

Metrics News, Vol. 4, Nr. 2, December 1999, pp. 29 - 37

Estimation of IT-Projects

Highlights of a Workshop

Manfred Bundschuh, CFPS, President of DASMA

- ☺ Object of estimation
- ☺ Timing of estimation
- ☺ Accuracy of estimation
- ☺ Effort for estimation
- ☺ Monitoring/tracking of estimation
- ☺ Tools for estimation
- ☺ Honesty of estimation
- ☺ Experience of estimation
- ☺ Culture for estimation

Estimation of IT-Projects

Estimation is influenced by uncertainty and inaccuracy.

A prerequisite for estimation is an object to be estimated.

The greatest danger during Estimation is management's persistent claim for Estimation too early in the process leading to the result that Estimation is mistaken for bargaining.

Before project start, the estimation of effort, costs, dates and duration is **the basis for profound planning** as well as for the measurement of project success. **Estimation before project start** asks for know-how-collections from corporate planning, where project risks from market trends and violation of trends as well as technological developments and scenarios are available from experiences of past project portfolios.

Estimation is a process influenced by conflicts due to resistance (not having a mind to estimate or to make a commitment for persons not available), which leads to the question of **honesty of estimation.**

Estimation delivers figures for planning which can be tracked continuously. That is the basis for measuring success. **It's a paradox that project leaders don't measure enough when evaluating their projects.**

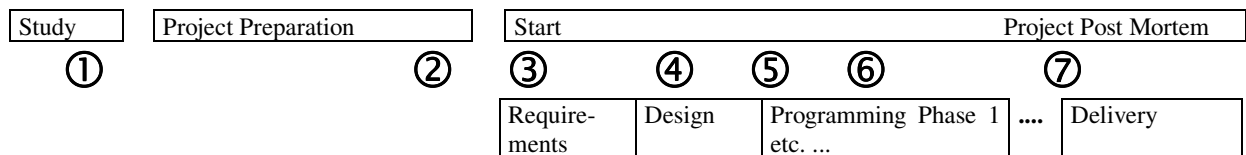
We have learned from publications that (mostly larger) software projects exceed the effort or dates to an amount of 300 to 1,500 %. In workshops and on conferences, a deviation of 10 – 20 % - from the first estimation – is said to be a successful estimation.

Pragmatic Hints

Some **pragmatic hints** which help to sharpen the consciousness for the problem of estimation:

- The earlier an estimation – the larger is the inaccuracy of the estimation.
- Every estimation is better than no estimation.
- The better estimations are documented – the better is the chance to gain experience in estimation.
- The more documented estimations are available – the better projects can be estimated.
- The more precise information about an object to be estimated is available – the more precise the estimation can be.
- The estimated objects should be kept small and the work units independent.
- The communication factor will mostly be forgotten.
- A 1:1 transferable formulae for estimation doesn't exist.
- Estimation should help in decision making and shouldn't be an end in itself.

The requirements for **controlling estimations** should be met; i.e., estimations should be repeated with growing knowledge during project progress (**tracking**), in order to actualize and make precise the estimation and to document the experiences for future estimations. Only through such a consequent **management of estimation** can **expertise in estimation** be gained.



Legend: ① = raw estimation
 ② = liable estimation (base for measurement of Project success)
 ③ ... ⑦ = tracking of estimation
 ⑦ = documentation of estimation experience

Basic Parameters of Estimation

From the above mentioned premise can be learned, **that an estimation**

- should be repeated;

A follow up estimation allows - with more precise information – a better estimate. Comparison with the preceding estimation delivers experiences for future estimations. A continuous tracking of the estimation allows the installation of an **early warning system for deviations** and supports the transparency of actual changes (e.g., measurement of requirements creep, usually about 1 – 3 % each month of project duration).

-should be performed in more than one variant;

The use of multiple estimation methods allows comparison of estimations from diverse viewpoints, reduces the inaccuracy of the estimation and reinforces the sensitivity.

