UNIT IV: PROJECT MANAGEMENT (syllabus)

Estimation – FP Based, LOC Based, Make/Buy Decision, COCOMO II – Planning – Project Plan, Planning Process, RFP Risk Management – Identification, Projection, RMMM – Scheduling and Tracking . Software Project Management: Estimation – LOC and FP Based Estimation, COCOMO Model – Project Scheduling – Scheduling, Earned Value Analysis – Risk Management.

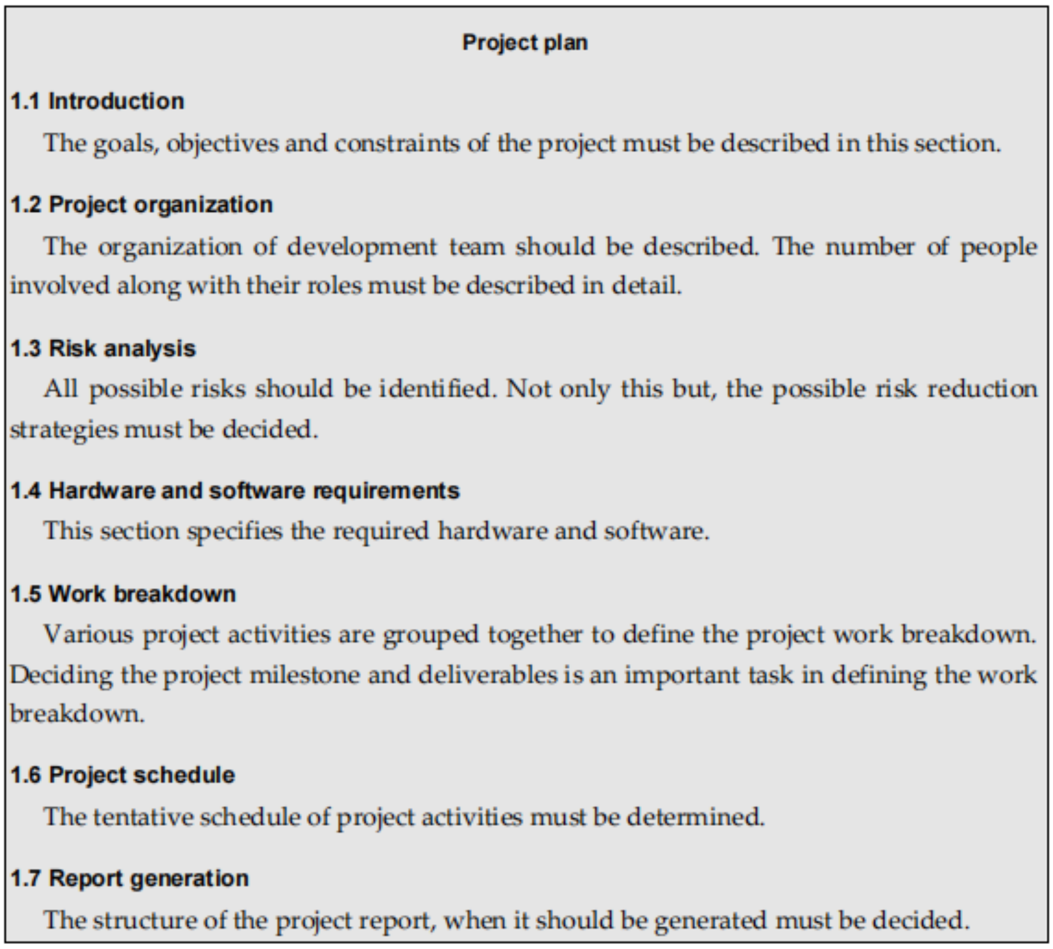
Software Project Management

Management is an essential activity for the computer based systems and product. The term project management involves various activities such as planning, monitoring, control of people, process and various events that occur during the development of software.

Building the software system is a complex activity and many people get involved in this activity for relatively long time. Hence, it is necessary to manage the project. The project management is carried out with the help of **4 P's** i.e. people, product, process and project.

Project Plan

A project plan must be prepared in advance from the available information. The project planning is an **iterative process** and it gets completed only on completion of the project. This process is iterative because new information gets available at each phase of project development. Hence the plan needs to be modified on regular basis for accommodating new requirements of the project.



