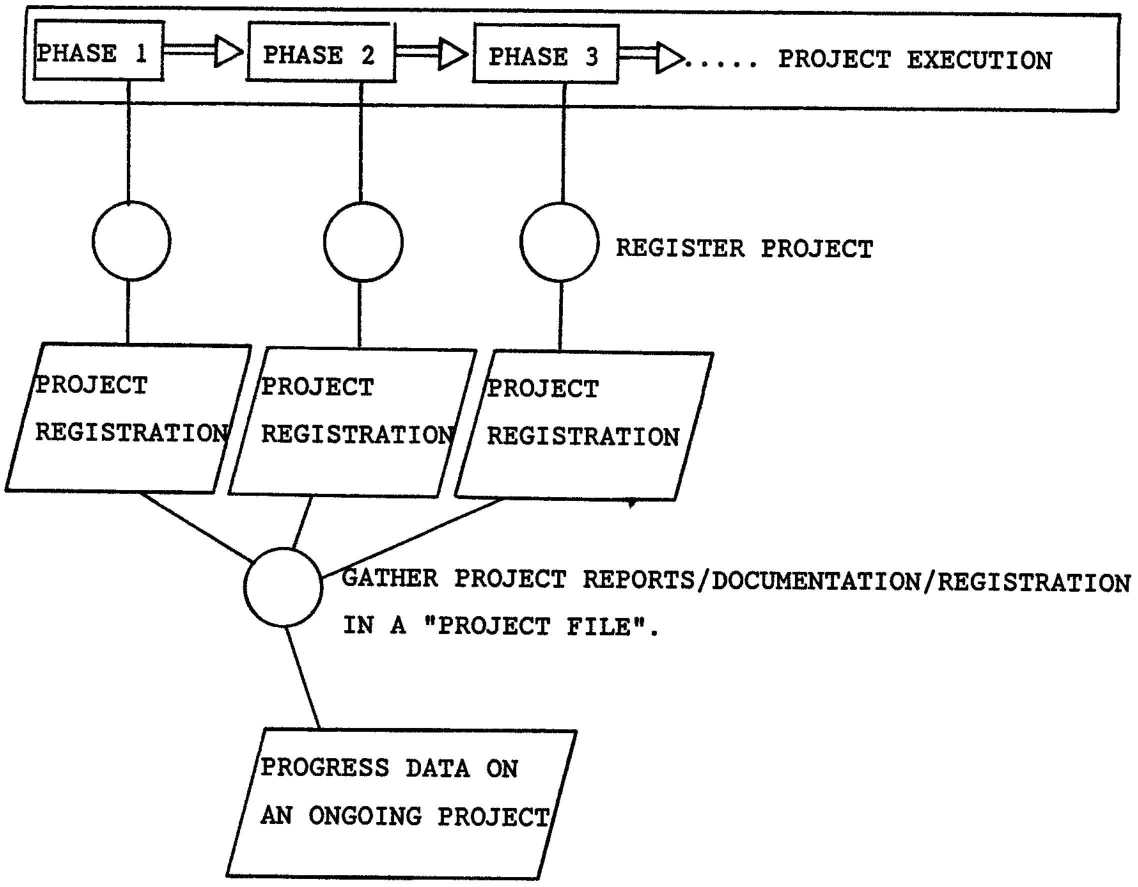

Figure 9: Project registration of an ongoing project

Execution and registration

After the project estimation has been made it is now possible to start with the project execution. Together with this it is necessary to start with the registration of the project. As mentioned in the previous section registration is an essential part of the project control cycle. Registration is not a once only event at the end of the project, but an ongoing activity during project execution. Especially for software project control it is important to compare continuously the estimated and realized expenditure. Comparing is only possible in an effective way if for each phase (for each milestone) registration of costs, effort and development time spent is carried out. In figure 9 it is indicated that the number of data on an ongoing project is growing during project execution. These registered data are used for two purposes. At first they are needed to control the project in execution. Later they will form a part of the historical projects database. In sections 6 and 7 we will say more about the item "registration".

