University of Rostock
Department of Business, Economics and Social Science
Chair of Business Informatics

Calculation of Sequences of Activities

Seminar Business Informatics

Summer semester 2005

Author:	André Schütz
Course of Study:	Business Informatics
date of lecture:	08 July 2005
Adviser:	Dr. Horst Günther, Chair of Business Informatics

Contents

List of Figures

Abstract

Modern organizations always strive to optimize their sequence of activities and internal processes. Therefore, they have to calculate the relevant time and cost factors to get measurable clues for the activities. These factors are sometimes guessed and sometimes calculated. That's why concrete methods are necessary to deliver the relevant factors and to make the organizational processes comparable.

1 Introduction

Nowadays, every organization has to optimize their activities and internal events. Therefore, they have to subdivide the complete processes into smaller activities which are easier to understand and to influence. By it, they will be able to calculate individual working hours and get an overview about the necessary skills which are useful for the single activities.

But how does such companies get the necessary time and cost values
to specify the individual activities?
And why are these time and budget values so important?

The organizations have to calculate or guess the relevant factors what can lead to a lot of problems. Comparing to figure 1[1], the calculation of activities can be full of mistakes and the determined values are very often insufficient.

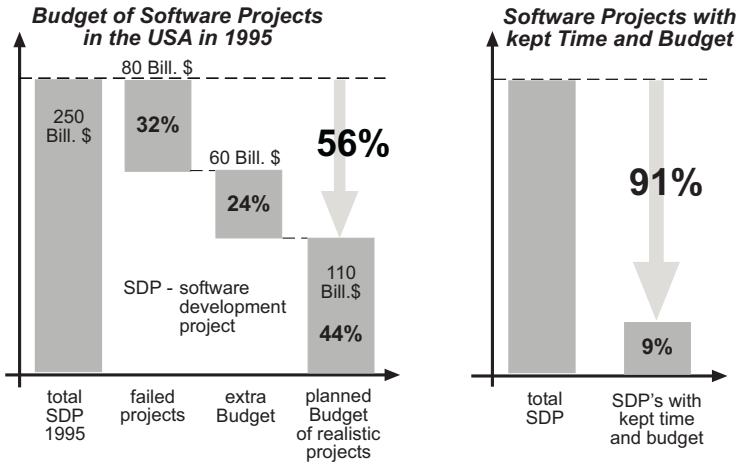

Figure 1: Budget of Software Projects

The paper is subdivided into three parts. Part one is an introduction about the activity definition (1.1), the evaluaton criterion (1.1.3) and how to evaluate the single activities (1.2), calculation methods (Empirical Estimations 1.3 and Algorithmic Estimations 1.4), an overview of problems and mistakes which can occur during the calculation of the estimated time and cost vales (1.5) and the graphical representation (1.6).
Part two describes the Cocomo II model which is used to estimate time and cost values for the development of software projects. Therefore, it is subdivided into Sizing ($\Rightarrow$ estimation of lines of code 2.1), the three Cocomo II sub-models *The Application Composition Model* (2.2), *The Early Design Model* (2.3) and *The Post - Architecture Model* (2.4). Finally, a short example is given (2.5).
Part three is a conclusion about the calculation of sequences of activities and Cocomo II (3).

[1] comp. [Wil, page 6]

1.1 Organization, Process, Activity

The calculation of sequences of activities takes place in the individual processes of modern organizations. Therefore, the management of the respective organization has to analyze their processes and subdivide the complex processes into further activities which make it possible to visualize the specific time or cost expenditures in the smallest units of an organization. Such an unit can be a working place or an individual person.
The heart of every organization are the employees who are working for it. If an organization wants to work effective and economic, they have to assign specific time values to the respective activities of their employees. These make it possible to compare the output of the individual employee in relation to the time and to optimize the working behavior of single employees. Furthermore, these time values are necessary to determine the process progress, they are necessary for the calculation of cost factors and are useful for the planning of resource consumptions and reservations[2].

1.1.1 The Process

The process can be seen as a result of single activities. *Becker* speaks from a process as a "[..] completed, chronological and factual order of activities which are necessary to work on process oriented and business related objects." [3]
Such an object could be an order or an article and could be seen as the output of the process. Another necessary factor are input values like resources or intermediate products. The process will transform the input values by the help of the different activities into the required and expected output values.
The processes can be subdivided into core processes and support processes, where the support processes are mostly necessary to carry out core processes.

So a process is specified by an expected result, the single activities and a time factor which describes the length of the process. The assigned time factor on the process level gives only a rough overview and makes it impossible to get an outline of the necessary working times and cost values in the single activities. Therefore, the next lower level, the level of the single activities, is of relevance for the detailed assignment of working times and resource allocations.

1.1.2 An Activity

An activity is an elemental part of a process. You can say that processes are composed of activities which are the single events of the process. Therefore, an activity is an operational step which is necessary to fulfill the whole process and to realize an expected output. [4]
The management has to assign specified cost and/or time values to the different activities of the

[2] compare [Krc03], [Sch97], [Lau03], [Bro99], [Sch98]
[3] [Bec00, page 4]
[4] ibidem [Bec00, page 4]

process which serve as parameters for special valuation methods. In the next step, the summarized time values of the single activities lead to the total amount of time which is necessary to fulfill the whole process.

