Unit-II

(11) copy

Project Planning

T142 02
34

Project Planning Objectives

The objective of software project planning is to provide a framework that enables the manager to make reasonable estimates of:

- Resources
- Cost, and
- Schedule

Objectives

These estimates are made within a limited time frame at the beginning of a software project and should be updated regularly as the project progresses.

In addition, estimates should attempt to define best case and worst-case scenarios so that project outcomes can be bounded.

Planning is one of the most important management activities and is an ongoing effort throughout the life of the project.

Software project management begins with a set of activities that are collectively called Project Planning.

The software project planner must estimate following things before a project begins:

(i). How much will it cost?
(ii). How long will it take?
(iii). How many people will it take?
(iv). What might go wrong?

Before starting a software project, it is essential to determine the tasks to be performed and properly manage allocation of tasks among individuals involved in the software development. Hence, planning is important as it results in effective software development.

Project planning is an organized and integrated management process, which focuses on activities required for successful completion of the project. It prevents obstacles that arise in the project such as changes in projects or organization's objectives, non-availability of resources, and so on. Project planning also helps in better utilization of resources and optimal usage of the allotted time for a project. The other objectives of project planning are listed below.

- It defines the roles and responsibilities of the project management team members.
- It ensures that the project management team works according to the business objectives.

- It checks feasibility of the schedule and user requirements.
- It determines project constraints.

Several individuals help in planning the project. These include senior management and project management team. Senior management is responsible for employing team members and providing resources required for the project. The project management team, which generally includes project managers and developers, is responsible for planning, determining, and tracking the activities of the project. Table lists the tasks performed by individuals involved in the software project.

Tasks of Individuals involved in Software Project

Senior Management	Project Management Team
Approves the project, employ personnel, and provides resources required for the project. Reviews project plan to ensure that it accomplishes the business objectives. Resolves conflicts among the team members. Considers risks that may affect the project so that appropriate measures can be taken to avoid them.	Reviews the project plan and implements procedures for completing the project. Manages all project activities. Prepares budget and resource allocation plans. Helps in resource distribution, project management, issue resolution, and so on. Understands project objectives and finds ways to accomplish the objectives. Devotes appropriate time and effort to achieve the expected results. Selects methods and tools for the project.

Project planning should be effective so that the project begins with well-defined tasks. Effective project planning helps to minimize the additional costs incurred on the project while it is in progress. For effective project planning, some principles are followed. These principles are listed below.

- **Planning is necessary:** Planning should be done before a project begins. For effective planning, objectives and schedules should be clear and understandable.
- **Risk analysis:** Before starting the project, senior management and the project management team should consider the risks that may affect the project. For example, the user may desire changes in requirements while the project is in progress. In such a case, the estimation of time and cost should be done according to those requirements (new requirements).
- **Tracking of project plan:** Once the project plan is prepared, it should be tracked and modified accordingly.
- **Meet quality standards and produce quality deliverables:** The project plan should identify processes by which the project management team can ensure quality in software. Based on the process selected for ensuring quality, the time and cost for the project is estimated.

- **Description of flexibility to accommodate changes:** The result of project planning is recorded in the form of a project plan, which should allow new changes to be accommodated when the project is in progress.

Project planning comprises project purpose, project scope, project planning process, and project plan. This information is essential for effective project planning and to assist project management team in accomplishing user requirements.

Project Purpose

Software project is carried out to accomplish a specific purpose, which is classified into two categories, namely, project objectives and business objectives. The commonly followed project objectives are listed below.

- **Meet user requirements:** Develop the project according to the user requirements after understanding them.
- **Meet schedule deadlines:** Complete the project milestones as described in the project plan on time in order to complete the project according to the schedule.
- **Be within budget:** Manage the overall project cost so that the project is within the allocated budget.
- **Produce quality deliverables:** Ensure that quality is considered for accuracy and overall performance of the project.

Business Software Engineering

Business objectives ensure that the organizational objectives and requirements are accomplished in the project. Generally, these objectives are related to business process improvements, customer satisfaction, and quality improvements. The commonly followed business objectives are listed below.