Project plan

1.1 Introduction

The goals, objectives and constraints of the project must be described in this section.

1.2 Project organization

The organization of development team should be described. The number of people involved along with their roles must be described in detail.

1.3 Risk analysis

All possible risks should be identified. Not only this but, the possible risk reduction strategies must be decided.

1.4 Hardware and software requirements

This section specifies the required hardware and software.

1.5 Work breakdown

Various project activities are grouped together to define the project work breakdown. Deciding the project milestone and deliverables is an important task in defining the work breakdown.

1.6 Project schedule

The tentative schedule of project activities must be determined.

1.7 Report generation

The structure of the project report, when it should be generated must be decided.

Planning Process

The first step to be taken in project management is **Project planning**. There are five major activities that are performed in project planning -

1. Project estimation

2. Project scheduling

3. Risk analysis

4. Quality management planning

5. Change Management planning

Software estimation begins with a description of the **scope of software product.**

For the meaningful project development the scope must be **bounded.** The problem for which the product is to be built is then decomposed into a set of smaller problems. Each of these is estimated using historical data (metrics) and / or previous experience as a guide. The two important issues-**problem complexity** and **risk** are considered before final estimate is made.

Risk Management

Definition of risk : The risk denotes the uncertainty that may occur in the choices due to past actions and risk is something which causes heavy losses.

Definition of risk management : Risk management refers to the process of making decisions based on an evaluation of the factors that threats to the business.

Various activities that are carried out for risk management are -

1. Risk identification
2. Risk projection
3. Risk refinement.
4. Risk mitigation, monitoring and management.

Process of Risk Management

- The risk management process continues until the project gets completed successfully.
- The risk planning phase is required in order to minimize or avoid the risks.
- These risks get monitored and mitigated in the risk monitoring phase.
- After risk mitigation certain amount of risks may remain in the project, the risk planning is required again, in order to minimize or avoid those risks.

- · Thus frequent reviews and corrective actions must be taken in order to make the project with minimized risk. Hence the **risk management is an iterative process**.

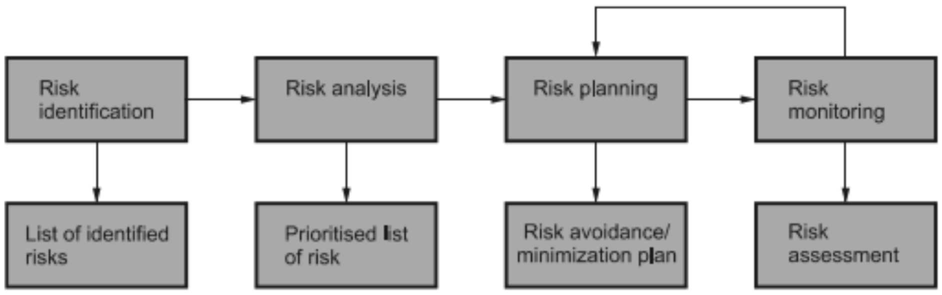


Fig. 5.9.1 Risk management process

From Fig. 5.9.1 risk management is performed in following stages.

1. Risk identification : *In this phase all possible risks are anticipated and a list of potential risks is prepared.*

The risk identification is based on two approaches

1. Generic risk identification - It includes potential threat identification to software project.

2. Product-specific risk identification - It includes product specific threat

identification by understanding people, technology and working environment in which the product gets built.

Step 1 : Preparation of risk item check list

Step 2 : Creating risk components and drivers list.

The set of risk components and drivers list is prepared along with their probability of occurrence. Then their impact on the project can be analysed. Let us understand which are the risk components and drivers.

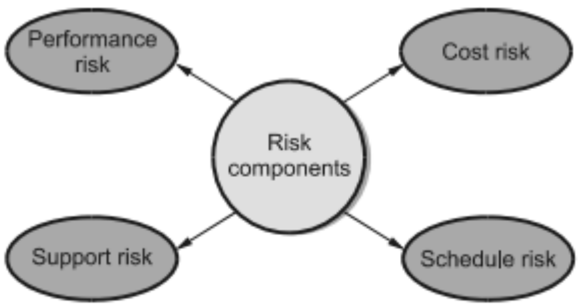


Fig. 5.9.3 Components of risk

2. Risk analysis / Risk Projection : *The after-effects of the risks are obtained and a list is prepared in which, the risks that need to be handled are prioritised.* The risk projection is also called risk estimation.