1.1.3 Evaluation Criterion

The evaluation of the single activities of a process has to be done with the help of fixed factors which are the holders of specific values. Such factors can be e.g. the time, costs or the proportional use of a capacity. The content of the relevant factors are values like hours, days, amount of dollar or resources.
The most often used factors are the time and costs. These can be numerous in their meaning so that the following time and cost factors are thinkable.
Cost factors:

- Costs for the preparation of the transformation process (e.g. setup costs).
- Costs for the use of input factors, i.e. resources.
- Costs for a late delivery time (e.g. delay costs).

Time factors:

- A time value for the realization of an activity (e.g. minutes, hours, days).
- Specific time values for the activities: start and end of processing, waiting time, passing time, deadline deviation, delays, etc.
- Efforts of software projects in person months or person days. Normally, one person day has 8 hours.

1.2 Evaluation of Activities

The management level of an organization has to determine the time and cost values for the activities. Therefore, they have to use concrete methods which make it possible to guess or calculate the required values. Such methods have to analyze the working places, classify them depending on their demands and produce concrete time or cost values.
These methods can be subdivided into *Empirical Estimations* and *Algorithmic Estimations*, which include the following methods:

Empirical Estimation:

- Expert Estimation
- Delphi Method

Algorithmic Estimation:

- Cocomo Method
- Function Point Method

1.3 Empirical Estimation

Empirical Estimations are based on the experiences of the single individuals who are involved. Because of that, the quality of the estimation is mainly dependent on the personal experiences and measurable observations of the person who is guessing the facts.

1.3.1 Expert Estimation

Expert Estimation will be done by so called *experts*, who guess the expected time and cost values by their personal experiences and by comparison with earlier projects[5].
But the name of the method *Expert Estimation* can be misleading, because it allows a reliable estimation by real experts and in the same way a rough estimation by unexperienced persons. Therefore, the quality of the method depends on the experience of the chosen expert and his knowledge to rate the respective activities as exact as possible. To do so, the expert needs data about already finished projects by the respective project teams and has to use suitable techniques (like in 1.4.2 and 1.4.1) to guess the necessary values.
In the practice, the Expert Estimation is the most frequently used method and can deliver reliable results if the experts are experienced and have comparable data of already finished projects.

1.3.2 Delphi Method

The Delphi Method is an objectified Expert Estimation. Different persons make individual estimations, independent from each other. In a second round, the persons get a summary of the first estimation and have to make another estimation. If they differ from the mean value of the previous round, they have to explain their changed estimation. The whole procedure will be repeated until the estimations of the different persons are nearly equal[6].
The Delphi method delivers better results than the Expert Estimation because it eliminates runaways in the estimations. As a disadvantage, the Delphi Method has a bigger estimation expenditure.

1.3.3 Techniques for Expert Estimations

The two most important techniques for the Expert Estimation are the *Direct Analogy Estimation* and the *Estimation by Decomposition*[7].

[5] comp. [Gli, page 52]
[6] ibidem [Gli, page 52]
[7] ibidem [Gli, page 52 / 53]

Direct Analogy Estimation

A finished project, which is nearly like the current one, will be used to rate the present project. The known time and cost values of the finished project are the basic values which will be raised or reduced depending on the differences between these two projects. The additional or reduced time and cost values are again estimations.

Estimation by Decomposition

The estimated project will be decomposed into the single activities. The individual activities will be estimated and the expenditures summarized to get a total time and cost overview.
To get useful results, the estimated cost and time values have to be corrected because of the fact that the relation between the product size and the expenditures is not linear[8]. So additional expenditures for the communication, the coordination, the problem complexity and the project management have to be considered.

1.4 Algorithmic Estimation

Algorithmic Estimations are made of one or several algorithms to calculate the time or cost function out of different variables.
The exactness of these methods depends on two important dimensions [9]:

- The *input values* of the cost function have to be precise estimated.
- The model has to be adapted, i.e. the values of the cost attributes have to be adjusted to the respective development environment.

The *Algorithmic Estimations* deliver the best estimations, if the preparatory work has been done precisely.

1.4.1 Cocomo Method

Cocomo (Constructive Cost Model) was developed by Boehm. The product will be estimated by KDSI (Kilo lines of delivered source instructions). The KDSI and some further cost factors are used to calculate the time and cost values of the project.
Depending on their complexity, the software projects will be subdivided into application programs (organic mode), program systems (semi - detached mode) and embedded systems (embedded mode) [10].
Boehm revised the Cocomo method and refined it, after its development in 1981. The result was summarized in Cocomo II in 2000 (A detailed explanation can be seen in section 2).

[8] ibidem [Gli, page 53]
[9] ibidem [Gli, page 54]
[10] ibidem [Gli, page 55]

1.4.2 Function - Point - Method

The *Function Point Method* is a relative measure to determine the functionality, i.e. the scope of the system[11]. It was developed by Albrecht at IBM in 1979.
The method is based on the idea to count and rate factors like the external in- and output, the external inquiry, interfaces to external data and internal data. The factors will be classified with simple, normal or complex and rated with a weight. The sum of the factors represent the Function Points, which will be corrected with a weight factor that reflects the technical complexity of the system.
The Function Points can be converted into man month[12] or lines of code, so that the time or cost values can be derived.