- **Evaluate processes:** Evaluate the business processes and make changes when and where required as the project progresses.
- **Renew policies and processes:** Provide flexibility to renew the policies and processes of the organization in order to perform the tasks effectively.
- **Keep the project on schedule:** Reduce the downtime (period when no work is done) factors such as unavailability of resources during software development.
- **Improve software:** Use suitable processes in order to develop software that meets organizational requirements and provides competitive advantage to the organization.

optional

Project Scope

With the help of user requirements, the project management team determines the scope of the project before the project begins. This scope provides a detailed description of functions, features, constraints, and interfaces of the software that are to be considered. Functions describe the tasks that the software is expected to perform. Features describe the attributes required in the software as per the user requirements. Constraints describe the limitations imposed on software by hardware, memory, and so on. Interfaces describe the interaction of software components (like modules and functions) with each other. Project scope also considers software performance, which in turn depends on its processing capability and response time required to produce the output.

Once the project scope is determined, it is important to properly understand it in order to develop software according to the user requirements. After this, project cost and duration are estimated. Lf the project scope is not determined on time, the project may not be completed within the specified schedule. Project scope describes the following information.

- The elements included and excluded in the project
- The processes and entities
- The functions and features required in software according to the user requirements.

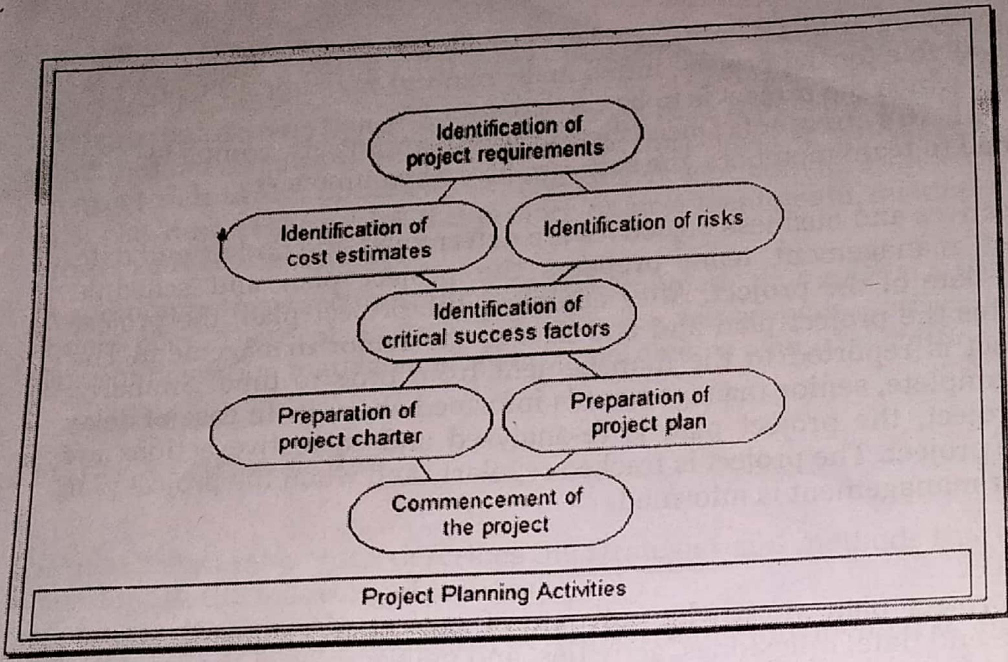
Note that the project management and senior management team should communicate with the users to understand their requirements and develop software according to those requirements and expected functionalities.

Project Planning Process

The project planning process involves a set of interrelated activities followed in an orderly manner to implement user requirements in software and includes the description of a series of project planning activities and individual(s) responsible for performing these activities. In addition, the project planning process comprises the following.

1. Objectives and scope of the project
2. Techniques used to perform project planning
3. Effort (in time) of individuals involved in project
4. Project schedule and milestones
5. Resources required for the project
6. Risks associated with the project.

Project planning process comprises several activities, which are essential for carrying out a project systematically. These activities refer to the series of tasks performed over a period of time for developing the software. These activities include estimation of time, effort, and resources required and risks associated with the project.



Identification of project requirements
Identification of cost estimates
Identification of risks
Identification of critical success factors
Preparation of project charter
Preparation of project plan
Commencement of the project

Project Planning Activities

Project planning process consists of the following activities.