- should be queried critically;

In any case, the parameters of an estimation must be transparent because they strongly influence the result of estimation a priori. Developments in client-server environments or host programming with 4GL-languages must be estimated differently from usual host COBOL developments. Large companies are different from small firms with only few staff. It must be clearly documented for Lines of Code, if comments in programs are counted or when using generators – the generated commands. Generally, standards must be provided for time accounting; e.g., how many hours in a person day, person month or person year.

- should be controllable;

Only controllable estimations give the chance to compare and allow a **feed forward** (learning from past estimations for future estimations).

- should be documented;

The main problem of estimation is the lack of available documentation and hence the lack of experience from past estimations.

The better the estimations are documented –

the more precise estimations can be and expertise in estimation be gained.

Rules of Thumb

Rules of thumb are not exact!

Rules of thumb shouldn't be used as liable standards!

Rules of thumb are easily applicable and can be used to ensure plausibility of estimations!

Rules of thumb are well known for their inaccuracy!

- A Function Point counter can count approximately 1,000 Function Points (FP) per day.
- The productivity of a programmer is approximately 5 FP per person month (PM).
- 1 Function Point corresponds to about 100 LOC (Lines Of Code) of COBOL - 20 LOC for OO or generators and more than 300 LOC of Assembler.
- With 20 Project members, a critical limit of team size is reached.
- Between 500 and 700 FPs the productivity ratio increases more than in other ranges.
- The quantity of Function Points increases about 1 – 3 % per month of project duration (creeping requirements scope); i.e., a project duration of a year means that at the end of the project 24 % more effort has been accomplished as there was demanded in the original requirements.
- The recommended quantity of test cases can be computed by the quantity of FPs to the exponential power of 1.2.
- Each test case will be performed about 4 times during project progress.
- The error potential of a project consists of the sum of errors in all project phases, in coding, in user documentation and errors arising from corrections of errors.
- The error potential of a new development project can be calculated from the quantity of Function Points to the exponential power of 1.25; enhancement projects: quantity of Function Points to the exponential power of 1.27 (greater exponent due to latent errors in the base system).
- With each review, each inspection or each test step about 30 % of the existing errors can be found. Thus high quality can be reached by 6 to 12 of such subsequent quality measures.

- The staff required for maintenance of an application amounts to the quantity of Function Points divided by 500. One person can maintain and enhance (in small limits) about 500 Function Points.
- The project duration (in calendar months) is about the quantity of FPs to the exponential power of 0.4.
- The staff required for a project amounts to the quantity of Function Points divided by 150 to 200.
- Project duration times quantity staff accounts to approximately the quantity of person months.

Source of information: **Capers Jones' books and his article: "Software Estimating Rules Of Thumb", in the periodical IEEE Computer, "Software Challenges" column, July 28th, 1995**

Estimation Methods

Most estimation methods deliver as result a figure as measure for the size of the object to be estimated. Based on this a time relevant figure (effort) is elaborated from which costs can be derived. The total effort should be divided into the project phases according to a **percentage method**. The high esteem of the total effort from the actual effort of the first project phase due to the percentage method can parallel be used as comparative estimation.

There exist many known and valuable estimation methods. We learned from literature that the Function Point method is champion in comparison of the methods. Besides Cocomo (Constructive cost model, LOC-based) is in common usage. In principle this two approaches to estimate effort exist, based on requirements or program size.

There also exist **tools supporting the estimation process as well as different estimation methods** (e.g. Checkpoint, Function Point Workbench etc.).

The problems of the LOC Methods are that LOC are first available in a late phase of the project and that coding makes up only 10% of the size of system development. When the coding phase is reached, there exist some LOC methods for the estimation of the effort for component- and integration test.

The advantage of the Function Point method can be seen from the fact that there exists some variants of the Function Point method: the **Data Point Method**, the **Object Point Method**, **Mark II**, the **Full Function Point Method** and **IFPUG 4.1**. IFPUG is the International Function Point User Group, which posted an international standard for this most commonly used method, which evaluates as object for estimation the requirements analysis document designed from users perspective).