1.5 Problems and Mistakes

1.5.1 Friendly Guessing

Today, the business is hard and the customer wants to have the cheapest contract that is possible. Therefore, the software organizations try to get nearly every project that is somehow possible to solve with the available resources. And there comes the big mistake. Many organizations overestimate their possibilities and make wrong or unrealistic offers. Mostly, the calculated time values are unattainable for the teams and the estimated costs are to low calculated, like in figure 2[13]. The offer is corrected downwards to get the contract. In a next step, the organization has to raise the costs every time the real costs are catched up with. It seems that the first realistic cost calculation was done after 9 months, where the estimated costs are trebled. Here, it becomes clear, that the first cost estimation was to optimistic and that the contract estimation, which was presented to the customer to get the contract, was completely unrealistic.
That creates time and cost schedules which are impossible to fulfill and makes it necessary to touch up the whole schedule during the project development.

1.5.2 The Content can Change

The negotiated content of the program and the defined functions, data structures and technologies are only a first approach to describe the software. Mostly, the customer expects additional functions during the development of the project or gets the feeling that the program does not fit his expectations.
Then the programmer has to talk with the customer about a change in the concept and has to find out, what is possible to integrate in the current version of the program and where are bigger changes to make. To add some additional functions into a running system, which does not change the basic structure of the program are not such a problem. Only, if the customer demands for

[11] ibidem [Gli, page 56]
[12] or person month
[13] ibidem [Gli, page 51]

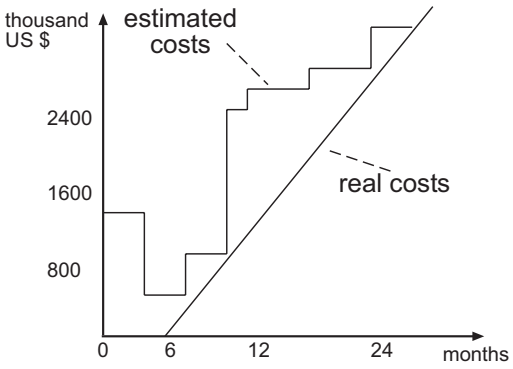

Figure 2: Cost Schedule

additional functions or technologies which change the basic structure and which make it necessary to change the planned software development progress are very time consuming and cost increasing. Such a change in the concept or the buffer for additional demands should be included in the project planning and in the calculation of the single activities.

1.5.3 The Experience

The calculation models need specified time or cost values to assess the respective activities. These values will be calculated by valuation keys, experiences of former projects or assumptions. And that is the problem!
Mostly, software projects are completely new software components which are hardly to rate. Methods like the Function Point Method or the COCOMO (Constructive Cost Model) Method try to make it possible to calculate the complexity of the respective program and to estimate first time values.
But nevertheless, because of the complexity of the modern software projects, the estimated time and cost values are very vague and only a rough assessment. They are a first clue for contract negotiations but will be very inexact if the customer changes the concept. Mostly, they can be refined during the development of the project. So it takes many years to gain advantage out of the experiences of former software projects which can then be used in new projects.

1.5.4 The Programmers

Another important factor is the programmer. Experienced programmers are much more productive in solving basic and unexpected problems in much shorter periods of time than unexperienced programmers. It is possible to consider the productivity of the programmers but it is hard to predict the improvements of the single programmers during such a project and to consider the individual creativity in solving complex problems. Furthermore, the software development is mostly brainwork, and is depending on the physical fitness of the programmer which can sway by more

than one order of magnitude[14].
That makes it hard to make perfect predictions so that the time estimations are full of mistakes and uncertainties.

[14]ibidem [Gli, page 52]

1.6 Representation

Different models were invented to represent the sequences of activities of industrial processes. They show the single activities of an industrial process in a graphical form and give information on the necessary time and/or cost values of the process. The process will be represented in the form of the single activities and the edges or the nodes are the holders of the time or cost values. Figure 3 shows an abstract image of such a model. The nodes are filled with time values for the different activities like earliest and latest starting and ending times, a puffer, the length of the respective activity and descriptions of the specific node[15].