Comparing, analyzing and correction

Maybe one of the most important parts of the conceptual framework of software cost control and estimation is:

- comparing the real expenditures and the estimations,
- analyzing any possible differences and
- translating the results of the analyses in corrections.

As can be seen in figure 3 this comparison between realizations and estimations can take place in several ways. First the realized progress is compared with the progress that should have been attained according to the estimations. This is done on the basis of a comparison of the realized expenditure in time and effort to the estimated expenditure. If there is no significant deviation between the two no further action has to be taken and the next phase of the project can be executed. If however a significant deviation occurs one has to look at the cause of this deviation. It could well be that the estimate itself is not correct. In order to find out if this is the case first the determined project characteristics and, if this gives no result, next the project assumptions are compared to the reality as is put down in the project documentation. It will

be clear that such a comparison is only feasible if the project estimations, the project characteristics, the project assumptions and the project documentation are all recorded in such a way that a match between them is possible. This means that in all cases a similar format must be used. If after this comparison, possibly followed by a re-estimation of the project, significant deviations are still found other measures have to be taken.
Together this means that three kinds of corrections are possible (see also figure 10):

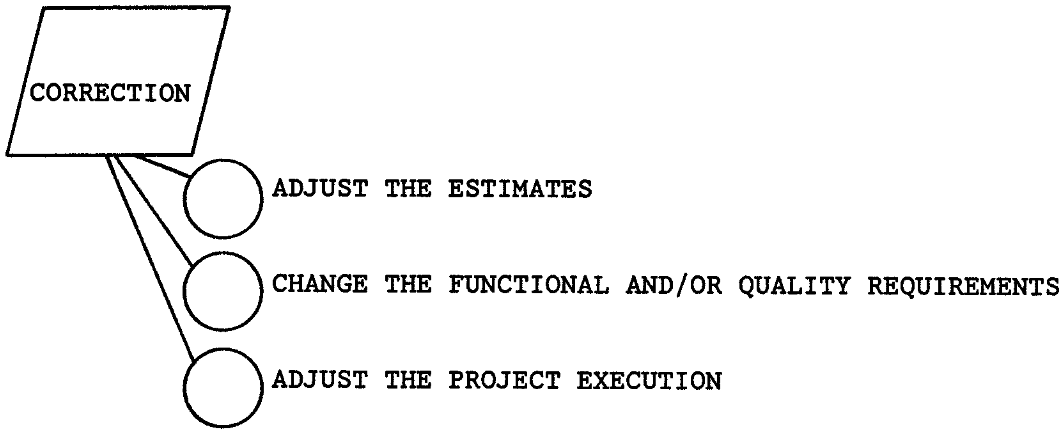
CORRECTION
ADJUST THE ESTIMATES
CHANGE THE FUNCTIONAL AND/OR QUALITY REQUIREMENTS
ADJUST THE PROJECT EXECUTION

Figure 10: Different kinds of correction measures

1. <u>Adjust project execution</u>
   Examples are: to work overtime (the development team tries to win back lost time), moving budgets within a project, rearranging resources, adding more people to a project etc.);
2. <u>Adjust the estimation</u>
   It is possible that the original estimation proves to be unrealistic. In that case plans and estimates have to be corrected. There are two ways of doing that. It might be necessary to re-determine the project characteristics, starting the whole process again from the top. It is also possible that an adjustment of the project assumptions will suffice. In both cases a new estimate is produced.
3. <u>Adjust the target of the project</u>
   This kind of correction is already covered by the previous point. Adjusting the target of the project is in fact carried out by means of an adaption of the project characteristics. However, due to the different nature of this adaption it was deemed necessary to mention it separately. In our opinion, this

kind of adaption means a formulation of a new project and as a result new agreements on needed time, costs and effort.