*3.**Risk planning :** The risk avoidance or risk minimization plan is prepared in this phase.*

4. Risk monitoring : *Identified risks must be mitigated. Hence risk mitigation plan must*

be prepared once the risks are discovered.

Risk monitoring

The **objective** of risk monitoring is

1. To check whether the predicted risks really occur or not.

2. To ensure the steps defined to avoid the risk are applied properly or not.

3. To gather the information which can be useful for analyzing the risk.

Software Risks

There are two characteristics of the risks

1. The risk may or may not happen. It shows the **uncertainty** of the risks.

2. When risks occur, unwanted consequences or **losses** will occur.

Different types of risk

1. Project risk

Project risks arise in the software development process then they basically affect budget, schedule, staffing, resources, and requirements. When project risks become severe then the total cost of project gets increased.

2. Technical risk

These risks affect quality and timeliness of the project. If technical risks become reality then potential design implementation, interface, verification and maintenance problems gets created. Technical risks occur when problem becomes harder to solve.

3. Business risk

When feasibility of software product is in suspect then business risks occur. Business risks can be further categorized as -

i) Market risk - When a quality software product is built but if there is no customer for this product then it is called market risk (i.e. no market for the *product*).

ii) Strategic risk - When a product is built and if it is not following the company's business policies then such a product brings strategic risks.

iii) Sales risk - When a product is built but how to sell is not clear then such a situation brings sales risk.

iv) Management risk - When senior management or the responsible staff leaves the organization then management risk occurs.

v) Budget risk - Losing the overall budget of the project is called budget risk.

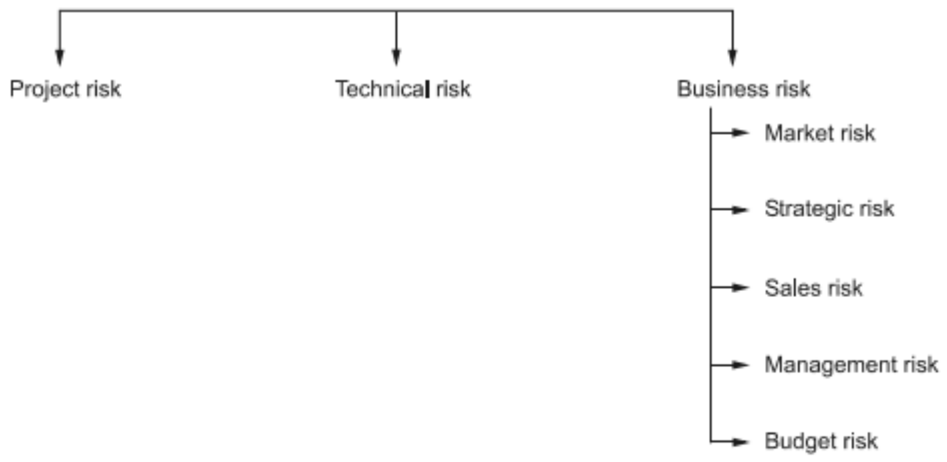


Fig. 5.9.2 Categorization of risk

COCOMO is one of the most widely used software estimation models in the world. This model is developed in 1981 by Barry Boehm to give an estimate of the number of man-months it will take to develop a software product. COCOMO predicts the efforts and schedule of a software product based on size of the software.

COCOMO stands for "COnstructive COst MOdel".

COCOMO has three different models that reflect the complexity -

1. Basic model

2. Intermediate model

3. Detailed model

- Similarly there are three classes of software projects.
  1) Organic mode : In this mode, relatively small, simple software projects with a small
  team are handled. Such a team should have good application experience to less rigid requirements.

- 2) Semi-detached projects : In this class an intermediate projects in which teams with mixed experience level are handled. Such projects may have mix of rigid and less than rigid requirements.

- 3) Embedded projects : In this class, projects with tight hardware, software and operational constraints are handled.

Software projects	a_b	b_b	c_b	d_b
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

- 1) Basic model : The basic COCOMO model estimates the software development effort
  using only Lines of Code. Various equations in this model are -
- $E = a_b(KLOC)^{b_b}$
  $D = C_b(E)^{d_b}$
- $P = E/D$
- Where E is the effort applied in person-months.
  D is the development time in chronological months.
  KLOC means kilo line of code for the project.
  P is total number of persons required to accomplish the project.
  The coefficients a_b , b_b , c_b , d_b for three modes are as given below.