- **Identification of project requirements:** Before starting a project, it is essential to identify the project requirements as identification of project requirements helps in performing the activities in a systematic manner. These requirements comprise information such as project scope, data and functionality required in the software, and roles of the project management team members.
- **Identification of cost estimates:** Along with the estimation of effort and time, it is necessary to estimate the cost that is to be incurred on a project. The cost estimation includes the cost of hardware, network connections, and the cost required for the maintenance of hardware components. In addition, cost is estimated for the individuals involved in the project.
- **Identification of risks:** Risks are unexpected events that have an adverse effect on the project. Software project involves several risks (like technical risks and business risks) that affect the project schedule and increase the cost of the project. Identifying risks before a project begins helps in understanding their probable extent of impact on the project.
- **Identification of critical success factors:** For making a project successful, critical success factors are followed. These factors refer to the conditions that ensure greater chances of success of a project. Generally, these factors include support from management, appropriate budget, appropriate schedule, and skilled software engineers.
- **Preparation of project charter:** A project charter provides a brief description of the project scope, quality, time, cost, and resource constraints as described during project planning. It is prepared by the management for approval from the sponsor of the project.

- **Preparation of project plan:** A project plan provides information about the resources that are available for the project, individuals involved in the project, and the schedule according to which the project is to be carried out.
- **Commencement of the project:** Once the project planning is complete and resources are assigned to team members, the software project commences.

Once the project objectives and business objectives are determined, the project end date is fixed. The project management team prepares the project plan and schedule according to the end date of the project. After analyzing the project plan, the project manager communicates the project plan and end date to the senior management. The progress of the project is reported to the management from time to time. Similarly, when the project is complete, senior management is informed about it. In case of delay in completing the project, the project plan is re-analyzed and corrective actions are taken to complete the project. The project is tracked regularly and when the project plan is modified, the senior management is informed.

Project Plan

As stated earlier, a project plan stores the outcome of project planning. It provides information about the end date, milestones, activities, and deliverables of the project. In addition, it describes the responsibilities of the project management team and the resources required for the project. It also includes the description of hardware and software (such as compilers and interfaces) and lists the methods and standards to be used. These methods and standards include algorithms, tools, review techniques, design language, programming language, and testing techniques.

A project plan helps a project manager to understand, monitor, and control the development of software project. This plan is used as a means of communication between the users and project management team. There are various advantages associated with a project plan, some of which are listed below.

- It ensures that software is developed according to the user requirements, objectives, and scope of the project.
- It identifies the role of each project management team member involved in the project.
- It monitors the progress of the project according to the project plan.
- It determines the available resources and the activities to be performed during software development.
- It provides an overview to management about the costs of the software project, which are estimated during project planning.

Note that there are differences in the contents of two project plans depending on the kind of project and user requirements. Atypical project plan is divided into the following sections.

1. **Introduction:** Describes the objectives of the project and provides information about the constraints that affect the software project.
2. **Project organization:** Describes the responsibilities assigned to the project management team members for completing the project.
3. **Risk analysis:** Describes the risks that can possibly arise during software development as well as explains how to assess and reduce the effect of risks.

4. **Resource requirements:** Specifies the hardware and software required to carry out the software project. Cost estimation is done according to these resource requirements.
5. **Workbreakdown:** Describes the activities into which the project is divided. It also describes the milestones and deliverables of the project activities.
6. **Project schedule:** Specifies the dependencies of activities on each other. Based on this, the time required by the project management team members to complete the project activities is estimated.

In addition to these sections, there are several plans that may be a part of or 'linked to a project plan. These plans include quality assurance plan, verification and validation plan, configuration management plan, maintenance plan, and staffing plan.

Quality Assurance Plan

The quality assurance plan describes the strategies and methods that are to be followed to accomplish the following objectives.

1. Ensure that the project is managed, developed, and implemented in an organized way.
2. Ensure that project deliverables are of acceptable quality before they are delivered to the user.

Verification and Validation Plan

The verification and validation plan describes the approach, resources and schedule used for system validation. The verification and validation plan, which comprises the following sections.