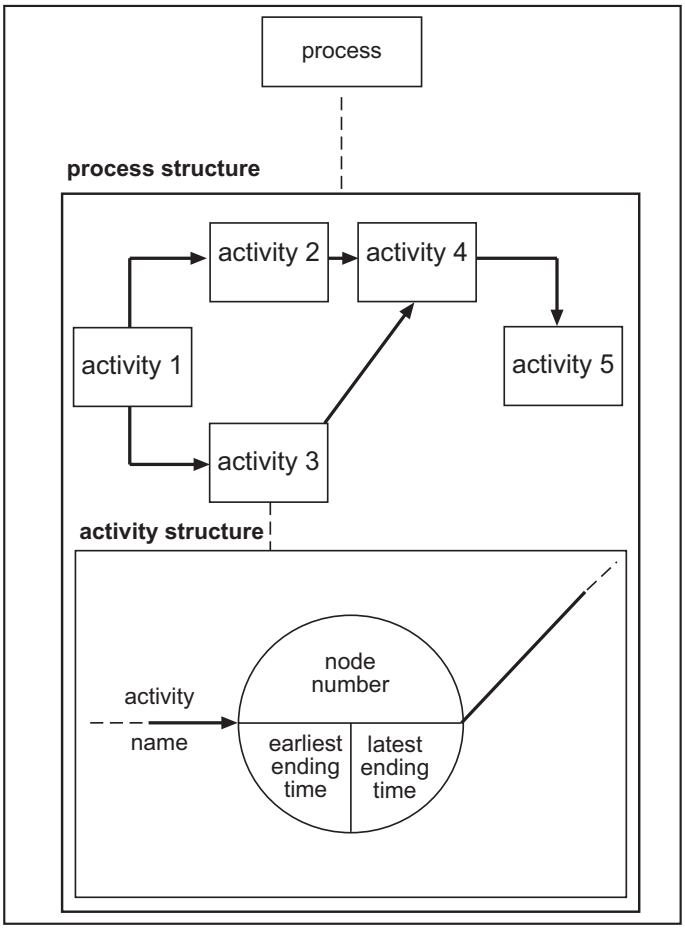

Figure 3: Theoretical Process Model

A very famous model is the network representation of a process. This model visualizes the activities of a process as edges and the nodes represent important events which are necessary for the beginning or the end of an activity.

[15] comp. [Ber04]

The following software project (see figure 4) is an example to see the calculation of activities in a network model. The whole development process is subdivided into nine activities (Requirements Formulation, Potential Configurations, System Architecture, Project Scheduling, Interface Design, Structural Design, Implementation, Documentation, System Test) which have specific time values (here only the earliest and latest ending times).
The critical path goes over the Requirements Formulation, the Project Scheduling, the Interface Design, the Implementation and the System Test and has a length of 23 months.

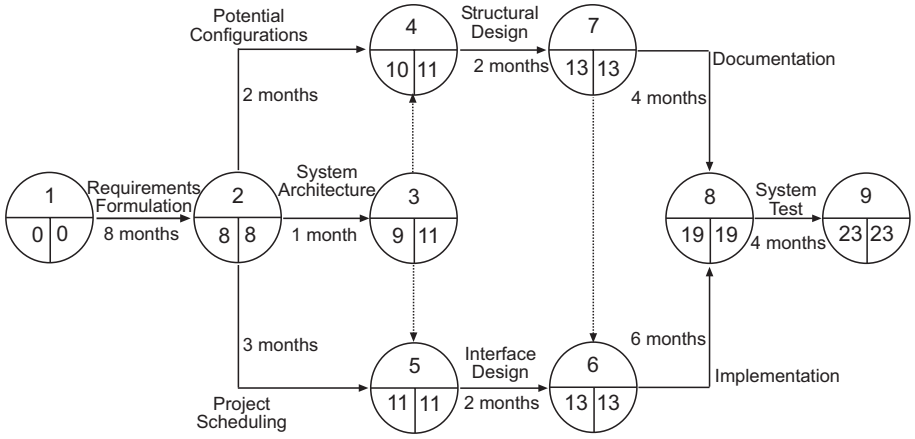

Figure 4: Network Overview

To see the allocation of the activities in another perspective, a Gantt Diagramm (see figure 5) gives a detailed overview.

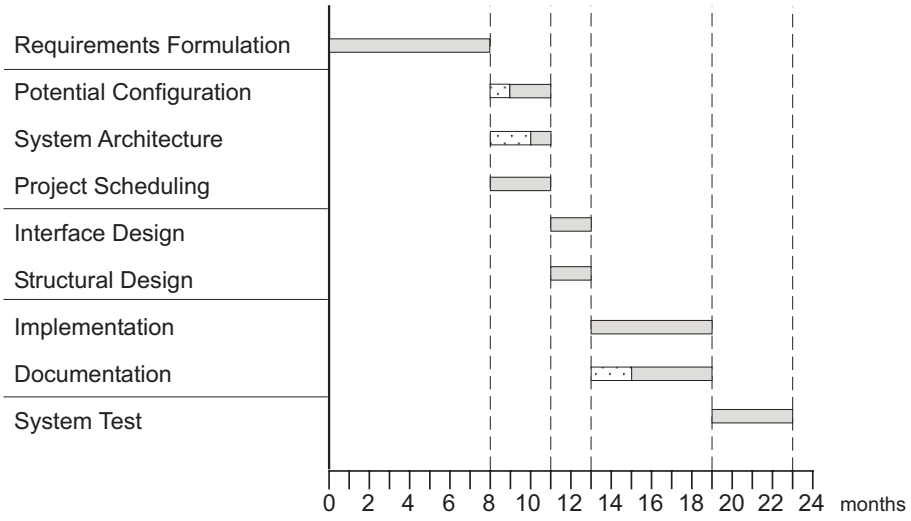

Figure 5: Gantt Overview

2 Cocomo II

The Cocomo II[16] model is the improvement and further development of the Cocomo model (see 1.4.1) of 1981[17]. It will be used to assist in the budgetary planning and schedule estimation of software projects [18].