Software projects	a_b	b_b	c_b	d_b
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

Table 5.4.1

-

Merits of basic COCOMO model

Basic COCOMO model is good for quick, early, rough order of magnitude estimates of software project.

Limitations of basic model

1. The accuracy of this model is limited because it does not consider certain factors for cost estimation of software. These factors are hardware constraints, personal quality, and experience, modern tachniques and tools.

2. The estimates of COCOMO model are within a factor of 1.3 only 29 % of the time and within the factor of 2 only 60 % of time.

Example

Consider a software project using semi-detached mode with 30,000 lines of code. We will obtain estimation for this project as follows -

i) Effort estimation

i.e. $E = a_b(KLOC)^{b_b}$

$3.0\ (30)^{1.12}$ where lines of code = 30000 = 30 KLOC 135 person-month

ii) Duration estimation

$D = C_b(E)^{d_b}$

$2.5(135)^{0.35}$ 14 months

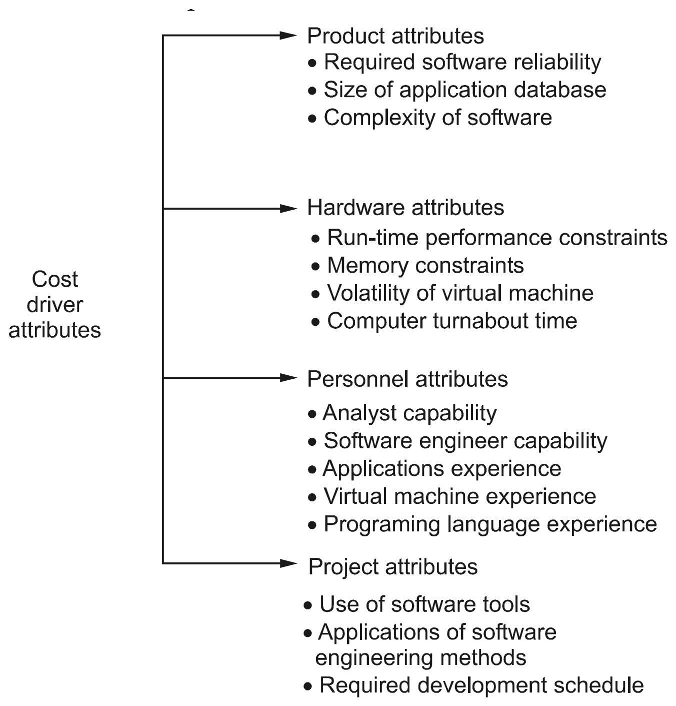
iii) Persons estimation

P = E/D
= 135/14

P = 10 persons approximately

2) Intermediate model

This is an extension of Basic COCOMO model. This estimation model makes use of set of "Cost driver attributes" to compute the cost of software.


Cost driver attributes
Product attributes
• Required software reliability
• Size of application database
• Complexity of software
Hardware attributes
• Run-time performance constraints
• Memory constraints
• Volatility of virtual machine
• Computer turnabout time
Personnel attributes
• Analyst capability
• Software engineer capability
• Applications experience
• Virtual machine experience
• Programing language experience
Project attributes
• Use of software tools
• Applications of software engineering methods
• Required development schedule

Now these 15 attributes get a 6- point scale ranging from "very low" to "extra high". These ratings can be viewed as

The effort multipliers for each cost driver attribute is as given in following table. The product of all effort multipliers result in "Effort Adjustment Factor" (EAF).

The formula for effort calculation can be -

$E = a_i \, (KLOC)^{b_i} \cdot EAF$ person-months

The values for a_i and b_i for various class of software projects are -

Software project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Table 5.4.3

The duration and person estimate is same as in basic COCOMO model. i.e.

$D = c_b (E)^{db}$ months i.e. use values of c_b and d_b coefficients that are in Table

P = E/D persons

Merits of intermediate model

1. This model can be applied to almost entire software product for easy and rough cost estimation during early stage.

2. It can also be applied at the software product component level for obtaining more accurate cost estimation.

Limitations of intermediate model

1. The estimation is within 20 % of actual 68 % of the time. 2. The effort multipliers are not dependent on phases.
3. A product with many components is difficult to estimate.