- **General information:** Provides description of the purpose, scope, system overview, project references, acronyms and abbreviations, and points of contact. Purpose describes the procedure to verify and validate the components of the system. Scope provides information about the procedures to verify and validate as they relate to the project. System overview provides information about the organization responsible for the project and other information such as system name, system category, operational status of the system, and system environment. Project references provide the list of references used for the preparation of the verification and validation plan. Acronyms and abbreviations provide a list of terms used in the document. Points of contact provide information to users when they require assistance from organization for problems such as troubleshooting and so on.
- **Reviews and walkthroughs:** Provides information about the schedule and procedures. Schedule describes the end date of milestones of the project. Procedures describe the tasks associated with reviews and walkthroughs. Each team member reviews the document for errors and consistency with the project requirements. For walkthroughs, the project management team checks the project for correctness according to software requirements specification (SRS).
- **System test plan and procedures:** Provides information about the system test strategy, database integration, and platform system integration. System test strategy provides an overview of the components required for integration of the database and ensures that the application runs on at least two specific platforms. Database

integration procedure describes how database is connected to the Graphical User Interface (GUI).Platform system integration procedure is performed on different operating systems to test the platform.

- **Acceptance test and preparation for delivery:** Provides information about procedure, acceptance criteria, and installation procedure. Procedure describes how acceptance testing is to be performed on the software to verify its usability as required. Acceptance criteria describes that software will be accepted only if all the components, features and functions are tested including the system integration testing. In addition, acceptance criteria checks whether the software accomplishes user expectations such as its ability to operate on several platforms. Installation procedure describes the steps of how to install the software according to the operating system being used.

Configuration Management Plan

The configuration management plan defines the process, which is used for making changes to the project scope. Generally, the configuration management plan is concerned with redefining the existing objectives of the project and deliverables (software products that are delivered to the user after completion of software development).

Maintenance Plan

The maintenance plan specifies the resources and processes required for making the software operational after its installation. Sometimes, the project management team (or software development team) does not carry out the task of maintenance. In such a case, a separate team known as software maintenance team performs the task of software maintenance.

The maintenance plan, which comprises the sections listed below.

- **Introduction and background:** Provides a description of software to be maintained and the services required for it. It also specifies the scope of maintenance activities that are to be performed.
- **Budget:** Specifies the budget required for carrying out software maintenance and operational activities.
- **Roles and responsibilities:** Specifies the roles and responsibilities of the team members associated with the software maintenance and operation. It also describes the skills required to perform maintenance and operational activities. In addition to software maintenance team, software maintenance comprises user support, user training, and support staff.
- **Performance measures and reporting:** Identifies the performance measures required for carrying out software maintenance. It also describes how measures required for enhancing the performance of services (for the software) are recorded and reported.
- **Management approach:** Identifies the methodologies that are required for establishing maintenance priorities of the projects. For this purpose, the management either refers to the existing methodologies or identifies new methodologies. Management approach also describes how users are involved in software maintenance

and operations activities as well as how users and project management team communicate with each other.

- **Documentation strategies:** Provides a description of the documentation that is prepared for user reference. Generally, documentation includes reports, information about problems occurring in software, error messages, and the system documentation.
- **Training:** Provides information about the training activities.
- **Acceptance:** Defines a point of agreement between the project management team and software maintenance team after the completion of implementation and transition activities. Once the agreement has been made, the software maintenance begins.

Staffing Plan

The staffing plan describes the number of individuals required for a project. It includes selecting and assigning tasks to the project management team members. It provides information about appropriate skills required to perform the tasks to produce the project deliverables and manage the project. In addition, it provides information of resources such as tools, equipment, and processes used by the project management team.

Staff planning is performed by a staff planner, who is responsible for determining the individuals available for the project. Other responsibilities of a staff planner are listed below.