The differences between Cocomo II and the original Cocomo model[19] are as follows[20]:

1. The original Cocomo model used KDSI[21] as the basic input value. Cocomo II uses different effort estimating models depending on the stage of the development of the current project.
2. The original Cocomo model delivered point values for the effort, but the Cocomo II model creates ranges of estimates which are one standard deviation around the estimate.
3. Cocomo II can be used for the work with reused software, where automated tools make the translation of the existing code. The original Cocomo model did not make useful accommodations for this kind of estimation.

Cocomo II has three different models which are *The Application Composition Model*, *The Early Design Model* and *The Post - Architecture Model*.
The Application Composition Model is suitable for projects which will be developed from reusable components, scripting[22] or modern GUI builder tools and are based on *Application Points*[23] (explained in section 2.2).

The second model, *The Early Design Model*, makes a rough estimation of a project's cost and duration before the entire architecture is completely determined. Therefore, Cost Drivers, Scale Factors and a specific estimation equation will be used (explained in section 2.3).
The Cost Drivers and the Scale Factors will be determined with the help of standardized tables. The single factors are divided into six categories from *very low*, *low*, *nominal*, *high*, *very high* to *extra high* and which have specific values for the estimation equation. These values are known as Effort Multipliers.
As an example[24]:
The Cost Driver *APEX*[25] will be *very low* if the team only has two months of experience and is

[16] The equations of the following sections are only the most general ones. A more detailed form and additional informations about the calculations of single parts of the equations can be found in *Software Cost Estimation with Cocomo II* [[Boe00]].
[17] comp. [Boe, page 626]
[18] comp. [Boe00, page 391], [IES]
[19] Also known as *COCOMO'81*
[20] comp. [Sul]
[21] K Delivered Source Instructions
[22] Art of programming that uses existing components to create applications.
[23] Also known as *Object Points*
[24] comp. [Kam, page 3]
[25] Applications Experience

very high if the team has more than 6 years of experience. The Effort Multiplier of *very low* is 1.22 and for *very high* 0.81.

The Post - Architecture Model is the most detailed one of the three Cocomo II models. It is used after the completion of the whole project architecture and uses new Cost Drivers and a modified estimation equations.

2.1 Sizing

The estimated size of the lines of code is one of the most important factors of a good project estimation. If the estimated lines of code are not precise enough, the result of the estimation will be not very accurate and the organization won't get a good effort estimation for the current project. Therefore, the estimations has to consider different kinds of code. The code can be new, it can be reused (here with modifications or without) and it can be automatically translated (by specific programs which adjust it to the current project).
Cocomo II uses methods to adjust reused code, so that it is equivalent to new lines of code and can be used in the same way.

To count new lines of code, expert estimations of former projects which were similar to the current one or the Function Point method which determines an amount of Function Points that can be translated in new lines of code. The code size will be expressed in thousands of source lines of code (KSLOC) and is defined as amount of intellectual work which was put into the program development[26].
If the organization chooses the Function Point method to determine the lines of code, they have to analyze the amount of functionalities of the product and have to calculate the resulting Function Points for the software project.
In a second step, the determined Function Points will be transformed into lines of code. Therefore, Cocomo II uses standardized tables to convert the Function Points in the new lines of code.

The use of already existing code also has to be considered in the current size of the product. Firstly, the reused code has to be classified [27]:

- Preexisting code that is plugged into the project without modifications is called *reused code*.
- Preexisting code that is modified for the use in the current project is called *adapted code*.

Secondly, the amount of *reused* and *adapted code* has to be adjusted so that it can be treated as new lines of code. This adjusted code will be called *equivalent source lines of code* and is

[26] ibidem [Boe00, page 15]
[27] ibidem [Boe00, page 19 / 20]

abbreviated with *ESLOC* (see equation 1 [28]). The adjustment for the reused code depends on the additional effort that is necessary to modify the code for the use in the current project.

$$ESLOC \ = \ Adapted\ KSLOC \ * \ (\ 1 \ - \ \frac{AT}{100} \ ) \ * \ AAM \tag{1}$$

where

AT = Percentage of code that was re-engineered by an automatic translation.
AAM = Adaption Adjustment Modifier (The effort that is necessary to fit the adapted code into the existing code. For example, the effort to modify a reused module can be twice as high as the development of new code. Or there is a "one for one correspondence between adapting an existing product and developing it from scratch" [29].)

2.2 The Application Composition Model

The *Application Composition Model* can be used to make estimations of prototype developments[30].
The estimation for the software size will be done on the basis of Application Points and the effort can be guessed with the help of a simple size/productivity formula.
The Application Points are an alternative to the well known Function Points and can be seen as an weighted estimate of [31]:

1. *The number of single screens which will be displayed.* One point for simple screens, average complex screens get 2 points and very complex screens are weighted with 3 points.

2. *The produced reports by the system.* Simple reports count 2 Application Points, average complex ones 5 points and very difficult ones count 8 points.

3. *Number of modules in imperative programming languages as JAVA or C++ which have to be developed to supplement the database programming code.* Each module will be counted with 10 Application Points.

The advantage of Application Points in relation to Function Points is the fact, that they are easier to estimate for the project. They only have something to do with screens, reports and modules, but nothing with any implementation details[32].