The problem of the Function Point method is that the requirements analysis documents in early phases of the project are not precise enough.

Important is the result of a study performed by Jeffrey, 1987, who found that the effort in projects up to a size of about 10 person years grows approximately linear and exceedingly exponential.

Many estimation methods can be found in literature: the following are the better known:

- The Pi times thumb method

The average from a worst case and a best case estimation is computed. A variant is – A quarter of: the worst case plus best case plus 2 times the average.

- The Analogy method

Comparison of size in LOC of project post mortems.

- The Relational Method

Comparison of indices of project post mortems, e.g. COBOL=100, Assembler=130, PL/I=85 or Skill = 100, 120 or 90.

- The Weightiness method

Estimation with formulas which give different weights to different parameters and / or phases of system development.

- The method of parametrical estimation equations

Estimation with formulas for parameters which strongly influence the effort of system development; e.g., the formula of Putnam (SLIM = Software Life Cycle Management).

- The Multiplier Method

The average productivity of programmers in LOC is multiplied by the estimated LOC, e.g., the Wolverton method.

- The Percentage Method

The effort is relatively divided into the phases of system development, e.g.:

Phase No.	Percentage	Phase	or	Phase No.	Percentage	Phase
1.	10 %	Requirements Analysis		1.	11 %	Requirements Analysis
2.	30 %	Requirements Specification		2.	11 %	Anforderungs-Specification
3.	30 %	DP-Concept		3.	5%	Logical System Specification
4.	25 %	Coding		4.	10 %	Physical Design
5.	5 %	Delivery		5.	46 %	Coding and Module Test
				6.	5 %	Implementation
				7.	12 %	System Test

- The IFA-PASS-Method

A method based on the waterfall model and results from the firm IFA-PASS (Zürich Institute for Automation).

- The COCOMO Method

A three step LOC based method, developed by Barry Boehm from TRW.

Variants of the Function Point Method

All Function Point Variants consist of 3 (or 4) Parts:

- Estimation of entities (objects, functions, transactions)
- Weighting of adjustment factors
- Computation of effort according an "Experience curve"
- (- Computation of an experience curve before introduction of the method)

Following is a short characteristic of each Function Point variant.

- The IFPUG 4.0 Method

Information about the IFPUG 4.0 Function Points is available in the following Internet URL:
<http://www.ifpug.org>

- The Data Point Method

Developed by Harry Sneed from SES (Software Engineering Systems) for estimation based on data objects.

- The Object-Point-Method

Developed by Software AG for estimation of Object Oriented System Development.

- The Mark II Method

Charles Symons developed a method for estimation of effort in 4GL environments. Mark II uses less parameters than IFPUG 4.0 and is easier to use. In smaller projects, there are slight differences to IFPUG 4.0 when measuring the project size.

Mark II was developed according Charles Symons,

- to reduce the subjectivity by counting the quantities of the entities instead of the data sets;
- to measure the development effort instead of the delivered functionality;
- to add 6 complexity factors to the 14 VAFs.

- SPR Function Points

SPR Function Points are a simplification of IFPUG 4.0. The complexity of data objects and transactions is counted as average throughout, and the VAF is based on 2 instead of 14 factors.

- Problem Complexity and Data Complexity.

A specialty of the SPR Function Points is the **approximation** of Function Points if only Parts of the entities are known, they thus can be used,

if only the ILF's (Internal Logical Files) or
only the EIF's (External Interface Files) or
only the EI's (External Inputs) or
only the EO's (External Outputs) or
only the EQ's (External Inquiries) are known.

- Feature Points

Feature Points are also a simplification of IFPUG 4.0. They were developed 1986 by Capers Jones regarding the complexity of the system development process. Besides the data objects (which contribute less) and transactions, Feature Points measure the **algorithms**.