3) Detailed COCOMO model

The detailed model uses the same equations for estimation as the Intermediate Model. But detailed model can estimate the effort (E), duration (D) and persons (P) of each of development phases, subsystems, modules.

The experimentation with different development strategies is allowed in this model. Four phases used in detailed COCOMO model are -

1. Requirements Planning and Product Design (RPD)

2. Detailed Design (DD)

3. Code and Unit Test (CUT)

4. Integrate and Test (IT)

The effort multipliers for detailed COCOMO are

Phases	Very low	Low	Nominal	High	Very high
RPD	1.80	0.85	1.00	0.75	0.55
DD	1.35	0.85	1.00	0.90	0.75
CUT	1.35	0.85	1.00	0.90	0.75
IT	1.50	1.20	1.00	0.85	0.70

Using these detailed cost drivers, an estimate is determined for each phase of the lifecycle.

COCOMO II is applied for modern software development practices addressed for the projects in 1990's and 2000's.

The sub-models of COCOMO II model are -

1. Application composition model

- For estimating the efforts required for the prototyping projects and the projects in which the existing software components are used application-composition model is introduced.
- The estimation in this model is based on the number of application points. The application points are similar to the object points.
- This estimation is based on the level of difficulty of object points. Boehm has suggested the object point productivity in the following manner.

Developers experience and capability	Very low	Low	Nominal	High	Very high
CASE maturity	Very low	Low	Nominal	High	Very high
Productivity (NOP/Month)	4	7	13	25	50

Effort computation in application-composition model can be done as follows - $PM = (NAP^{(1-\%reuse/100)}) / PROD$

where
PM means effort required in terms of person-months.

NAP means number of application points required.

% reuse indicates the amount of reused components in the project.

These reusable components can be screens, reports or the modules used in previous projects.

PROD is the object point productivity. These values are given in the above table.

2. An early design model

- This model is used in the early stage of the project development. That is after gathering the user requirements and before the project development actually starts, this model is used. Hence approximate cost estimation can be made in this model.

- The estimation can be made based on the functional points.

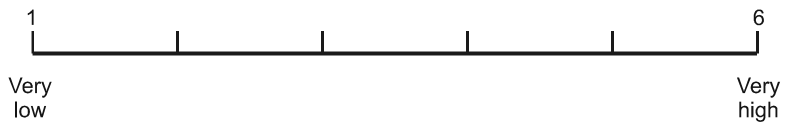
- In early stage of development different ways of implementing user requirements
can be estimated.

- The effort estimation (in terms of person month) in this model can be made using the following formula :
$Effort = A . size^{B} \cdot M$
where
Boehm has proposed the value of coefficient A = 2.94.
Size should be in terms of Kilo source lines of code i.e. KSLOC.

The lines of code can be computed with the help of function point. The value of B is varying from 1.1 to 1.24 and depends upon the project. M is based on the characteristics such as

1. Product reliability and complexity (RCPX)
2. Reuse required (RUSE)
3. Platform difficulty (PDIF)
4. Personnel capability (PERS)
5. Personnel experience (PREX)
6. Schedule (SCED)
7. support facilities (FCIL)

These characteristics values can be computed on the following scale –



- Hence the effort estimation can be given as

$$\mathbf{PM = 2.94 \times size^B \times M}$$

$$\mathbf{M = RUSE \times PDIF \times PERS \times PREX \times SCED \times FCIL}$$

3. A reuse model

- This model considers the systems that have significant amount of code which is reused from the earlier software systems. The estimation made in reuse model is nothing but the efforts required to integrated the reused models into the new systems.

- There are two types of reusable codes : black box code and white box code. The black box code is a kind of code which is simply integrated with the new system without modifying it. The white box code is a kind of code that has to be modified to some extent before integrating it with the new system, and then only it can work correctly.

- There is third category of code which is used in reuse model and it is the code which can be generated automatically. In this form of reuse the standard templates are integrated in the generator. To these generators, the system model is given as input from which some additional information about the system is taken and the code can be generated using the templates.