As mentioned in section 2 software cost estimation is not a once-only event. More than once during project execution estimations and re-estimations have to be made. It is obvious that uncertainties on the characteristics and the cost drivers of a project and also on the aspects time, money, quality, organization and information becomes less as project progresses. In cases of large uncertainties and considerable investments it is recommended to consider each next phase of a project as a new project. Before starting a next phase an

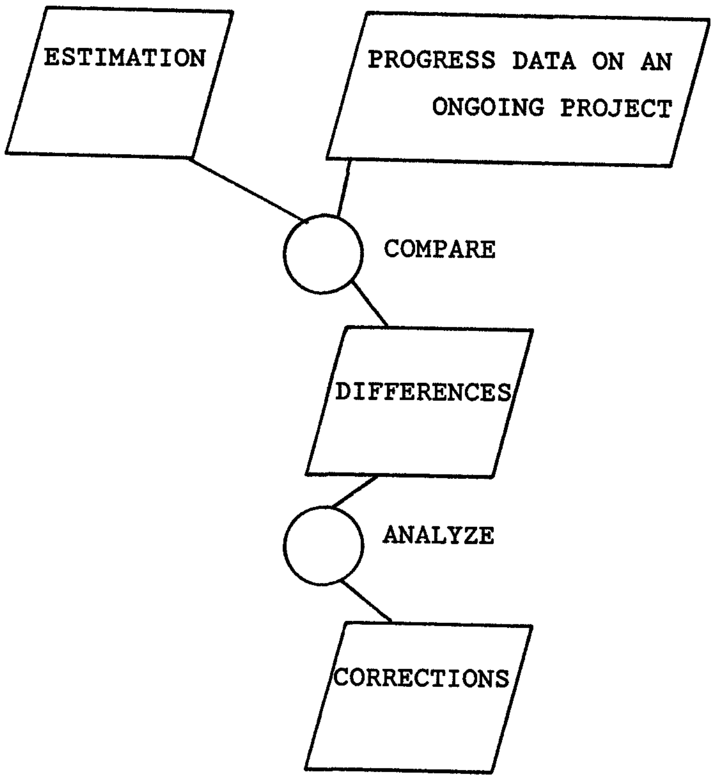

Figure 11: Comparing estimations and realizations, analyzing differences

estimation of time, money and resources has to be made for that specific phase. Furthermore estimations and agreements must be frozen. At the end of a phase it is decided whether to continue or not.

Project documentation and aggregation

The data on a completed (internal) project is the result of registration/documentation of an ongoing project (see figure 12). Once the project is finished this comprehensive set of data is aggregated into a formatted data set. The target of project registration changes from primary a tool for project control to primary a means to maintain a database filled with facts (for example: type of application, duration (total, for each phase), source lines of code (total, for each module), personnel (number, qualification) etc. and experience (causes of miscalculations, things that had not been taken notice of etc.). By means of the activity "aggregate" the data for project control are condensed into a surveyable, easy accessible form and stripped of all its frills. In the study of Noth (1987) an outline is given for such a software project summary. The research of Kitchenham and Smith (1985) within the scope of the Alvey project also gives a initial impetus to the most important data and facts of experience, that have to be saved.

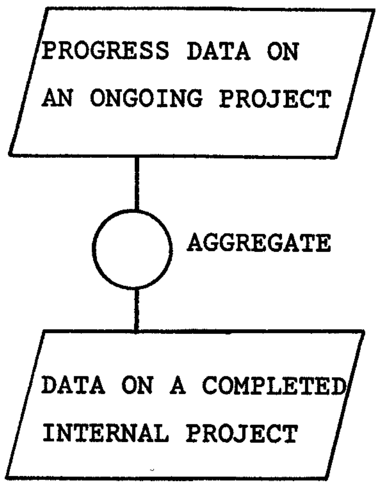

Figure 12: Project registration of an ongoing project

Maintaining the database on completed projects

The database on completed projects is maintained by internal and external data (see figure 13). Internal data are related to data on

completed projects within the same organization; external data deal with software projects from other organizations.
The importance of a good prediction of costs, effort and duration is obvious and the need for data on historical projects to help achieving this is recognized everywhere. Many organizations try to compensate for the lack of this kind of data by purchasing software cost estimation models. These models are after all based on a large number of historical projects. It could well be possible that estimations becomes rather worse than better by using only external data. Besides an intelligent use of external data (for examples results of research on the influence of a specific software tool or productivity metrics from similar organizations, development costs ratios for the different phases of a software project), every organization has to register data on its own projects.