Additionally, a productivity has to be estimated which describes the difficulty to develop each Application Point. The productivity depends on the experiences of the programmers and the

[28] ibidem [Boe00, page 22 - 25]
[29] ibidem [Boe00, page 25]
[30] ibidem [Boe, page 626]
[31] ibidem [Boe, page 618]
[32] ibidem [Boe, page 618]

capability of the development environments (CASE tools[33]) used for the project. Figure 6[34] shows the Application Point Productivity (PROD) for different complexities: The final equation for the

Developer's experience and capability	Very Low	Low	Nominal	High	Very High
CASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

Figure 6: Productivity of Application Points

estimation of the effort of system prototypes is:

$$PM = \frac{(NAP * (1 - \frac{\%reuse}{100}))}{PROD} \tag{2}$$

where

PM = The effort in person month.
NAP = The total number of Application Points determined for the respective system.
$\%reuse$ = The amount of reused code in the project.
$PROD$ = The Productivity of the Application Points as shown in Figure 6.

2.3 The Early Design Model

The basic effort equation for the *Early Design Model* is defined as follows[35]:

$$Effort = A * Size^E * \prod EM_i + \frac{Adapted\ SLOC * \frac{AT}{100}}{ATPROD} \tag{3}$$

with

$$E = B + 0.01 * \sum SF_j \tag{4}$$

$$Size = (1 + \frac{REVL}{100}) * (New\ KSLOC + Equivalent\ KSLOC) \tag{5}$$

$$Equivalent\ KSLOC = (Adapted\ KSLOC * (1 - \frac{AT}{100}) * AAM) \tag{6}$$

where

[33] Computer Aided Software Engineering
[34] ibidem [Boe, page 628]
[35] ibidem [Boe00, page 69]

$A = 2.94$, $SF_j = 5$ Scale Factors, $EM_i = 7$ Effort Multipliers (Cost Driver)
E = The scaling exponent for the effort equation
B = Scaling base - exponent for effort that can be calibrated, currently set to 0.91
$REVL$ = Percentage of Requirements Evolution and Volatility
AT = Percentage of the Adapted KSLOC that is re - engineered by automatic translation
$ATPROD$ = Automatic Translation Productivity
AAM = Adaption Adjustment Modifier

The original Cocomo model was adjusted by 15 Cost driver attributes. In Cocomo II, 7 Cost Drivers (EM - the values of the seven Cost Drivers are the Effort Multipliers) are given for the estimation of the *Early Design Model*. That are[36]:

1. RCPX - Product Reliability and Complexity (Combines the four Post - Architecture multipliers: RELY, DATA, CPLX and DOCU.)
2. RUSE - Developed for Reusability (Additional effort for the construction of components which will be used in other current or future projects.)
3. PDIF - Platform Difficulty (Combines the three Post - Architecture multipliers: TIME, STOR and PVOL.)
4. PERS - Personnel Capability (Combines the three Post - Architecture multipliers: ACAP, PCON and PCAP.)
5. PREX - Personnel Experience (Combines the three Post - Architecture multipliers: APEX, LTEX and PLEX.)
6. FCIL - Facilities (Combines the two Post - Architecture multipliers: TOOL and SITE.)
7. SCED - Required Development Schedule (Additional time for the development of the project which is necessary to add to the nominal schedule.)

The five Scale Factors (SF) are [37]:

1. PREC - Precedentedness (Is the project a new territory for the organization?)
2. FLEX - Development Flexibility (How flexible is the development team regarding to the development? This means, are there specified goals or is the team free in its choice?)
3. RESL - Architecture / Risk Resolution (The rating is the weighted average of characteristics like: Are critical modules identified in the development plan? Are there any uncertainties in the architecture? Number of critical items? These characteristics are subdivided into 6 classes (from *very low* to *extra high*) with individual numeric values.)

[36] imbidem [Boe00, page 41 - 55]
[37] imbidem [Boe00, page 33 - 34]

4. TEAM - Team Cohesion (Cohesion of the team members.)

5. PMAT - Organization Process Maturity (Level in the Capability Maturity Model.)

2.4 The Post - Architecture Model

The most detailed model of the Cocomo II method is the *The Post - Architecture Model*. It uses new Cost Drivers, new line counting rules and the equation (see equation *3*) of the *Early Design Model* for the estimation of the effort[38]. The effort equation will be used with 17 Cost Drivers (EM) in the *Post - Architecture Model*.
This model defines 17 Cost Drivers which are grouped into four categories as follows:

1. Product factors

2. Platform factors

3. Personnel factors

4. Project factors

The new 17 Cost Drivers (RUSE and SCED are known from the *Early Design Model*) are[39]:

1. RELY - Required Software Reliability (Expected reliability of the product. How dangerous are software errors?)

2. DATA - Database Size (Quantity of the database.)

3. CPLX - Product Complexity (Complexity of a product depends on the focus of the product on Control Operations, Computational Operations, Device Dependent Operations, Data Management Operations and User Interface Operations)

4. RUSE - Developed for Reusability (see *Early Design Model* Cost Drivers)

5. DOCU - Documentation Match to Life - Cycle Needs (Suitability of the documentation to the progress of the project.)

6. TIME - Execution Time Constraints (Necessary processor time.)

7. STOR - Main Storage Constraint (Necessary memory.)

8. PVOL - Platform Volatility (Depends on the used hardware, operating system, database management system and webserver. How often they can chang?)