- The efforts required for automatically the generated code is

$PM = (ASLOC \times AT/100)/ATPROD$

where
AT is percentage of automatically generated code.
ATPROD is the productivity of engineers in estimating such code

- Sometimes in the reuse model some white box code is used along with the newly developed code. The size estimate of newly developed code is equivalent to the reused code. Following formula is used to calculate the effort in such a case -

- $ESLOC = ASLOC \times (1 - AT/100) X AAM$ where
  ESLOC means equivalent number of lines of new source code.

ASLOC means the source lines of code in the component that has to be adapted.

AAM is adaptation Adjustment multiplier. This factor is used to take into account the efforts required to reuse the code.

4. Post architecture model

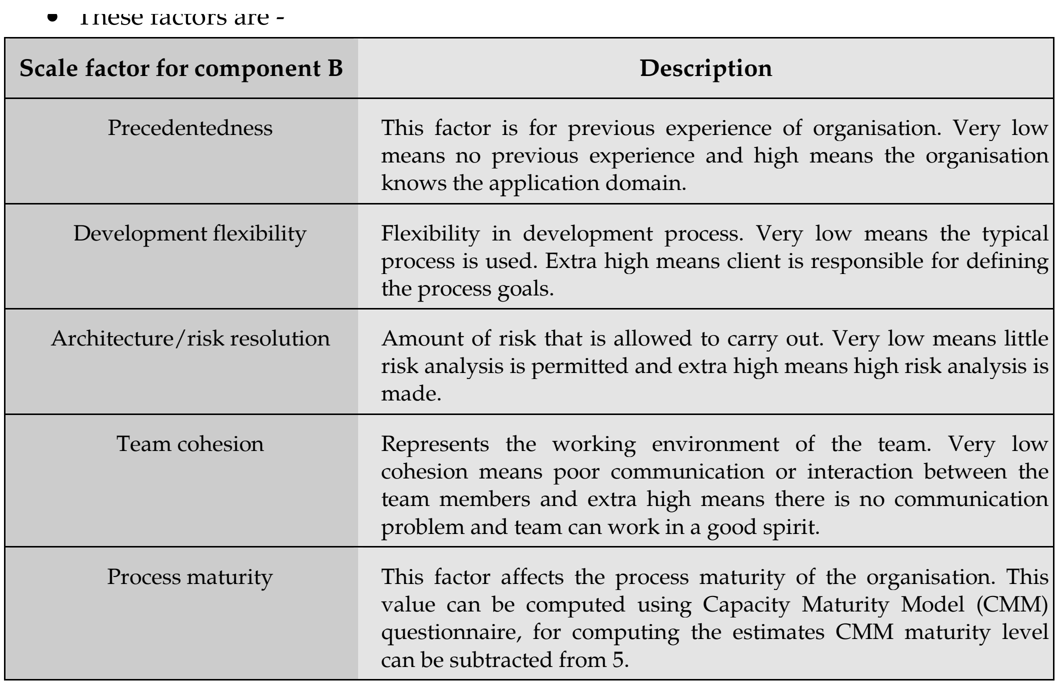
This model is a detailed model used to compute the efforts. The basic formula used in this model is

$Effort = A . Size^{B} \cdot M$

In this model efforts should be estimated more accurately. In the above formula A is the amount of code. This code size estimate is made with the help of three components -

The estimate about new lines of code that is added in the program. Equivalent number of source lines of code (ESLOC) used in reuse model.

Due to changes in requirements the lines of code get modified. The estimate of amount of code being modified.



- These factors are -

Scale factor for component B	Description
Precedentedness	This factor is for previous experience of organisation. Very low means no previous experience and high means the organisation knows the application domain.
Development flexibility	Flexibility in development process. Very low means the typical process is used. Extra high means client is responsible for defining the process goals.
Architecture/risk resolution	Amount of risk that is allowed to carry out. Very low means little risk analysis is permitted and extra high means high risk analysis is made.
Team cohesion	Represents the working environment of the team. Very low cohesion means poor communication or interaction between the team members and extra high means there is no communication problem and team can work in a good spirit.
Process maturity	This factor affects the process maturity of the organisation. This value can be computed using Capacity Maturity Model (CMM) questionnaire, for computing the estimates CMM maturity level can be subtracted from 5.

The exponent term B has three possible values that are related to the levels of project complexity. The values of B are continuous rather than discrete. It depends upon the five scale factors. These scale factors vary from very low to extra high (i.e. from 5 to 0).