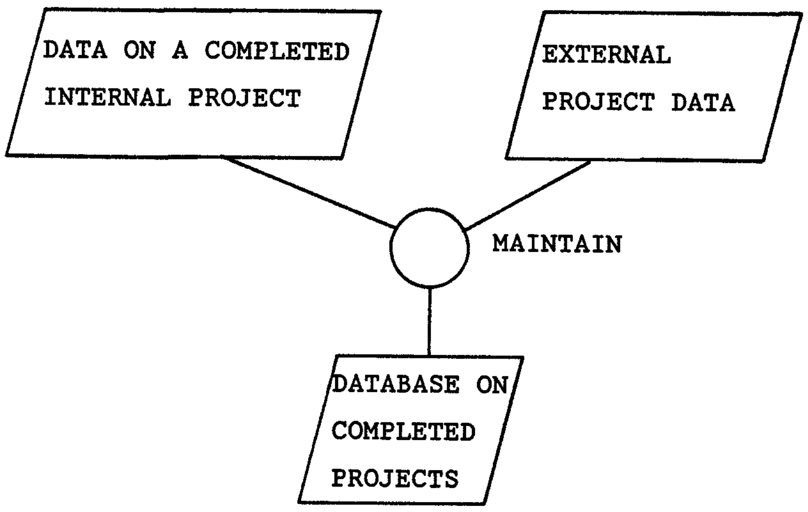
DATA ON A COMPLETED INTERNAL PROJECT
EXTERNAL PROJECT DATA
MAINTAIN
DATABASE ON COMPLETED PROJECTS

Figure 13: Maintain the database on completed projects

4. Software cost estimation

As mentioned above the estimation of effort, time and costs needed for the planned activities, is an important element in software cost control. It was also mentioned that the mechanics of software cost estimation would not be discussed in this context. However, since the subject is so important, we will provide a short overview of the main methods. It has to be kept in mind that, whatever method is chosen, the conceptual framework as described above will retain its

validity. Especially the need for registration and the need for data of completed projects will always remain.
The three most important methods that merit serious attention are:
1. Estimations made by an expert,
2. Estimations based on reasoning by analogy,
3. Estimations based on the use of parametric models.

The reliability of estimates, based on expert judgement, depends for a great deal on the degree in which a new project concurs with the experience and the ability of the expert to remember facts of historical projects. Registration of data from completed projects will be of great value because an expert doesn't have to trust on his memory only. Furthermore it is possible that the expertise becomes portable to other, not-experienced software cost estimators.

The basis of a cost estimation method based on reasoning by analogy, are data of similar historical projects or similar project parts or modules. To find a similarity between a new project and one or more completed projects it will be necessary to gather and register data and characteristics of old projects.

The reliability of software cost estimation models depends for an important part on the content of the project database the model is based on. The equations in Boehm's COCOMO (Boehm 1981) are derived from a database of 63 projects, carried out between 1964 and 1979 by TRW Systems Inc. The proprietary algorithms in SPQR are derived from a knowledge base of over 3000 software projects from more than 200 organizations. Estimacs has built in experiences of some 5000 Data Processing projects. However one will always have to keep in mind that these models are based on projects, carried out in other organizations that each will have their own peculiarities. It is not realistic to suppose that experiences of other organizations are directly applicable to your own organization.

