Estimating, Planning and Risk

Extracted from the "Mythical Man Month" book

$T_{c\,=}\,T_{p\,+}\,T_{d\,+}\,T_{t\,+}\,T_s$

If we assume that T is the total time allowed for the project then:

- T_c = Time to complete the project.
- T_p = Planning time (T/3).
- T_d = development time (T/6).
- T_t = Testing time (T/6).
- T_s = System test time (T/4).

Brooks (1975) provides 2 equations for project estimation

T_c = MMn/N+(MMc/2)*N

Where **T_c** is the time to completion

MMc = average no. of months' effort per (programming) pair

MMn = no. of months' effort required without communications overhead.

This equation assumes you already know the number of developers (N) you will employ.

To calculate the optimum number of developers, the equation below is used.

N_{opt} = 1.41*(MMn/MMc)^{½}

COCOMO

Cost model introduced by Boehm in 1981. Boehm defined three modes of development

Organic: small team, well defined project

Embedded: operates within defined constraints such as regulatory, operational and hardware or software complexity

Semi-detached: Middle ground between organic mode and embedded.

Er = a(KLOC)^b - to calculate the effort required (Er) determined by thousands of lines of code (KLOC) multiplied by constant a and to the power of constant b.

T = c(Er)^d - to calculate the elapsed time (T) based on effort multiplied by constant c, to the power d.

P = Er/T - to calculate the number of developers required (P) which is Effort/time.

Table of constants for each model

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

The Function Point (FP) estimation method

It was developed by Allan Albrecht and first published in 1987 by Kemerer.

This estimation is not dependent on lines of code therefore it can be done earlier on in a SDLC stage.

FP estimation is a 2-step process

First count the user functions (represented as FC in the first equation below)

$FP = FC(PCA)$

secondly adjust for processing complexity (PCA).

$PCA = 0.65 + 0.01\Sigma\ ci$

The initial model provides 5 user function categories:

- External input.
- External output.
- Logical internal file processing.
- External interface to files.
- External inquiries.

In addition, each function can be adjusted by up to 14 complexity factors (ci) that are shown in the table below.

General System Characteristic	Brief Description
Data Communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
Distributed Data Processing	How are distributed data and processing functions handled?
Performance	Did the user require response time or throughput?
Heavily Used Configuration	How heavily used is the current hardware platform where the application will be executed?
Transaction Rate	How frequently are transactions executed daily, weekly, monthly, etc.?

General System Characteristic	Brief Description
On-Line Data Entry	What percentage of the information is entered online?
End-user Efficiency	Was the application designed for end-user efficiency?
Online Update	How many ILFs are updated by online transaction?
Complex Processing	Does the application have extensive logical or mathematical processing?
Reusability	Was the application developed to meet one or many user's needs?
Installation Ease	How difficult is conversion and installation?
Operational Ease	How effective and/or automated are start-up, back-up, and recovery procedures?
Multiple Sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organisations?
Facilitate Change	Was the application specifically designed, developed, and supported to facilitate change?

The table below shows a list of factors. Each factor is rated as to its degree of influence from 1-5.

Rating	Degree of Influence
0	Not present, or no influence
1	Incidental influence
2	Moderate influence
3	Average influence
4	Significant influence
5	Strong influence throughout

Expert Judgement

The following tables are from Cohn (2011).

UUCP = UUCW + UAW shows how to calculate the Unadjusted Use Case Points (UUCP) from the Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW).

The first table below shows the classes of use case points and the second shows the type of actors.

Use Case Complexity	Number of Transactions	Weight
Simple	3 or fewer	5
Average	4 to 7	10
Complex	More than 7	15

Actor Type	Example	Weight
Simple	Another system through an API.	1
Average	Another system through a protocol. A person through a text-based interface.	2
Complex	A person through a graphical user interface.	3

TCF = 0.6 + (0.01*TFactor) shows how to adjust for technical complexity, using the table below (which lists the Tfactors).

Factor	Description	Weight
T1	Distributed system	2
T2	Performance objectives	2
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1

Factor	Description	Weight
T10	Concurrent use	1
T11	Security	1
T12	Access for third parties	1
T13	Training needs	1

EF =1.4 + (-0.03*EFactor) shows how to allow for environmental factors, using the weightings given in the table below.

Factor	Description	Weight
E1	Familiar with the development process	2
E2	Application experience	2
E3	Object-oriented experience	1
E4	Lead analyst capability	1
E5	Motivation	1
E6	Stable requirements	0.5
E7	Part-time staff	0.5
E8	Difficult programming language	2

Finally, **UCP = UUCP*TCF* EF** shows how to calculate the resultant use case points, allowing for the various weighting factors discussed above. This provides an estimate of effort but not duration. Cohn mentions a number of duration estimating approaches but ultimately recommends basing duration estimates on previous projects with similar use case point ratings.

Planning Poker

In this method, the product owner (a business stakeholder) reads a user story to the team and each team writes down an estimate based on the story. The team members then compare their estimates and If they all agree, that becomes the accepted estimate. However, If there is disagreement, there will be further discussions and members will try to reach an agreement.

Risk Management Frameworks

One of the key responsibilities of a SEPM is Risk Management.

There are a number of frameworks that are specific to the risk process, as well as some that mention or encompass risk in some way. These include:

- Open FAIR - a framework produced by the Open Group
- The Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE) framework standard created at Carnegie Mellon University (CMU) in 1999 (Alberts et al, 1999).
- NIST

Risk Identification, Analysis and Mitigation are all part of the Risk Management Process.

To mitigate risks, various kinds of controls can be put in place including physical controls (cameras, physical barriers), procedural controls (policies) and technical controls (IDS, Firewalls)