- The staff planner determines individuals, who can be from existing staff, staff on contract, or newly employed staff. It is important for the staff planner to know the structure of the organization to determine the availability of staff.
- The staff planner determines the skills required to execute the tasks mentioned in the project schedule and task plan. In case staff with required skills is not available, staff planner informs the project manager about the requirements.
- The staff planner ensures that the required staff with required skills is available at the right time. For this purpose, the staff planner plans the availability of staff after the project schedule is fixed. For example, at the initial stage of a project, staff may consist of a project manager and a few software engineers whereas during software development, staff consists of software designers as well as the software developers.
- The staff planner defines roles and responsibilities of the project management team members so that they can communicate and coordinate with each other according to the tasks assigned to them. Note that the project management team can be further broken down into sub-teams depending on the size and complexity of the project.
- The staffing plan comprises the following sections.
- **General information:** Provides information such as name of the project and project manager who is responsible for the project. In addition, it specifies the start and end dates of the project.
- **Skills assessment:** Provides information, which is required for assessment of skills. This information includes the knowledge, skill, and ability of team members who are required to achieve the objectives of the project. In addition, it specifies the number of team members required for the project.
- **Staffing profile:** Describes the profile of the staff required for the project. The profile includes calendar time, individuals involved, and level of commitment. Calendar time specifies the period of time such as month or quarter for which individuals are required to complete the project. Individuals who are involved in the project have specific designations such as project manager and the developer. Level

of commitment is the utilization rate of individuals such as work performed on full-time and part-time basis.

- **Organization chart:** Describes the organization of project management team members. In addition, it includes information such as name, designation, and role of each team member.

Decomposition Techniques

Software project estimation is a form of problem solving, and in most cases, the problem to be solved (i.e., developing a cost and effort estimate for a software project) is too complex to be considered in one piece. For this reason, we decompose the problem, re-characterizing it as a set of smaller (and hopefully, more manageable) problems.

Before an estimate can be made, the project planner must understand the scope of the software to be built and generate an estimate of its "size."

Software Sizing

Software sizing is an activity in software engineering that is used to estimate the size of a software application or component in order to be able to implement other software project management activities (such as estimating or tracking). Size is an inherent characteristic of a piece of software just like weight is an inherent characteristic of a tangible material.

The accuracy of a software project estimate is predicated on a number of things:

(1) The degree to which the planner has properly estimated the size of the product to be built

(2) The ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects)

(3) The degree to which the project plan reflects the abilities of the software team

(4) The stability of product requirements and the environment that supports the software engineering effort.

As project estimate is only as good as the estimate of the size of the work to be accomplished, sizing represents the project planner's first major challenge.

In the context of project planning, size refers to a quantifiable outcome of the software project. If a direct approach is taken, size can be measured in LOC. If an indirect approach is chosen, size is represented as FP.

There are four different approaches to the sizing problem:

1. **"Fuzzy logic" sizing:** This approach uses the approximate reasoning techniques that are the cornerstone of fuzzy logic. To apply this approach, the planner must identify the type of application, establish its magnitude on a qualitative scale, and then refine the magnitude within the original range. Although personal experience can be used, the planner should also have access to a historical database of projects8 so that estimates can be com- pared to actual experience.

2. Function point sizing: The planner develops estimates of the information domain. Its characteristics will be discussed later in the session.

3. Standard component sizing: Software is composed of a number of different "standard components" that are generic to a particular application area. For example, the standard components for an information system are subsystems, modules, screens, reports, interactive programs, batch programs, files, LOC, and object-level instructions. The project planner estimates the number of occurrences of each standard component and then uses historical project data to determine the delivered size per standard component. To illustrate, consider an information systems application. The planner estimates that 18 reports will be generated. Historical data indicates that 967 lines of COBOL [PUT92) are required per report. This enables the planner to estimate that 17,000 LOC will be required for the reports component. Similar estimates and computation are made for other standard components, and a combined size value (adjusted statistically) results.

4.Change sizing: This approach is used when a project encompasses the use of existing software that must be modified in some way as part of a project. The planner estimates the number and type (e.g., reuse, adding code, changing code, and deleting code) of modifications that must be accomplished. Using an "effort ratio" [PUT92) for each type of change, the size of the change may be estimated.

PROCESS BASED ESTIMATION:

The most common technique for estimating a project is to base the estimate on the process that will be used. That is, the process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated. Like the problem based techniques, process based estimation begins with a delineation of software functions obtained from the project scope. A series of software process activities must be performed for each function. Functions and related software process activities may be represented as part of a table similar to the one presented Once problem functions and process activities are melded, the planner estimates the effort (e.g., person-months) that will be required to accomplish each software process activity for each software function. Senior staff heavily involved in early activities are generally more expensive than junior staff involved in later design tasks, code generation.