5. Registration

All through this article the need for registration has been stressed. Together with estimating this is the main element of

software project control. In this section we will take a closer look at what it is that has to be registered. As mentioned before in section 2 a description of the project consists of both a description of the product to be made (in terms of modules, etc.) and of a description of the project (in terms of phases, activities etc.). In figure 14 a rough framework of the relation of both kind of descriptions is given (Faszbender 1988).
This framework can be used as a starting point for the estimation of the time, costs and resources needed in the different phases and activities for developing the described components of the desired product. The cells of the framework are filled with estimations concerning the amount of resources, time and costs needed to produce a specific product component during a specific phase or activity. For example: type of application, duration (total, for each phase),

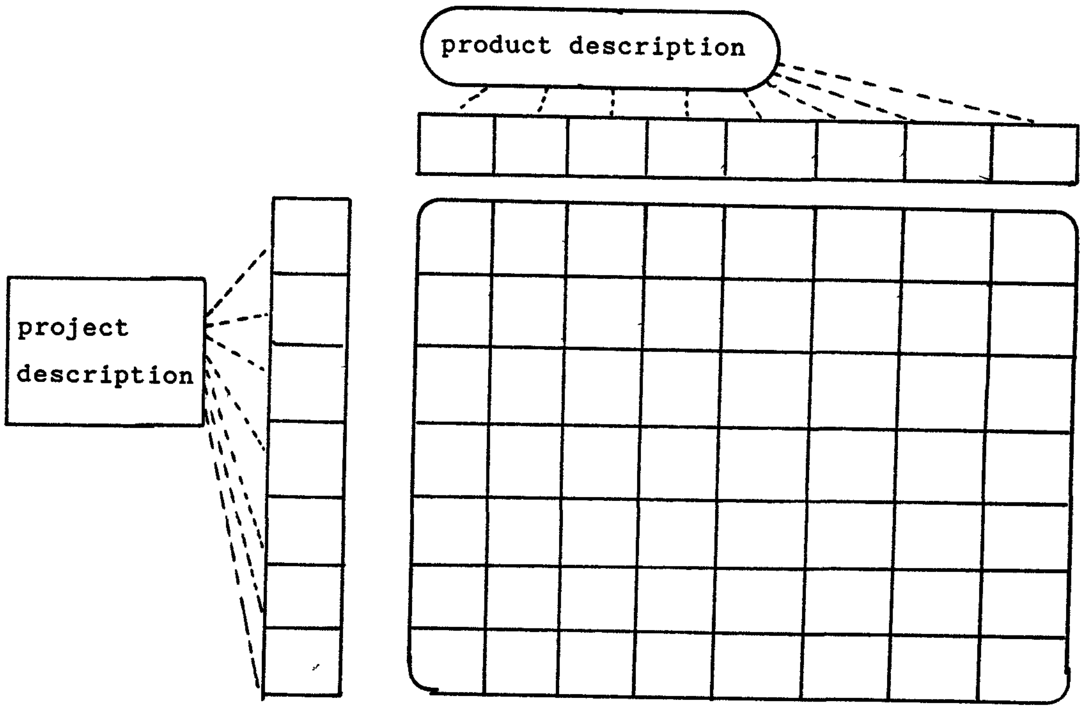

Figure 14: A rough framework of the relation of product and project description.

source lines of code (total, for each module), personnel (number, qualification) etc. and experience (causes of miscalculations, things that had not been taken notice of etc.. it should be noted that the data on a ongoing project have to be recorded in such a way that a quick retrieval and selection of relevant data is possible. This can best be achieved by means of a registration on a computer, all the more because a lot of data on the product, the process and project are already available in a computerized version.

6. A project database

One of the important features of the conceptual model as it was presented in section 3 is the use of a database filled with the condensed results of finished projects. The need for data on completed projects has been underlined by a number of studies (Kitchenham and Smith 1985, Heemstra 1987a, Vliet 1987) and has been recognized in industry more than once.
It makes no difference if a software cost estimate is based on expert judgement, on reasoning by analogy or on parametric models. Estimates must always be based on data of completed projects. If there is a lack of these kind of data the knowledge of and experience with developing software, on specific project and product characteristics and on the influence of cost driver only exists in the heads of a few people. For others, who are confronted with similar problems it is difficult and in most cases impossible to locate this fragmentary and unstructured knowledge and experience. In this way mistakes are repeated. A database with historical project data in which the knowledge and experience of the individuals are made explicit, can support project management in estimating the effort, costs and time required by offering relevant information on old and comparable software projects (Noth 1987).
In this article the importance of this kind of data is focused on the control of ongoing software projects an on the use of estimation methods. Besides these two applications, data on completed projects can be an important aid in some other areas. In (Heemstra 1987b) an overview of these other application areas is given:

1. Calibrating a software cost estimation model.
   Before using a model in a specific development environment for the first time it must be tailored to that environment. In other words: it is necessary to calibrate.