9. ACAP - Analyst Capability (Skills of the programmers in analysis, design, communication, team work and efficiency.)

[38] imbidem [Sul]
[39] imbidem [Boe00, page 41 - 55]

10. PCAP - Programmer Capability (Programming skills of the programmers.)
11. PCON - Personnel Continuity (Fluctuation of the personal in the organization.)
12. APEX - Applications Experience (The experience of the team with the project type.)
13. PLEX - Platform Experience (Experience of the programmers with the platform (hardware, operating system, database management system and webserver))
14. LTEX - Language and Tool Experience (Experience with the programming language, the development environment and the programming tools.)
15. TOOL - Use of Software Tools (Use of programming tools, CASE - Tools and integrated development environments (IDE))
16. SITE - Multisite Development (Geographical distribution of the development team.)
17. SCED - Required Development Schedule (see *Early Design Model* Cost Drivers)

The Time to Develop (TDEV), which is defined for all three Cocomo II models, has the following equation[40]:

$$TDEV = [\ C * PM^F\ ] * \frac{SCED\%}{100} \tag{7}$$

$$F = (\ D + 0.2 * [\ E - B\ ]) \tag{8}$$

where

C = Schedule coefficient that can be calibrated, currently set to 3.67
PM = Estimated Person months
F = Scaling exponent for schedule
$SCED$ = Percentage of Required Schedule Compression
D = Scaling base - exponent that can be calibrated, currently set to 0.28
E = The scaling exponent for the effort equation
B = The scaling base - exponent for the effort equation, currently set to 0.91

[40] imbidem [Boe00, page 70]

2.5 Example

The project is a database system which has the following characteristics[41]:

- Holds information about the customer and the sales.
- Will be used by sales representatives on their laptops.
- It will be used for evaluations within the organization.
- Programming language will be Java.
- The programmers are familiar with Java but not with CRM.

The Scale Factors are determined as follows (Estimated by experts depending on the project design with the help of the scale factor table. (see Appendix)):
Precedentedness is nominal (means *somewhat unprecedented* → 3.72), the Development Flexibility is high (means *general conformity* → 2.03), the Architecture / Risk Resolution is nominal (means *often (60%)* → 4.24), the Team Cohesion is high (means *largely cooperative* → 2.19) and the Process Maturity is nominal (means *SW-CMM Level 2* → 4.68).
The scaling exponent of equation *4* will be calculated in this way:

$$E = 0.91 + 0.01 * (3.72 + 2.03 + 4.24 + 2.19 + 4.68) = 1.0786 \quad (9)$$

The Cost Drivers are determined with the following values (Effort Multiplier) (see Appendix): Required Software Reliability is low (0.92), Database Size is high (1.14), Product Complexity is nominal (1.00), Developed for Reusabiliyt is very high (1.15), Documentation is nominal (1.00), Time execution constraint is nominal (1.00), Main storage Constraint is high (1.05), Platform Volatility is low (0.87), Analyst Capability is nominal (1.00), Programmer Capability is high (0.88), Personnel Continuity is nominal (1.00), Applications Experience is low (1.10), Platform Experience is high (0.91), Language and Tool Experience is high (0.91), Use of Software Tools is high (0.90), Multisite Development is very high (0.86), Required Development Schedule is nominal (1.00). The Product of the Effort Multipliers is:

$$\prod EM_i = 0.92 * 1.14 * 1.00 * ... = 0.684 \quad (10)$$

The size in thousand lines of code was estimated with 40.8 and the Effort equation *3* will be calculated as follows:

$$Effort = 2.94 * 40.8^{1.078} * 0.684 = \mathbf{109.57}\ person\ months \quad (11)$$

As result, the project will be estimated with 109.57 person months.

$$TDEV = 3.67 * 109.57^{0.28 + 0.2 * (1.0786 - 0.91)} = \mathbf{16.01}\ months \quad (12)$$

The time for the project will be estimated with nearly 16 months.

[41] comp. [Wie]

3 Conclusion

The calculation of sequences of activities is an important and complex field during the development of software projects.
It is necessary to estimate the time and cost values very exact to have a good basis for the development process. Otherwise, under- or overrated project costs can have negative influence in the single phases of the project development and the negotiations with the customer.

The available estimation methods have different approaches and complexities. Depending on the respective situation, the management has to select the best method for the estimation process. The individual methods have variable estimation accuracies dependent on the project characteristics. So, they are more or less suitable for the respective project. Therefore, individual experiences with the estimation methods and continual improvement of the estimation during the project development process are incessant for a good and realistic estimation of the project costs.

Cocomo II is a very complex and advanced estimation method which considers many factors and condition of the software project. That makes it possible to adjust the method very exactly to the individual features of the project. As a consequence, the results of the Cocomo II estimation equations are much more dependent on the project.
On the other side, Cocomo II is very complicated and needs a lot of preparatory work for the different estimation parts of the equations. Therefore, the analysis of the project has to be complete and exact.
Furthermore, the persons who use the method need a lot of experiences with the model and a detailed introduction into the estimation method.