The VAF is also based on 2 instead of 14 factors. The problem of the method is, that The definition of algorithms is not precise enough to use it as a standard.

- 3D Function Points

3D Function Points were developed 1989 by Boeing Computer Services. They measure system development by data, functions and control flux. Data are measured according IFPUG 4.0. The quantity and complexity of functions is added as well as the quantity of control statements (system states and state transitions).

- Full Function Points

Full Function Points were developed during recently by the Software Engineering Management Research Laboratory (UQAM) of the **University in Quebec**, for technical and scientific Applications. It consists of two parts: Function Point counting similar to IFPUG 4.0 and a measurement of the complexity of functions. This is a compromise

with

project leader's complaints about this topic. A public domain version of the FFP Measurement manual is available in the following Internet URL:

<http://www.lrgl.uqam.ca/ffp.html> or <http://www.lmagl.qc.ca>.

- Function Point Prognosis

During the FESMA 1998 Conference in Antwerp Manfred Bundschuh from CNV AG (IT of AXA Colonia Insurance) presented his research of regression analysis on project estimation data based on about 20 projects. During the FESMA 1999 Conference in Amsterdam, he reported that the former results proved to be consistent and only slightly different, based this time on about 40 counts. His main result was, that there is a strong correlation between the sum of the quantity of EI's and EO's (he called IO) and the unadjusted Function Points. He examined the error and found about 15 % deviation. Since estimation is influenced by inaccuracy per se, his formulae for Function Point Prognosis can be used early in system development when only the quantities and not the function points are known, in order to estimate the Function Points. Here are the formulae:

CNV Prognosis		Formula	R ²
1998		FP = 7,3 IO + 56	0,9525
1999	Total	FP = 7,6 IO + 39	0,9509
	PC	FP = 6,5 IO + 134	0,9760
	Host	FP = 7,8 IO + 11	0,9580

The 1998 paper is publicly available in the following Internet URL:

<http://www.gm.fh-koeln.de/~bundschu>, there in Vorträge, FESMA-Vortrag, Antwerpen 1998.

Also the following function proportions (collected by Manfred Bundschuh from different literature in his FESMA 1999 report) can be used for a Function Point prognosis:

Average Function Complexity			EI	EO	EQ	ILF	EIF
	IFPUG		4	5	4	10	7
ISBSG Rel. 5	Total	4,3	5,4	3,8	7,4	5,5	
	Asia Pacific	4,0	5,6	3,9	7,4	5,6	
	Europe	4,2	4,9	3,8	7,2	5,3	
	North America	4,9	5,2	3,8	7,6	5,5	
CNV 1998	All	4,6	5,5	4,3	8,0	5,9	
	PC	4,1	5,7	3,9	7,1	5,3	
	Host	4,9	5,5	4,6	8,5	6,1	
CNV 1999	All	4,6	5,7	4,3	8,2	6,1	
	PC	4,0	5,7	3,9	7,3	5,4	
	Host	4,8	5,7	4,5	8,5	6,2	

The following Function ratios (also collected by Manfred Bundschuh from different literature in his FESMA 1999 report) deliver another possibility for Function Point Prognosis:

Function (Percentage)	Ratios	Number of Projects	EI	EO	EQ	ILF	EIF
CNV 1996		8	34	35	11	18	2
CNV 1997		12	18	43	12	18	9
CNV 1996/97 = FESMA 1998		20	27	39	11	18	5
CNV 1998		19	21	41	13	15	9
CNV Σ (1996-1998)		39	25	39	14	17	6
ISBSG Rel. 5		451	37,2	23,5	13,2	22,3	3,8
Metricviews			26-39	22-24	12-14	24	4-12
Checkpoint			20	24	10	43	3
Nigel Scrivens			33	21	19	23	4
ISBSG Rel. 5 Enhancement Projects		119	36	32	12	15	5
Average of this table:			28,6	26	13,8	20,1	5,8

Nigel Scrivens from shl.com via internet reported industry averages