2. Analyzing software development.

   Examples are:

   - Translating data into productivity metrics and analyzing the effect of certain developments (for example fourth generation tools on productivity);
   - Support marketing strategies. With what kind of applications, programming language, customers, etc. do we have the experience and do we have to focus our attention in marketing;
   - Learning effects;
   - Standardization.

It is remarkable that it is hard to find examples of databases on completed projects in practice, especially databases that can be used for software cost estimation in the way mentioned here. It is all the more remarkable because the advantages for data on historical projects are obvious and are recognized everywhere. There might be several reasons for the lack of this kind of data.

- Usually a software developer is not inclined to commit his activities to paper in detail. An argumentation often heard among analysts, designers and programmers is that it is better for them to spend their time on developing the system than waste it by filling in a diversity of forms (Dawes 1985). The people most concerned often consider such a registration system as a control mechanism for projectmanagement, that will only work against them. Effective registration is not possible if this attitude exists in an organization. Software developers have to be convinced that such a system can only be to their advantage. Software cost estimates based on reliable data from completed projects are more realistic; for the developing team this means that they can do their work under less time pressure and with more care for quality.
- Registration of data on completed projects is focused nowadays mainly on financial aspects like development costs for programming, analysis and design, costs for travelling, hardware etc. Data like the degree of user participation, the desired reliability of the software, the complexity etc. are not recorded. These data, however, are necessary for software cost estimation.
- In the case that data are available, these will mostly be scattered in several departments and development teams, usually

with little standardization. The same kind of data are gathered seldom, the same concept often is defined in a different way etc. Agreements and guidelines are a first condition for a consistent set of project data.

- For the time being it is not clear exactly which data have to be registered. In this section some remarks have been made on this aspect. Overlooking several existing databases on completed projects (for example: the IBM-FSD database (Walston and Felix 1977), the COCOMO database (Boehm 1981), the database of the Rome Air development Centre (Putnam 1978), the Systems Development Corporation (SDC) database (Bailey and Basili 1981), the "Dekker en van den Bosch" database (Dekker and Bosch 1983) and in particular the research of Kitchenham and Smith (1985) as a part of the Alvey Software Data library project) it is remarkable that different kind of data are registered in the different databases. These differences can be explained partly because of the variety of development environments in which the databases are created. However this meant that a project manager with good intentions got no clear guidelines. From literature he won't get a satisfying answer to his question, what kind of data are relevant for registration.

Although these arguments might sound compelling, we have shown conclusively that such a database will be a useful help in managing software projects.

7. Conclusions

In this article we have indicated the central position of software cost estimation in software project management. Controlling software projects requires the next five activities, which are indissolubly linked:

* formulating project result,
* planning and estimating,
* checking (measuring progress),
* correcting

The execution of these activities is not a once only affaire, but a continuous necessity during project progress. Registration becomes a

key element now. Controlling isn't possible without registration of the ongoing project. Furthermore registration of completed projects has to be the basis for each estimation/re-estimation. Prediction without taking in consideration in an illusion. The general conceptual model presented in this paper can be used as a referential framework for controlling software projects. All relevant activities, their interdependencies and the required data sets are given.
This framework can now be used as a reference model against which it is possible to judge existing project organizations. It can furthermore be used for the selection of tools and can of course serve as a basis for continuing research.

References

Bailey, J.W., and Basili, V.R. "A meta-model for software development resource expenditures." Proceedings of the 5th international conference on software engineering, IEEE 1981.