Appendix

COCOMO II Scale Factors and Effort Multipliers [42]

Drivers	Symbol	XL	VL	L	N	H	VH	XH	Productivity Range
Scale Factors									
PREC	SF1		6.20	4.96	3.72	2.48	1.24	0.00	1.33
PLEX	SF2		5.07	4.05	3.04	2.03	1.01	0.00	1.26
RESL	SF3		7.07	5.65	4.24	2.83	1.41	0.00	1.39
TEAM	SF4		5.48	4.38	3.29	2.19	1.10	0.00	1.29
PMAT	SF5		7.80	6.24	4.68	3.12	1.56	0.00	1.43
Early Design Effort Multipliers									
RCPX	EM1	0.49	0.60	0.83	1.00	1.33	1.91	2.72	5.55
RUSE	EM2			0.95	1.00	1.07	1.15	1.24	1.31
PDIF	EM3			1.00	1.00	1.00			1.00
PERS	EM4	2.12	1.62	1.26	1.00	0.83	0.63	0.50	4.24
PREX	EM5	1.59	1.33	1.12	1.00	0.87	0.74	0.63	2.56
FCIL	EM6	1.43	1.30	1.10	1.00	0.87	0.73	0.62	2.31
SCED	EM7		1.43	1.14	1.00	1.00	1.00		1.43
Post - Architecture Effort Multipliers									
RELY	EM1		0.82	0.92	1.00	1.10	1.26		1.54
DATA	EM2			0.90	1.00	1.14	1.28		1.42
CPLX	EM3		0.73	0.87	1.00	1.17	1.34	1.74	2.38
RUSE	EM4			0.95	1.00	1.07	1.15	1.24	1.31
DOCU	EM5		0.81	0.91	1.00	1.11	1.23		1.52
TIME	EM6				1.00	1.11	1.29	1.63	1.63
STOR	EM7				1.00	1.05	1.17	1.46	1.46
PVOL	EM8			0.87	1.00	1.15	1.30		1.49
ACAP	EM9		1.42	1.19	1.00	0.85	0.71		2.00
PCAP	EM10		1.34	1.15	1.00	0.88	0.76		1.76
PCON	EM11		1.29	1.12	1.00	0.90	0.81		1.51
APEX	EM12		1.22	1.10	1.00	0.88	0.81		1.51
PLEX	EM13		1.19	1.09	1.00	0.91	0.85		1.40
LTEX	EM14		1.20	1.09	1.00	0.91	0.84		1.43
TOOL	EM15		1.17	1.09	1.00	0.90	0.78		1.50
SITE	EM16		1.22	1.09	1.00	0.93	0.86	0.80	1.53
SCED	EM 17		1.43	1.14	1.00	1.00	1.00		1.43

Abbreviations[43]

[42] ibidem [Boe00, cover]

[43] XL .. extra low, VL .. very low, L .. low, N .. nominal, H .. high, VH .. very high, XH .. extra high

References

[Bec00] Jörg Becker. *Prozessmanagement*. Springer - Verlag, 1st edition, 2000.

[Ber04] Prof. Dr. B. Berten. *Lecture notes*, 2003 / 2004.
`http://www.uh.edu/ jbutler/anon/carr.html`.

[Boe] Barry Boehm. *Software Cost Estimation*. 09.05.2005.
`http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/SampleChapters/ch26.pdf`.

[Boe00] Barry Boehm. *Software Cost Estimation with Cocomo II*. Prentice Hall, 1st edition, 2000.

[Bro99] Rolf Bronner. *Planung und Entscheidung*. Oldenbourg Verlag München Wien, 3rd edition, 1999.

[Gli] Martin Glinz. *Software - Aufwandschätzung*. 02.05.2005.
`http://www.ifi.unizh.ch/groups/req/ftp/se_I/kapitel_05.pdf`.

[IES] Fraunhofer IESE. *Cocomo - Methode*. 12.05.2005.
`http://www.software-kompetenz.de/?4774`.

[Kam] Christian Kamm. *Systematische Aufwandsschätzung für Software im Fahrzeug*. 24.05.2005.
`http://www.sdm.de/web4archiv/objects/download/fachartikel/sdm_pub_os_siedersleben.pdf`.

[Krc03] Helmut Krcmar. *Informationsmanagement*. Springer - Verlag, 3rd edition, 2003.

[Lau03] Helmut Laux. *Entscheidungstheorie*. Springer - Verlag, 5th edition, 2003.

[Sch97] Günter Schmidt. *Prozessmanagement - Modelle und Methoden*. Springer - Verlag, 1st edition, 1997.

[Sch98] Jochen Schwarze. *Informationsmanagement*. Verlag Neue Wirtschafts - Briefe / Herne Berlin, 1st edition, 1998.

[Sul] Sencer Sultanoğlu. *Cost Estimation Models*. 19.05.2005.
`http://yunus.hun.edu.tr/~ sencer/cocomo.html`.

[Wie] Wirtschaftsuniversität Wien. *Praxisbeispiel*. 24.05.2005.
`http://wwwai.wu-wien.ac.at/koch/lehre/oss-ws-01/schaetzung/node10.html`.

[Wil] Horst Wildemann. *Management Software Engineering*. 28.04.2005.
`http://www.forsoft.de/publikationen/JL01.pdf`.

Index