PROBLEM BASED ESTIMATION:

Lines of code and function points were described as measures from which productivity metrics can be computed. LOC and FP data are used in two ways during software project estimation: (1) as an estimation variable to "size" each element of the software and (2) as baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections LOC and FP estimation are distinct estimation techniques. Yet both have a number of characteristics in common. The project planner begins with a bounded statement of software scope and from this statement attempts to decompose software into problem functions that can each be estimated individually. LOC or FP (the estimation variable) is then estimated for each function. Alternatively, the planner may choose another component for sizing such as classes or objects.

What is COCOMO? Explain COCOMO model in detail.

The Constructive Cost Model (COCOMO) is the most widely used software estimation model in the world. It was developed by Barry Boehm of TRW and first published in his book Software Engineering Economics in 1981. The COCOMO model predicts the effort and duration of a project based on inputs relating to the size of the resulting systems and a number of "cost drives" that affect productivity. The most fundamental calculation in the COCOMO model is the use of the Effort Equation (Equation1) to estimate the number of Person-Months required developing a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity.

COCOMO is defined in terms of three different models:

- The Basic Model
- The Intermediate Model
- The Detailed Model

Basic COCOMO compute software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).

It gives an approximate estimate of the various parameters of the project

COCOMO applies to three classes of software projects:

- Organic projects - "small" teams with "good" experience working with "less than rigid" requirements
- Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
- Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

The basic COCOMO equations take the form

Effort Applied (E) = $a_b(\text{KLOC})^{b}{}_{b}$ [man-months]

Development Time (D) = $c_b(\text{Effort Applied})^{d}{}_{b}$ [months]

People required (P) = Effort Applied / Development Time [count]

where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients a_b, b_b, c_b and d_b are given in the following table (note: the values listed below are from the original analysis, with a modern reanalysis[4] producing different values):

Software project	a_b	b_b	c_b	d_b

SLOC - Source line of code
KLOC - 1000 LOC
KDLOC - 1000 delivered LOC
MLOC - 1,00,00,000 LOC
GLOC - 100,00,00,000 L[illegible]

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

Basic COCOMO is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

Intermediate COCOMOs

Intermediate COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes:-

It refines the above ~~es~~ estimate (Basic cocomo) using 15 multipliers that are based on different s/w developme attribute

- Product attributes
  - Required software reliability extent
  - Size of application database
  - Complexity of the product
- Hardware attributes
  - Run-time performance constraints
  - Memory constraints
  - Volatility of the virtual machine environment
  - Required turnabout time
- Personnel attributes
  - Analyst capability
  - Software engineering capability
  - Applications experience
  - Virtual machine experience
  - Programming language experience
- Project attributes
  - Use of software tools
  - Application of software engineering methods
  - Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an *effort adjustment factor (EAF)*. Typical values for EAF range from 0.9 to 1.4.

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	

Applications experience	1.29	1.13	1.00	0.91	0.82
Software engineer capability	1.42	1.17	1.00	0.86	0.70
Virtual machine experience	1.21	1.10	1.00	0.90	
Programming language experi ence	1.14	1.07	1.00	0.95	
Project attributes					
Application of software engin eering methods	1.24	1.10	1.00	0.91	0.82
Use of software tools	1.24	1.10	1.00	0.91	0.83
Required development sched ule	1.23	1.08	1.00	1.04	1.10

The Intermediate Cocomo formula now takes the form:

$E=a_i(KLoC)^{(b_i)}(EAF)$

where E is the effort applied in person-months, **KLoC** is the estimated number of thousands of delivered lines of code for the project, and **EAF** is the factor calculated above. The coefficient **a_i** and the exponent **b_i** are given in the next table.

Software project	a_i	b_i
Organic	3.2	1.05

Semi-detached	3.0	1.12
Embedded	2.8	1.20

The Development time **D** calculation uses **E** in the same way as in the Basic COCOMO.

Detailed COCOMO[edit]

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

The detailed model uses different effort multipliers for each cost driver attribute. These **Phase Sensitive** effort multipliers are each to determine the amount of effort required to complete each phase. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software life cycle.

A Detailed project schedule is never static.

The Six phases of detailed COCOMO are:-

- planning and requirements
- system design
- detailed design
- module code and test
- integration and test
- Cost Constructive model

It calculate the development time and effort taken by the total of the estimates of all the individual System.