Bemelmans, T.M.A. Bestuurlijke informatiesystemen en automatisering, Chapter 8, third edition, Stenfert Kroese 1987.

BIS/Estimator. User Manual version 4.4, BIS applied System Ltd. 1987.

Boehm, B.W. Software engineering economics, Prentice Hall 1981.

Boehm, B.W. "Software engineering economics." IEEE transactions on software engineering, volume se-10, no. 1, january 1984.

Before You Leap. User's Guide, Gordon Group, 1986.

Computer Associates. CA-Estimacs User Guide, Release 5.0, july 1986.

Craenen, G. "begroten van automatiseringsprojecten". Informatie, volume 26, no. 3, march 1984.

Dawes, B. "Management techniques to control software development." Data Processing, volume 27, no. 9, november 1985.

Dekker G.J., and Van Den Bosch F.J. "Functional requirements for the development and use of a software cost database." Information and Managament, no. 6, 1983.

Demshki M., Ligett D., Linn B., McCluskey G. and Miller R. Wicomo tool, Wang institute cost model tool, User's Manual. Sofware product report PUM1 release 1-1-82, 1982.

Faszbender, F.J. Projectbeheersing: wie schrijft, die blijft. Afstudeerverslag Technische Universiteit Eindhoven, 1988.

Galbraith, J. Designing complex organizations. Addison-Wesley, 1973.

Van Genuchten, M., Heemstra, F.J. and Kusters, R. "A Theoretical and Empirical validation of automated tools for software cost estimation". To be published, 1988.

Van Genuchten, M., and Van Gils, A.C.E. The application of knowledge-based systems to software cost estimation. To be published, 1988.

Heemstra, F.J. "Wat bepaalt de kosten software?." Informatie volume 29, extra edition, pp. 586-601, 1987.

Heemstra, F.J. "Het begroten van automatiseringsprojecten." Proceedings NOBO, Groningen 1987b.

Heemstra, F.J. Het begroten van softwareprojecten: meten is weten!. Report EUT/BDK/28, ISBN 90-6757-028-1, Eindhoven 1987b.

Jones, C. Programming Productivity. McGraw-Hill, 1986.

Kemerer, C.F. "An empirical validation of software cost estimation models." Communications of the ACM. volume 30, no. 5, may 1987.

Kitchenham, B.A., and Smith, M. "Software data library Phase 1 manual software collection." SDM SIG/DP(86), 1985.

Lundeberg, M.; Goldkuhl, G.; and Nilsson, A. Systeemontwikkeling volgens ISAC, de ISAC-Methodiek. 2e edition, Samsom 1982.

Miyazaki, Y., and Mori, K. "COCOMO evaluation and tailoring." Proceedings of the 8th international conference on software engineering, IEEE 1985.

Noth, T., and Kretzschmar, M. Aufwandschaetzung von DV-Projekten. Springer Verlag, Berlin 1984.

Noth, T. "Unterstutzung des softwareprojectmanagements durch eine Erfahrungsdatenbank." Proceedings Compas '87, Erfolgs faktoren der integrierten Informationsverarbeitung, AMK Berlin, May 1987.

Putnam, L.H. "A general empirical solution to the macro software sizing and estimating problem". IEEE transactions on software engineering, SE-4, 4 , 1978.

Rubin, H.A. "A comparison of cost estimation tools". Proceedings of the 8th international conference on software engineering, IEEE 1985.

Stanley, M. "Software cost estimating". The proceedings of the ISPA 6th annual conference, 1984.

Van Vliet, J.C. "Wat kost dat nou, zo'n programma". Informatie volume 29, extra edition, 1987.

Walston, C.E., and Felix, C.P. "A method of programming measurement and estimating". IBM system journal, 16, 1977.

Wijnen, G., Storm P., and Renes W. Projectmatig werken. Marka paperback reeks, 1984.

Wolverton, R.W. "The cost of development large-scale software". IEEE transactions on computers, volume c-23, no. 6, june 1974.